Modeling the Relationship between User Comments and Edits in Document Revision

Xuchao Zhang[†*], **Dheeraj Rajagopal**[‡*], **Michael Gamon**[§], **Sujay Kumar Jauhar**[§] and **Chang-Tien Lu**[†]
[†]Discovery Analytics Center, Virginia Tech, Falls Church, VA, USA
[‡]Carnegie Mellon University, Pittsburgh, PA, USA
[§]Microsoft Research, Redmond, WA, USA
[†]{xuczhang, ctlu}@vt.edu, [‡]dheeraj@cs.cmu.edu, [§]{mgamon, sjauhar}@microsoft.com

Abstract

Management of collaborative documents can be difficult, given the profusion of edits and comments that multiple authors make during a document's evolution. Reliably modeling the relationship between edits and comments is a crucial step towards helping the user keep track of a document in flux. A number of authoring tasks, such as categorizing and summarizing edits, detecting completed to-dos, and visually rearranging comments could benefit from such a contribution. Thus, in this paper we explore the relationship between comments and edits by defining two novel, related tasks: Comment Ranking and Edit Anchoring. We begin by collecting a dataset with more than half a million comment-edit pairs based on Wikipedia revision histories. We then propose a hierarchical multi-layer deep neural-network to model the relationship between edits and comments. Our architecture tackles both Comment Ranking and Edit Anchoring tasks by encoding specific edit actions such as additions and deletions, while also accounting for document context. In a number of evaluation settings, our experimental results show that our approach outperforms several strong baselines significantly. We are able to achieve a precision@1 of 71.0% and a precision@3 of 94.4% for Comment Ranking, while we achieve 74.4% accuracy on Edit Anchoring.

1 Introduction

Comments are widely used in collaborative document writing as a natural way to suggest and track changes, annotate content and explain the intent of edits. Table 1 shows an example of a comment left by an editor in a Wikipedia revision. The comment explains the intent of adding a missing researcher's name (Sutskever) to a citation.

This example demonstrates that user comments closely relate to the intent of an edit, the actual edit operation, and the content and location in the document that underwent change. However, during collaborative document authoring, the tracking and maintenance of comments becomes increasingly more challenging due to the large number of edits and comments that authors make. For example, structurally refactoring a document can significantly change the order of paragraphs and sentences, stranding comments in confusing and contextually inappropriate locations. Or, comments may have already been addressed but continue to linger in the document without having been marked as completed. These issues, among others, are exacerbated when multiple authors collaborate on a document, often asynchronously. It becomes difficult to know which tasks have been completed and which haven't, especially if authors are not proactive about marking comments as addressed. This situation stands to benefit from an intelligent system capable of marking changes as completed, by understanding the relationship between edits and comments.

Many tasks in document management, such as summarizing and categorizing edits, detecting to-do item completion, prioritizing writing tasks and visually re-arranging document structure to reflect state in multi-author scenarios, require users to understand edit-comment interactions. Consider the scenario where multiple authors make edits to a document using the track-change feature available in popular document authoring apps. The result can be extremely confusing, and it is difficult to disentangle who edited what, over multiple versions. A feature that could summarize these edits, so that the visual burden of tracking changes is not on the UI, would certainly alleviate this problem. Such a system would necessarily first need to learn mappings between edits and natural lan-

[*] The work was done while these authors interned in Microsoft Research.

guage comments, before it could learn to generate them. Therefore, automatic solutions to these challenges stand to benefit from an ability to fundamentally model the relationship between edits and comments.

Yet, most existing studies on document revisions focus on modeling the document edits (Bronner and Monz, 2012) or comments (Shaver, 2016) separately; or using comments as a supplementary source of information to study the edits (Yatskar et al., 2010). To the best of our knowledge, no prior work focuses on jointly modeling the relationship between comments and edits.

Thus, in this paper we tackle two novel tasks. **Comment Ranking** considers a document edit operation and seeks to rank a set of candidate comments in order of their relevance to the edit. **Edit Anchoring** seeks to identify the locations in a document that are most likely to have undergone change as the result of a specific comment. Crucially, both tasks require jointly modeling comment-edit relationship.

