StRE: Self Attentive Edit Quality Prediction in Wikipedia

Soumya Sarkar*[1], Bhanu Prakash Reddy*[2], Sandipan Sikdar[3] Animesh Mukherjee[4]

IIT Kharagpur, India[1,2,4],RWTH Aachen, Germany[3]
soumya015@iitkgp.ac.in[1],bhanu77prakash@gmail.com[2]
sandipan.sikdar@cssh.rwth-aachen.de[3], animesh@cse.iitkgp.ac.in[4]

Abstract

Wikipedia can easily be justified as a behemoth, considering the sheer volume of content that is added or removed every minute to its several projects. This creates an immense scope, in the field of natural language processing toward developing automated tools for content moderation and review. In this paper we propose Self Attentive Revision Encoder (StRE) which leverages orthographic similarity of lexical units toward predicting the quality of new edits. In contrast to existing propositions which primarily employ features like page reputation, editor activity or rule based heuristics, we utilize the textual content of the edits which, we believe contains superior signatures of their quality. More specifically, we deploy deep encoders to generate representations of the edits from its text content, which we then leverage to infer quality. We further contribute a novel dataset containing $\sim 21M$ revisions across $32K$ Wikipedia pages and demonstrate that *StRE* outperforms existing methods by a significant margin – at least 17% and at most 103%. Our pre-trained model achieves such result after retraining on a set as small as 20% of the edits in a wikipage. This, to the best of our knowledge, is also the first attempt towards employing deep language models to the enormous domain of automated content moderation and review in Wikipedia.

1 Introduction

Wikipedia is the largest multilingual encyclopedia known to mankind with the current English version consisting of more than $5M$ articles on highly diverse topics which are segregated into categories, constructed by a large editor base of more than $32M$ editors (Hube and Fetahu, 2019). To encourage transparency and openness, Wikipedia allows anyone to edit its pages albeit with certain guidelines for them[1].

Problem: The inherent openness of Wikipedia has also made it vulnerable to external agents who intentionally attempt to divert the unbiased, objective discourse to a narrative which is aligned with the interest of the malicious actors. Our pilot study on manually annotated 100 Wikipedia pages of four categories (25 pages each category) shows us that at most 30% of the edits are reverted (See Fig 1). Global average of number of reverted damaging edits is $\sim 9\%$[2]. This makes manual intervention to detect these edits with potential inconsistent content, infeasible. Wikipedia hence deploys machine learning based classifiers (West et al., 2010; Halfaker and Taraborelli, 2015) which primarily leverage hand-crafted features from three aspects of revision (i) basic text features like repeated characters, long words, capitalized words etc. (ii) temporal features like inter arrival time between events of interest (iii) dictionary based features like presence of any curse words or informal words (e.g., 'hello', 'yolo'). Other feature based approaches include (Daxenberger and Gurevych, 2013; Bronner and Monz, 2012) which generally follow a similar archetype.

Proposed model: In most of the cases, the edits are reverted because they fail to abide by the edit guidelines, like usage of inflammatory wording, expressing opinion instead of fact among others (see Fig 2). These flaws are fundamentally related to the textual content rather than temporal patterns or editor behavior that have been deployed in existing methods. Although dictionary based approaches do look into text to a small extent (swear words, long words etc.), they account for only a small subset of the edit patterns. We further hypothesize that owing to the volume and

*Both authors contributed equally

[1] en.wikipedia.org/Wikipedia:List of policies
[2] stats.wikimedia.org/EN/PlotsPngEditHistoryTop.htm

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3962–3972
Florence, Italy, July 28 - August 2, 2019.

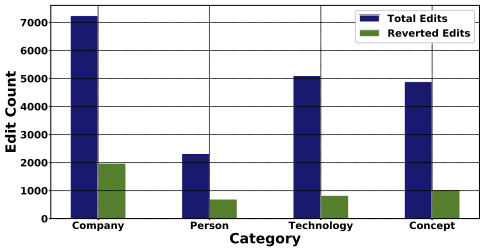

Figure 1: Average number of edits and average number of damaging edits, i.e., reverted edits for four different categories of pages. A fraction (at most 30%) of user generated edits are damaging edits.

Figure 2: Examples of edits in *Facebook* and *Google* Wikipedia page. The blue bubbles are the original sentences. The orange bubbles indicate damaging edits while the green bubbles indicate 'good faith' edits. Good faith edits are unbiased formal English sentence while damaging edits often correspond to incoherent use of language, abusive language, imperative mood, opinionated sentences etc.

variety of Wikipedia data, it is impossible to develop a feature driven approach which can encompass the wide array of dependencies present in text. In fact, we show that such approaches are inefficacious in identifying most of the damaging edits owing to these obvious limitations. We hence propose **S**elf A**t**ttentive **R**evision **E**ncoder (StRE) which extracts rich feature representations of an edit that can be further utilized to predict whether the edit has damaging intent. In specific, we use two stacked recurrent neural networks to encode the semantic information from sequence of characters and sequence of words which serve a twofold advantage. While character embeddings extract information from out of vocabulary tokens, i.e., repeated characters, misspelled words, malicious capitalized characters, unnecessary punctuation etc., word embeddings extract meaningful features from curse words, informal words, imperative tone, facts without references etc. We further employ attention mechanisms (Bahdanau et al., 2014) to quantify the importance of a particular character/word. Finally we leverage this learned representation to classify an edit to be damaging or valid. Note that *StRE* is reminiscent of structured self attentive model proposed in (Lin et al., 2017) albeit used in a different setting.

Findings: To determine the effectiveness of our model, we develop an enormous dataset consisting of $\sim 21M$ edits across $32K$ wikipages. We observe that *StRE* outperforms the closest baseline by at least 17% and at most 103% in terms of AUPRC. Since it is impossible to develop an universal model which performs equally well for all categories, we develop a transfer learning (Howard and Ruder, 2018) set up which allows us to deploy our model to newer categories without training from scratch. This further allows us to employ our model to pages with lower number of edits.

Contributions: Our primary contributions in this paper are summarized below -

(i) We propose a deep neural network based model to predict edit quality in Wikipedia which utilizes language modeling techniques, to encode semantic information in natural language.

(ii) We develop a novel dataset consisting of $\sim 21M$ unique edits extracted from $\sim 32K$ Wikipedia pages. In fact our proposed model outperforms all the existing methods in detecting damaging edits on this dataset.

(iii) We further develop a transfer learning set up which allows us to deploy our model to newer categories without the need for training from scratch.

Code and sample data related to the paper are available at `https://github.com/bhanu77prakash/StRE`.

2 The Model

In this section we give a detailed description of our model. We consider an edit to be a pair of sentences with one representing the original (P_{or}) while the other representing the edited version (P_{ed}). The input to the model is the concatenation of P_{or} and P_{ed} (say $P = \{P_{or}||P_{ed}\}$) separated by a delimiter ('||'). We assume P consists of w_i words and c_i characters. Essentially we consider two levels of encoding - (i) **character level** to extract patterns like repeated characters, misspelled words, unnecessary punctuation etc. and (ii) **word level** to identify curse words, imperative tone, opinionated phrases etc. In the following we present how

we generate a representation of the edit and utilize it to detect malicious edits. The overall architecture of *StRE* is presented in Fig 3.

2.1 Word encoder

Given an edit P with $w_i, i \in [0,L]$ words, we first embed the words through a pre-trained embedding matrix W_e such that $x_i = W_e w_i$. This sequence of embedded words is then provided as an input to a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) which provides representations of the words by summarizing information from both directions.

$$x_i = W_e w_i, i \in [0,L] \quad (1)$$

$$\overrightarrow{v}_i = \overrightarrow{LSTM}(x_i), i \in [0,L] \quad (2)$$

$$\overleftarrow{v}_i = \overleftarrow{LSTM}(x_i), i \in [L,0] \quad (3)$$

We obtain the representation for each word by concatenating the forward and the backward hidden states $v_i = [\overrightarrow{v}_i, \overleftarrow{v}_i]$. Since not all words contribute equally to the context, we deploy an attention mechanism to quantify the importance of each word. The final representation is then a weighted aggregation of the words.

$$u_i = \sigma(W_w v_i + b_w) \quad (4)$$

$$\beta_i = \frac{exp(u_i^T u_w)}{\sum_{i=0}^{T} exp(u_i^T u_w)} \quad (5)$$

$$R_w = \sum_i \beta_i v_i \quad (6)$$

To calculate attention weights (α_i) for a hidden state h_i, it is first fed through a single layer perceptron and then a *softmax* function is used to calculate the weights. Note that we use a word context vector u_w which is randomly initialized and is learnt during the training process. The use of context vector as a higher level representation of a fixed query has been argued in (Sukhbaatar et al., 2015; Kumar et al., 2016). Note that the attention score calculation is reminiscent of the one proposed in (Yang et al., 2016).