We start by collecting a dataset of 780K Wikipedia revisions, each with their associated comment and edits. We then build a hierarchical multi-layer deep neural network, a model we call CmntEdit, which is capable of learning the relationship between comments and edits from this data. Our approach addresses both Comment Ranking and Edit Anchoring by sharing many of the model's components and parameters across tasks. Since edits can apply to discontiguous sequences of text, which pose a challenge for sequence modeling approaches, we explore novel ways to represent a document both before and after an edit, while also accounting for contextual information surrounding an edit. To differentiate the context from edit words, we also explore a novel mechanism to encode edit operations such as additions and deletions explicitly in the model. [1]

Finally, we evaluate our model on both tasks and in a number of experimental settings, demonstrating that our solution is significantly better than several strong baselines on jointly capturing the relationship between edits and comments. Our model outperforms the best baseline by 34.6% on NDCG for Comment Ranking and achieves a best score of 0.687 F1 for Edit Anchoring. Additionally, in an ablation study we demonstrate that our various modeling choices, which tackle the inherent challenges of comment-edit understanding, each contribute positively to empirical results.

[1]We are making the code for the CmntEdit model, and generation of our dataset publicly available to the research community at https://github.com/microsoft/WikiCommentEdit.

Comment	# Sutskever missing
Pre-edit Version	In October 2012, a similar system by Krizhevsky and Hinton won the large-scale ImageNet competition by a significant margin over shallow...
Post-edit Version	In October 2012, a similar system by Krizhevsky and **Sutskever and** Hinton won the large-scale ImageNet competition by a significant margin over shallow...

Table 1: Example of an edit and its associated comment. The added words "Sutskever and" in post-edit version is marked in red.

2 Related Work

Document revisions have been the subject of several studies in recent years (Nunes et al., 2011; Fischer, 2013). Most prior work focuses on modeling document edits only. For instance, Bronner and Monz (2012) build a classifier to distinguish fluency edits from factual edits. Zhu et al. (2017) study the semantic distance between the content in different versions of documents to detect document revisions. Grossman et al. (2013) propose a hierarchical navigation method to display document revision histories.

Some work utilizes comments associated with document edits as supplementary information to study the document revisions. For example, Yatskar et al. (2010) consider both comment and document revision for lexical simplification. However, they use comments as meta-data to identify trusted revisions, rather than directly modeling the relationship between comments and edits. Yang et al. (2017) featurize both comments and revisions to classify edit intent, but without explicitly modeling edit-comment relationship.

Wikipedia revision history data (Nunes et al., 2008) has been used in many NLP tasks (Zesch, 2012; Max and Wisniewski, 2010; Ganter and Strube, 2009). For instance, Yamangil and Nelken (2008) model Wikipedia revision histories for improving sentence compression, Aji et al. (2010) propose a new term weighting model leveraging Wikipedia revision histories, and Zanzotto and Pennacchiotti (2010) expand textual entailment corpora from Wikipedia revision histories using co-training. Again, however, none of these methods directly consider or model the relationship between comments and edits.

At a basic level, modeling the connection between comments and edits can be seen as a text matching problem, with superficial similarity to other common NLP tasks, such as Question Answering (Seo et al., 2016; Yu et al., 2018), document search (Burges et al., 2005; Nalisnick et al., 2016), and textual entailment (Androutsopoulos and Malakasiotis, 2010), among others. Note however, that edits are a (possibly discontinuous and non-trivial) delta between two versions of a text, making their representation and understanding more challenging than that of a simple string. We demonstrate this in our evaluation in Section 5.2, where we compare against several competing models that were designed for other text matching challenges.

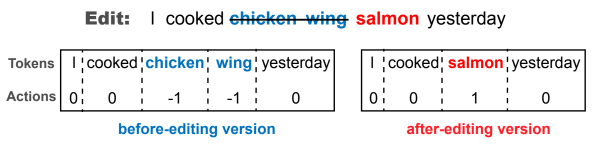
3 Dataset

Our dataset – which we call *WikiCmnt* – is generated from Wikipedia revision histories. In the absence of publicly available document data, Wikipedia is a particularly rich resource: (i) It maintains full revision histories of every Wikipedia page, along with associated editor comments as meta-data. (ii) It is a large-scale instance of multi-author collaboration, with many editors contributing to and maintaining pages.

The specific historical dump we use is from May 1, 2018. It contains approximately 52.7 million pages, and 755.5 million unique revisions made by 300.8 million users. WikiCmnt is a subsample of 786,866 Wikipedia page revisions along with their associated metadata. Revisions are filtered out before sampling, if they violate any one of the following criteria: (i) The length of the comment is longer than 8 words. (ii) The edits made to the Wikipedia page span more than one section[2]. (iii) The Wikipedia page has an edit history containing fewer than 10 unique revisions.