2.2 Character encoder

The character encoder module is similar to the word encoder module with minor differences. Formally we consider P ($\{P_{or}||P_{ed}\}$) as a sequence of T characters c_i, $i \in [0,T]$. Instead of using pre-trained embeddings as in case of word encoder, we define an embedding module, parameters of which is also learned during training which is basically an MLP. Each embedded character is then passed through a bidirectional LSTM to obtain the hidden states for each character. Formally, we have

$$y_i = \sigma(W_c c_i + b_c), i \in [0,T] \quad (7)$$

$$\overrightarrow{h}_i = \overrightarrow{LSTM}(y_i), i \in [0,T] \quad (8)$$

$$\overleftarrow{h}_i = \overleftarrow{LSTM}(y_i), i \in [T,0] \quad (9)$$

We next calculate the attention scores for each hidden state h_i as

$$z_i = \sigma(W_c h_i + b_c) \quad (10)$$

$$\alpha_i = \frac{exp(z_i^T u_c)}{\sum_{i=0}^{T} exp(z_i^T u_c)} \quad (11)$$

$$R_c = \sum_i \alpha_i h_i \quad (12)$$

Note that u_c is a character context vector which is learned during training.

2.3 Edit classification

The edit vector E_p (for an edit P) is the concatenation of character and word level encodings $E_p = [R_c, R_w]$ which we then use to classify whether an edit is valid or damaging. Typically, we perform

$$p = softmax(W_p E_p + b_p)$$

Finally we use binary cross entropy between predicted and the true labels as our training loss.

2.4 Transfer learning setup

Note that it is not feasible to train the model from scratch every time a new page in an existing or a new category is introduced. Hence we propose a transfer learning setup whereby, for a new page, we use the pre-trained model and only update the weights of the dense layers during training. The advantages are twofold - (i) the model needs only a limited amount of training data and hence can easily be trained on the new pages and (ii) we benefit significantly on training time.

3 Dataset

Wikipedia provides access to all Wikimedia project pages in the form of xml dumps, which is periodically updated[3]. We collect data from dumps made available by English Wikipedia project on June 2017 which contains information about 5.5M pages.

[3] https://dumps.wikimedia.org/enwiki/20181120

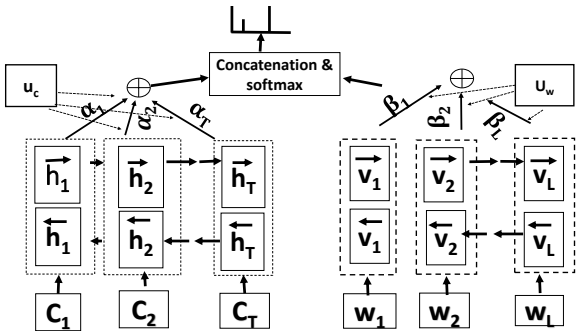

Figure 3: Overall architecture of StRE. The character encoding and the word encoding components of the model are shown in the left and right respectively. This is followed by the attention layer followed by concatenation and *softmax*.

We extract a subset of pages related to the Computer Science category in Wikipedia. Utilizing the category hierarchy[4] (Auer et al., 2007) (typically a directed graph containing parent and child categories), we extract all articles under the Computer Science category up to a depth of four levels which accounts for $48.5K$ Wikipedia pages across $1.5K$ categories[5]. We filter out pages with at least 100 edits which leaves us with $32K$ pages. For each page in our dataset we performed pairwise difference operation[6] between its current and previous versions to obtain a set of pairs with each consisting of a sentence and its subsequent modified version.

Edit quality: In order to train our model to identify quality edits from damaging edits we need a deterministic score for an edit. Our quality score is based on the intuition that if changes introduced by the edit are preserved, it signals that the edit was beneficial, whereas if the changes are reverted, the edit likely had a negative effect. This idea is adapted from previous work of Adler et al. (2011).

Consider a particular article and denote it by v_k its k-th revision (i.e., the state of the article after the k-th edit). Let $d(u,v)$ be the Levenshtein distance between two revisions. We define the *quality* of edit k from the perspective of the article's state after $\ell \geq 1$ subsequent edits as

$$q_{k|\ell} = \frac{d(v_{k-1}, v_{k+\ell}) - d(v_k, v_{k+\ell})}{d(v_{k-1}, v_k)}.$$

[4]Dbpedia.org

[5]Typical categories include 'Computational Science', 'Artificial Intelligence' etc.

[6]https://docs.python.org/2/library/difflib.html

Resources	Count
Pages	32394
Total edits	21,848960
Positive edits	15,791575
Negative edits	6,057385

Table 1: Summary of the dataset.

Intuitively, the quantity $q_{k|\ell}$ captures the proportion of work done on edit k that remains in revision $k+\ell$ and it varies between $q_{k|\ell} \in [-1,1]$, when the value falls outside this range, it is capped within these two values. We compute the mean quality of the edit by averaging over multiple future revisions as follows

$$q_k = \frac{1}{L}\sum_{\ell=1}^{L} q_{k|\ell}$$

where L is the minimum among the number of subsequent revisions of the article. We have taken $L = 10$, which is consistent with the previous work of Yardım et al. (2018).

Edit label: For each pair of edits we compute the edit quality score. If quality score is ≥ 0 we label an edit to be -1, i.e., done in good faith. However all edits with quality score < 0 are labeled 1, i.e., damaging edits. We further check that bad quality edits are indeed damaging edits by calculating what fraction of low score edits are reverted and what fraction of high score edits are not reverted. This result is illustrated in Figure 4. Information whether an edit is reverted or not can be calculated by mining Wikipedia's revert graph following the same technique illustrated by (Kittur et al., 2007). The results clearly show that a large proportion of bad quality edits are indeed reverted by the editors and similarly a large fraction of good quality edits are not reverted. Though bad quality edits are often reverted, all reverted edits are not bad. Malicious agents often engage in interleaving reverts, i.e., edit wars (Kiesel et al., 2017) as well as pesudo reverts. Hence we use quality metric to label damaging edits which is well accepted in the literature (Adler et al., 2008). We provide a summary of the data in Table 1. Our final data can be represented by a triplet $< s_i, s_f, l >$ where s_i is the initial sentence, s_f is the modified sentence and l indicates the edit label.

4 Experiments

In this section we demonstrate the effectiveness of our model compared to other existing techniques.

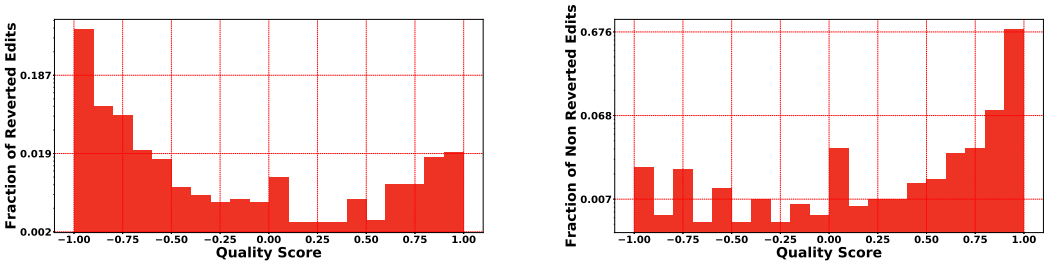

Figure 4: Distribution of the quality score for the revert edits (left) and non-reverted edits(right). The y-axis is in log scale. The plot shows that a large proportion of low quality edits are reverted and a large proportion of high quality edits are not reverted; hence this observation acts as a validation for our quality score metric.

Typically, we consider two sets of experiments - (i) category level and (ii) page level. In category level experiments (see section 4.3) we first form a random sample of data points belonging to pages in a fixed category. Our objective is to first train on edits related to a fixed page category and test on new edits belonging to pages of the same category. We further show through rigorous experiments that existing approaches of transfer learning and fine tuning (Howard and Ruder, 2018) can be applied to increase the efficacy of our approach. In page level experiments in section 4.4, we abandon the category constraint (as in case of category level experiments) and train (test) on edits irrespective of the category of the page which it belongs to and demonstrate that our model is equally effective.

4.1 Baseline approaches

We use two variants of our proposed model – word embedding with attention (*Word+Att*), character embedding with attention (*Char+Att*) as two baselines to compare to our model. We also compare existing feature based and event based approaches for edit quality prediction. We give a brief description of the other state-of-the-art baselines in the subsequent subsections.

4.1.1 ORES

The Objective Revision Evaluation Service (ORES) (Wikimedia, 2019) is a web service developed by Wikimedia foundation that provides a machine learning-based scoring system for edits. More specifically, given an edit, ORES infers whether an edit causes damage using linguistic features and edit based features (e.g., size of the revision etc.)

4.1.2 ORES++

In order to make it more competitive, we further augment ORES by adding linguistic quality indicators as additional features obtained from the *Empath* tool Fast et al. (2016). This tool scores edits on 16 lexical dimensions such as 'ugliness', 'irritability', 'violence' etc. We also use the count of POS tags following Manning et al. (2014) as well as the count of mispelled words as features using aspell dictionary Atkinson (2006).

4.1.3 Interrank

Interrank (Yardım et al., 2018) is a recent quality-prediction method which does not use any explicit content-based features but rather predicts quality of an edit by learning editor competence and page reputation from prior edit actions. The performance of Interrank has been revealed to be very close to ORES.