We extract and store a number of data fields from the Wikipedia revision history as summarized in Table 2. For each specific revision of a page, we not only retrieve the text of the comment and edit but also sample 10 non-related comments (*Neg-Cmnts*) and 10 non-related edits (*Neg-Edits*) from the same page's history. Finally we also encode the individual edit operations in both pre-edit and post-edit versions of a text. For example, consider Figure 1, which shows the edit action encoding associated with an change from "chicken wing" to "salmon".

[2] https://en.wikipedia.org/wiki/Help:Section



Edit: I cooked ~~chicken wing~~ salmon yesterday
Tokens
Actions
I cooked chicken wing yesterday
0 0 -1 -1 0
before-editing version
I cooked salmon yesterday
0 0 1 0
after-editing version

Figure 1: An example of the action encoding associated with changing the phrase "chicken wing" to "salmon". Specifically, the values -1, 1 and 0 are used to represent deletions, additions and unchanged tokens, respectively.

Field	Description
Revision ID	Wiki revision ID
Parent ID	Parent revision ID
Timestamp	Timestamp of revision
Page Title	Title of Wikipedia page
Comment	Revision comment
Neg-Cmnts	Negative sampled comments
Src-Tokens	Tokens of pre-editing document
Src-Actions	Action encoding of pre-editing document
Tgt-Tokens	Tokens of post-editing document
Tgt-Actions	Action encoding of post-editing document
Pos-Edits	Edited sentences in post-editing document
Neg-Edits	Negative sampled sentences in post-editing document

Table 2: Data Fields in *WikiCmnt* Dataset

4 Proposed Model

To model the relationship between edits and comments, we first formulate the problem with respect to the two tasks of Comment Ranking and Edit Anchoring; we then provide details of the neural architecture used to solve these problems; finally we provide some implementation details. We begin with preliminary notation.

Let us define c as a comment consisting of word tokens $\{w_1, ..., w_q\}$. Minimally, let us also define a pre-edit e_s, as the contiguous sequence of words spanning the first to the last edited token (with possibly intervening tokens that are not edited) in a document's *pre-edit version*. The edit e_s may optionally also contain some surrounding context. Formally this can be defined as:

$$\{\underbrace{\mathrm{w_{i-k}}, \dots \mathrm{w_{i-1}}}_{\text{context before edit}}, \underbrace{\mathrm{w_i}, \dots \mathrm{w_{i+p}}}_{\text{edit words}}, \underbrace{\mathrm{w_{i+p+1}}, \dots \mathrm{w_{i+p+k}}}_{\text{context after edit}}\}$$

where i and $i+p$ are the indices of the first and the last edited tokens in a revision, and k is the context window size. If there is more than one contiguous block of edited tokens, these blocks are concatenated to form the set of edit words with their context words.

We define a document edit as the pair $e = \{e_s, e_t\}$, where e_t is similarly defined over edited tokens and their context words in a document's *post-edit version*.

4.1 Problem Formulation

Comment Ranking is the task of finding the most relevant comment among a list of potential candidates, given a document edit. The inputs of the comment ranking task are some set of user comments $C = \{c_1, c_2, \ldots c_m\}$ pertaining to a document, and an edit $e = \{e_s, e_t\}$ in the same document.The goal of the task is to produce a ranking on C, such that the true comment c_i with respect to the edit $e = \{e_s, e_t\}$ is ranked above all the other comments (i.e. distractors). We use standard ranking metrics to evaluate model performance: Precision@K (P@K), Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain (NDCG).

Edit Anchoring is the task of finding the sentences in a document that most likely underwent change, given a specific user comment. The inputs to the task are a user comment c and a list of candidate sentences $S = \{s_1, s_2, \ldots s_n\}$ in the post-edit document e_t. Unlike with Comment Ranking, we operate under the assumption that an edit has already been completed, and therefore discard the information from the pre-edit version e_s. In the ground truth, at least one (but possibly more) of the sentences is an edit location for the comment c. The expected output is a list of binary classifications $R = \{r_i\}_{i=1}^{n}$, where $r_i = 1$ indicates that the sentence s_i is a likely edit location, given comment c. We use Accuracy and F1 score to evaluate performance on this task.

4.2 Model Overview

For both tasks, our models are hierarchical deep neural networks with four layers: an input embedding layer, a contextual embedding layer, a comment-edit attention layer, and an output layer. The overall architecture is shown in Figure 2. We describe each of the four layers in what follows. Since both models share many components we will describe the more general case that covers all inputs for Comment Ranking; for Edit Anchoring those components that correspond to the pre-edit input are suitably omitted.