4.2 Model configuration

We use 300 dimensional pre-trained word Glove vector (Pennington et al., 2014) and 300 dimensional ASCII character embedding (Woolf, 2017). We also use 64 dimensional hidden layer in our model, followed by attention layer and three stacks of dense layer. Our context vector in the attention layer is of 64 dimensions and dense layers are 256, 64 and 16 dimensions. We further utilize dropout probability of 0.5 in the dense layers. We also employ binary cross entropy as loss function and Adam (Kingma and Ba, 2014) optimizer with learning rate 0.01 and weight decay of 0.0001 to train our model. The batch size is set to 250.

4.3 Category level experiments

In this set of experiments we essentially train and test on pages in the same category.

4.3.1 Page specific model

As a first step towards determining the potential of our model, we train our model on a set of edits of a particular page and predict on the rest. To this aim

we manually annotate top 100 pages in terms of total number of edits, into four categories, i.e., company, concept, technology, and person. Such granular level category annotation is not available from Wikipedia hierarchy which directs us towards annotation. In each category we tabulate the count of positive and negative datapoints in Table 2). For each page we randomly select 80% edits for training, 10% edits for validation and 10% edits as held out set. We train our model on 80% and tune it on the validation set. We finally test on the held out set. The same procedure is followed for *Word+Att*, *Char+Att*, *ORES++*. For all these models the AUPRC (mean,std) across all pages are presented in Table 3. Since *ORES* is already a pretrained model we test on the combined held out set of the pages. Note that *Interrank* is not designed for page level training and further requires large training data. Hence, for *Interrank* we train on the combined training set of the pages and test on the combined held out set. Results obtained on the held out set are reported in Table 3. Our experiments clearly show that *StRE* outperforms baselines by a significant margin (at least 10%). We also see that individual components of our model, i.e., *Char+Att* and *Word+Att* do not perform as well as *StRE* which further validates our architecture. Moreover, *Interrank* performs poorly despite combined dataset which shows that language modelling is essential in edit quality prediction.

Edits	**Company**	**Concept**	**Technology**	**Person**
+ve examples	813400	227308	294125	79035
-ve Examples	649078	124323	169091	28505
Total examples	1462478	351631	463216	107540

Table 2: Total number of data points along with positive and negative samples for the top five pages in terms of edit count in each category.

4.3.2 New page: same category

We now explore a more challenging setup for our model whereby instead of training and testing on edits of a specific annotated category, we train on edits of pages of a particular category but test on a previously unseen (during training) page of the same category. Specifically, for a given category, we train our model on 90% of pages and test our models on unseen page edits in the same category from our annotated dataset. The obtained results are tabulated in Table 4(a). Results show that such an approach is indeed fruitful and can be applied on pages which has very few edits utilizing intra-category pages with large edit counts.

Transfer learning results: Our results can be further improved by applying ideas of transfer learning. For each new page, we can initialize our model by pre-trained weights learned from training on other intra-category pages. We can then train the dense layer with only 20% of new datapoints randomly selected from the new page and test on the remaining 80%. This approach is adapted from the state-of-the-art transfer learning approaches (Howard and Ruder, 2018; Dehghani et al., 2017) where it has been shown to work on diverse NLP tasks. Such an approach achieves at least 3% and at most 27% improvement over prior results.

4.3.3 New page: different category

We now probe into how our model performs when tested on a page belonging to a previously unseen category. As a proof of concept, we train on all pages belonging to three categories (inter-category training) and test on a new page from the fourth category. We perform this experiment considering each of the four categories as unknown one by one in turn. The obtained results are presented in Table 4(b). Clearly, the results are inferior compared to intra-category training which corroborates with our argument that different category of pages have unique patterns of edits.

Transfer learning results: However, we alleviate the above problem by utilizing transfer learning approaches. In specific, we initialize our model with weights pre-trained on inter-category pages and train only the final dense layer on 20% of the new edits from the fourth category. Results point that we can obtain significant improvements, i.e., at least 10% and at most 28%. This is very a promising direction to pursue further investigations, since it is very likely that abundant edits may be present in distant categories while very limited edits may manifest in a niche category that has low visibility.

4.3.4 Multi category training

Finally, we proceed toward a category agnostic training paradigm. Essentially, we hold out 10% pages of the annotated set for each category. We train on all remaining pages irrespective of the category information and test on the held out pages from each category. We report the results in Table 4(c). Since our model learns from edits in all category of pages, we are able to obtain better re-

	Company	Concept	Technology	Person
Models	**AUPRC**	**AUPRC**	**AUPRC**	**AUPRC**
ORES	0.72	0.76	0.71	0.63
ORES++	0.84 ± 0.03	0.85 ± 0.03	0.87 ± 0.02	0.85 ± 0.03
Interrank	0.35	0.47	0.42	0.38
Word+Att	0.63 ± 0.02	0.74 ± 0.03	0.72 ± 0.01	0.78 ± 0.02
Char+Att	0.91 ± 0.01	0.84 ± 0.02	0.83 ± 0.02	0.81 ± 0.02
StRE	0.95 ± 0.02	0.89 ± 0.01	0.91 ± 0.01	0.87 ± 0.02

Table 3: AUPRC scores, with the best results in bold and gray background on the annotated dataset.

Category	Testing without Retraining	Testing with 20% Retraining
Person	0.81	0.85
Concept	0.77	0.91
Company	0.76	0.88
Technology	0.68	0.88

(a) Intra category AUPRC.

Category	Testing without Retraining	Testing with 20% Retraining
Person	0.67	0.82
Concept	0.63	0.81
Company	0.71	0.82
Technology	0.72	0.89

(b) Inter category AUPRC.

Category	Testing without Retraining	Testing with 20% Retraining
Person	0.71	0.83
Concept	0.85	0.90
Company	0.74	0.86
Technology	0.77	0.84

(c) Category agnostic AUPRC.

Table 4: Results for intra-category, inter-category and category agnostic predictions without and with transfer learning. The transfer learning approach is always beneficial.

sults from inter category setup. We further employ transfer learning (as in previous sections) on the new page which improves the results significantly (at least 6% and at most 16%).

To summarize the results in this section, we observe that testing on a previously unseen category leads to under-performance. However, retraining the dense layers with a few training examples drastically improves the performance of our model.

4.4 Page level experiments

We now consider an experimental setup agnostic of any category. In this setting, to train our model we form a set of edits which comprises 20% of our total edits in the dataset. This edits are taken from the pages which have largest edit count. Quantitatively, we impose a cap on the total number of edits to be 20% of the entire edit count. Subsequently, we start pooling training data from the largest page, followed by the second largest page and so on until our budget is fulfilled. The whole data so accumulated is divided into 80% training, 10% validation and 10% test sets. Results on this 10% held out data are reported in Table 5 as training AUPRC. We compare our model against other text based and event based quality predictor baselines. Since *ORES* is an already pre-trained web based service, we obtained AUPRC on the 10% held out set. In case of *Interrrank*, 90% of the data is used for training and 10% is used as held out set (as reported in the paper (Yardım et al., 2018)). Results show that our model performs significantly better than the baselines (by 24% in case of *ORES* and by 131% in case of *Interrank*).

Transfer learning results: For each of the remaining pages in our data we first utilize our pretrained model from the last step. However, we train the dense layers with randomly selected 20% datapoints from the page to be tested. The remaining data is used for testing. We follow this procedure for all remaining pages and calculate the mean test AUPRC along with standard deviation which we report in Table 5. In case of *ORES* we evaluate on the 80% data. In case of *Interrrank*, we merge all remaining data into a single dataset and use 90% of the data for training and 10% for test. We show that transfer learning approach can be useful in this setting and we obtain 17% improvement compared to *ORES* and 103% improvement compared to *Interrank*.

Model	Training AUPRC	Testing AUPRC
ORES	0.77	0.75
Interrank	0.41	0.42
Word+Att	0.64	0.77 ± 0.1
Char+Att	0.92	0.83 ± 0.09
StRE	0.95	0.88 ± 0.09

Table 5: Comaprison between *StRE* and baselines on complete dataset.

5 Discussion

Model retraining: We demonstrate in our experiments that a fraction of edits from unseen pages results in the improvement over pretrained models. We further investigate the model performance if we increase the volume of the retraining data (results shown for the intra-category setup, all other setups show exactly similar trend). We vary the unseen data used for fine tuning the model from

Original version	Revised version
Google Maps offers detailed *streetmaps* and *route planning* information.	Google Maps offers detailed *streetmaps* and *route planning* information in United States and Canada.
Proponents argued that *privacy complaints* are baseless.	Proponents of *trusted computing* argue that *privacy complaints* have been addressed in the existing *specifications* - possibly as a result of criticism of early versions of the *specifications*.

Table 6: Anecdotal examples of edits in *Google Maps* and *Internet Privacy* wikipage. Here the general model fails to identify negative examples while retraining the dense layer learns better representations and identifies the negative examples correctly. Page specific tokens are colored in blue.