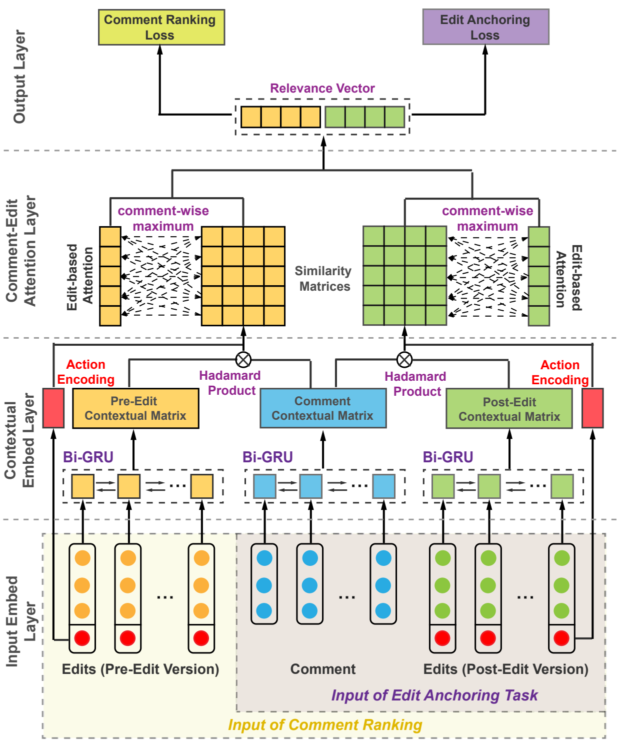


Figure 2: Overall Architecture of Proposed Model

Input Embedding Layer. The input embedding layer maps each word in user comments c and edits $e = \{e_s, e_t\}$ to a high-dimensional vector space. The output of the input embedding layer are matrices: $\boldsymbol{U} \in \mathbb{R}^{d \times M}$ representing the pre-edit document, $\boldsymbol{V} \in \mathbb{R}^{d \times M}$ representing the post-edit document, and $\boldsymbol{Q} \in \mathbb{R}^{d \times J}$ representing the comment. Here M is the length of edits and J is the length of the comment, while d is the fixed-length dimension of word vectors.

Contextual Embedding Layer. We use a bi-directional Gated Recurrent Unit (GRU) (Chung et al., 2014) to model the sequential interaction between words. Operating over the output of the previous layer we obtain the contextual embedding matrices $\boldsymbol{U}^c \in \mathbb{R}^{2d \times M}$ and $\boldsymbol{V}^c \in \mathbb{R}^{2d \times M}$ for both pre- and post-edit versions. Also, we obtain $\boldsymbol{Q}^c \in \mathbb{R}^{2d \times J}$ from the comment word vectors. Note that the row dimension of contextual matrices $\boldsymbol{U}^c$, $\boldsymbol{V}^c$ and $\boldsymbol{Q}^c$ are $2d$ because of the concatenation of the GRU's output in both forward and backward directions.

Comment-Edit Attention Layer. Inspired by the attention mechanisms utilized in machine comprehension (Seo et al., 2016; Yu et al., 2018), the comment-edit attention layer is designed to capture relationships between document edit and comment words. The attention layer maintains and processes both pre- and post-edit documents separately. This is to reduce the information loss that would have occurred if their representations were fused before the attention layer. Additionally, this layer incorporates an action encoding vector, which is designed to reflect the three kinds of edit operations: adding a word, deleting a word, or leaving it unchanged.

The inputs to the layer are the contextual matrices $\boldsymbol{U}^c$ and $\boldsymbol{V}^c$ of the pre- and post-edit documents respectively, the matrix $\boldsymbol{Q}^c$ representing the comment, and the supplemental action encoding vectors $\boldsymbol{a}^\dagger, \boldsymbol{a}^\ddagger \in \mathbb{Z}^M$ which encode the edit operation each token undergoes in the pre- and post-edit documents, respectively. The output is the comment-aware concatenated vector representations of the edit words in both pre- and post-edit documents, $\boldsymbol{h} \in \mathbb{R}^{2J}$.

Internally, we first calculate the shared similarity matrix $\boldsymbol{S}^\dagger \in \mathbb{R}^{M\times J}$ between the comment $\boldsymbol{Q}^c$ and contextual matrix $\boldsymbol{U}^c$ of pre-edit documents, while also accounting for the action encoding vector $\boldsymbol{a}^\dagger$. The elements of this shared similarity matrix are defined as follows:

$$\boldsymbol{S}^\dagger_{ij} = \mathcal{G}(\boldsymbol{U}^c_{:i}, \boldsymbol{Q}^c_{:j}, \boldsymbol{a}^\dagger_i) \qquad (1)$$

where $\mathcal{G}$ is a trainable function that generates the similarity between the word-level representations of comments and edits with respect to an edit operation.