5% to 50% and show that growth in AUPRC stabilizes (see Fig 5) which validates our proposal to utilize a smaller fraction.

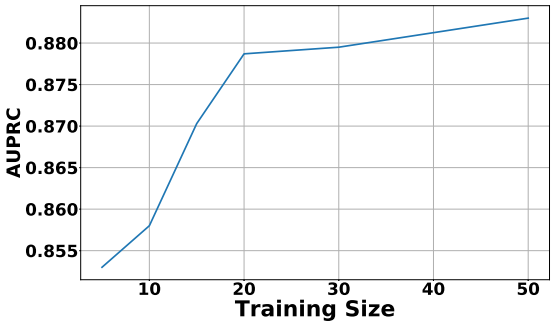

Figure 5: AUPRC using transfer learning in *intra-category setup* with gradual increase in retraining percentages. Similar trends are obtained with the other setups.

Anecdotal examples: In order to obtain a deeper understanding of the results, we explore few examples where the general model fails while retraining the dense layers leads to correct classification. In Table 6 we present two such examples. Note that our general model (without retraining the dense layers) wrongly classifies them as damaging edits while retraining leads to correct classification. We believe that retraining the dense layers leads to obtaining superior representation of edits, whereby, page specific words like 'streetmaps', 'route planning' in *Google Maps* or 'privacy complaints', 'trusted computing' in *Internet Privacy* are more pronounced.

Timing benefits: Another potential benefit is the amount of time saved per epoch as we are only back propagating through the dense layers. To quantify the benefit in terms of time, we select a random sample of pages and train one version of our model end-to-end across all layers and another version only up to the dense layer. For our model, the average time taken per epoch achieves $\sim$ 5x improvement over the traditional approach. The performance in the two cases are almost same. In fact, for some cases the traditional end-to-end training leads to inferior results as LSTM layers fail to learn the best weights with so few examples.

6 Related work

Edit quality prediction in Wikipedia has mostly been pursued in the lines of vandalism detection. Kumar et al. (2015) developed a system which utilized novel patterns embedded in user editing history, to predict potential vandalism. Similar feature based approach has also been applied in both standard (Green and Spezzano, 2017) and sister projects of Wikipedia such as wikidata (Heindorf et al., 2016; Sarabadani et al., 2017). Yuan et al. (2017) propose to use a modified version of LSTM to solve this problem, hence avoiding feature engineering. A complementary direction of investigation has been undertaken by (Daxenberger and Gurevych, 2013; Bronner and Monz, 2012) who bring forth a feature driven approach, to distinguish spam edit from a quality edit. A feature learning based approach has been proposed by Agrawal and Dealfaro (2016); Yardım et al. (2018) which observes all the past edits of a user to predict the quality of the future edits. Temporal traces generated by edit activity has also been shown (Tabibian et al., 2017) to be a key indicator toward estimating reliability of edits and page reputation. One of the major problems in these approaches is that they require user level history information which is difficult to obtain because the same user may edit different Wikipedia pages of diverse categories and it will be time consuming to comb through millions of pages for each user.

There has also been no work to understand the possibility of predicting edit quality based on edits in pages in a common category. However, there has been no work to leverage advanced machinery developed in language modeling toward predicting edit quality.

Transfer learning: Several works (Long et al., 2015b; Sharif Razavian et al., 2014) in computer vision (CV) focus on transfer learning approach as deep learning architectures in CV tend to learn generic to specific tasks from first to last layer. More recently (Long et al., 2015a; Donahue et al., 2014) have shown that fine tuning the last or several of the last layers and keeping the rest of the layers frozen can have similar benefits. In natural language processing (NLP) literature, (Severyn and Moschitti, 2015) showed that unsupervised language model based embedding can be tuned using a distant large corpus and then further applied on a specialized task such as sentiment classification. This approach of *weak supervision* followed by *full supervision* to learn a *confident model* (Dehghani et al., 2017; Howard and Ruder, 2018; Jan et al., 2016) has been shown to reduce training times in several NLP tasks. In this paper we apply a similar framework for the first time in predicting the edit quality in Wikipedia pages in one category by initializing parameters from a trained model of a different category. This is very effective in cases where the former category page has limited number of data points.

7 Conclusion

In this paper we proposed a novel deep learning based model *StRE* for quality prediction of edits in Wikipedia. Our model combines word level as well as character level signals in the orthography of Wikipedia edits for extracting a rich representation of an edit. We validate our model on a novel data set comprising millions of edits and show efficacy of our approach compared to approaches that utilize handcrafted features and event based modelling. One of the remarkable findings of this study is that only 20% of training data is able to boost the performance of the model by a significant margin.

To the best of our knowledge, this is the first work which attempts to predict edit quality of a page by learning signals from similar category pages as well as cross category pages. We further show applications of recent advances in transfer learning in this problem and obtain significant improvements in accuracy without compromising training times. We believe this work will usher considerable interest in understanding linguistic patterns in Wikipedia edit history and application of deep models in this domain.

References

B Thomas Adler, Luca De Alfaro, Santiago M Mola-Velasco, Paolo Rosso, and Andrew G West. 2011. Wikipedia vandalism detection: Combining natural language, metadata, and reputation features. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 277–288. Springer.

B Thomas Adler, Luca De Alfaro, Ian Pye, and Vishwanath Raman. 2008. Measuring author contributions to the wikipedia. In *Proceedings of the 4th International Symposium on Wikis*, page 15. ACM.

Rakshit Agrawal and Luca Dealfaro. 2016. Predicting the quality of user contributions via lstms. In *Proceedings of the 12th International Symposium on Open Collaboration*, page 19. ACM.

Kevin Atkinson. 2006. Gnu aspell 0.60. 4.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Amit Bronner and Christof Monz. 2012. User edits classification using document revision histories. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 356–366. Association for Computational Linguistics.

Johannes Daxenberger and Iryna Gurevych. 2013. Automatically classifying edit categories in wikipedia revisions. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 578–589.

Mostafa Dehghani, Aliaksei Severyn, Sascha Rothe, and Jaap Kamps. 2017. Learning to learn from weak supervision by full supervision. *arXiv preprint arXiv:1711.11383*.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655.

Ethan Fast, Binbin Chen, and Michael S Bernstein. 2016. Empath: Understanding topic signals in large-scale text. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 4647–4657. ACM.

Thomas Green and Francesca Spezzano. 2017. Spam users identification in wikipedia via editing behavior. In *Eleventh International AAAI Conference on Web and Social Media*.

Aaron Halfaker and Dario Taraborelli. 2015. Artificial intelligence service ores gives wikipedians x-ray specs to see through bad edits.

Stefan Heindorf, Martin Potthast, Benno Stein, and Gregor Engels. 2016. Vandalism detection in wikidata. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 327–336. ACM.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

Christoph Hube and Besnik Fetahu. 2019. Neural based statement classification for biased language. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 195–203. ACM.

Deriu Jan et al. 2016. Sentiment classification using an ensemble of convolutional neural networks with distant supervision. *Proceedings of SemEval (2016)*, pages 1124–1128.

Johannes Kiesel, Martin Potthast, Matthias Hagen, and Benno Stein. 2017. Spatio-temporal analysis of reverted wikipedia edits. In *Eleventh International AAAI Conference on Web and Social Media*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Aniket Kittur, Bongwon Suh, Bryan A Pendleton, and Ed H Chi. 2007. He says, she says: conflict and coordination in wikipedia. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 453–462. ACM.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Srijan Kumar, Francesca Spezzano, and VS Subrahmanian. 2015. Vews: A wikipedia vandal early warning system. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 607–616. ACM.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015a. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. 2015b. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Amir Sarabadani, Aaron Halfaker, and Dario Taraborelli. 2017. Building automated vandalism detection tools for wikidata. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 1647–1654. International World Wide Web Conferences Steering Committee.

Aliaksei Severyn and Alessandro Moschitti. 2015. Unitn: Training deep convolutional neural network for twitter sentiment classification. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 464–469.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Behzad Tabibian, Isabel Valera, Mehrdad Farajtabar, Le Song, Bernhard Schölkopf, and Manuel Gomez-Rodriguez. 2017. Distilling information reliability and source trustworthiness from digital traces. In *Proceedings of the 26th International Conference on World Wide Web*, pages 847–855. International World Wide Web Conferences Steering Committee.

Andrew G West, Sampath Kannan, and Insup Lee. 2010. Stiki: an anti-vandalism tool for wikipedia using spatio-temporal analysis of revision metadata. In *Proceedings of the 6th International Symposium on Wikis and Open Collaboration*, page 32. ACM.

Wikimedia. 2019. Objective revision evaluation service ORES.

Max Woolf. 2017. Pretrained word embeddings file into a character embeddings.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Ali Batuhan Yardım, Victor Kristof, Lucas Maystre, and Matthias Grossglauser. 2018. Can who-edits-what predict edit survival? *arXiv preprint arXiv:1801.04159*.

Shuhan Yuan, Panpan Zheng, Xintao Wu, and Yang Xiang. 2017. Wikipedia vandal early detection: from user behavior to user embedding. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 832–846. Springer.