Here $\boldsymbol{U}^c_{:i} \in \mathbb{R}^{2d\times 1}$ is the vector representation of the i-th word in the pre-edit document and $\boldsymbol{Q}^c_{:j} \in \mathbb{R}^{2d\times 1}$ is the vector representation of j-th word in the comment. $\boldsymbol{a}^\dagger_i \in \{-1, 0, 1\}$ is the action encoding for the edit operation performed on the i-th word in the pre-edit document. We choose the trainable function $\mathcal{G}(\boldsymbol{u}, \boldsymbol{q}, a) = \boldsymbol{w}^T[\boldsymbol{u} \otimes \boldsymbol{q}; a]$, where $\boldsymbol{w} \in \mathbb{R}^{(2d+1)\times 1}$ is a trainable weight vector, $\otimes$ is the element-wise multiplication operator and $[;]$ represents the vector concatenation across the row dimension. Similarly, we can calculate the shared similarity matrix $\boldsymbol{S}^\ddagger \in \mathbb{R}^{M\times J}$ between the comment and contextual matrix of the post-edit document as $\boldsymbol{S}^\ddagger_{ij} = \mathcal{G}(\boldsymbol{V}^c_{:i}, \boldsymbol{Q}^c_{:j}, \boldsymbol{a}^\ddagger_i)$. Note that the weight vectors in function $\mathcal{G}$ are different for pre- and post-edit versions, and both are trained simultaneously in the model.

After the similarity matrices are computed, we use them to generate the Comment-to-Edit Attention (C2E) weights. C2E is used to represent the relevance of words in the edit to those that appear in the comment. This is critical for modeling the relationship between comments and edits. We obtain the C2E attention weights $\boldsymbol{c}^\dagger \in \mathbb{R}^M$ for edit words in the pre-edit document by taking $\boldsymbol{c}^\dagger = \mathrm{softmax}(\max_{\mathrm{col}}(\boldsymbol{S}^\dagger))$, where the $\max_{\mathrm{col}}(\cdot)$ operator finds the column-wise maximum value from a matrix. Similarly, for the post-edit document, we have $\boldsymbol{c}^\ddagger = \mathrm{softmax}(\max_{\mathrm{col}}(\boldsymbol{S}^\ddagger))$.

Finally we multiply the attention vectors to the previously computed similarity matrices for both pre- and post-edit documents, and concatenate the results to obtain the relevance vector $\boldsymbol{h} \in \mathbb{R}^{2J}$:

$$\boldsymbol{h} = \left[\left(\boldsymbol{c}^\dagger\right)^T \boldsymbol{S}^\dagger \,;\, \left(\boldsymbol{c}^\ddagger\right)^T \boldsymbol{S}^\ddagger\right]^T \qquad (2)$$

The vector $\boldsymbol{h}$ intuitively captures the weighted sum of the relevance of the comment with respect to the edits in both pre- and post-edit documents.

Output Layer and Loss Function. The output layer and loss function of the network is task-specific. Comment Ranking requires ranking the relevance of candidate comments given a document edit. Broadly we obtain a ranked list by computing the relevance score between comments and edits by the output $r = \boldsymbol{\beta}^T\boldsymbol{h}$, where $\boldsymbol{\beta}$ is a trainable weight vector.

A data sample i in Comment Ranking consists of one true comment-edit pair and n_i negative comment-edit distractors. We denote r^+_i as the relevance score of the true pair, and r^-_{ij} as the relevance score of the j-th distractor pair (with $1 \leq j \leq n_i$). The goal of our task is to make $r^+_i > r^-_{ij}$ for all j. We therefore set the loss function to be a pairwise hinge loss between true and distractor relevance scores.

$$\mathcal{L}_c(\Theta) = \sum_{i=1}^{N}\sum_{j=1}^{n_i} \max(0, 1 - r^+_i + r^-_{ij}) \qquad (3)$$

where Θ is the set of all trainable weights in the model and N is the number of training samples in the dataset.

For Edit Anchoring, the goal is to predict whether a sentence in the document is likely to be the location of an edit, given a comment. This is a binary classification problem, and we can suitably set the output layer as $p = \mathrm{softmax}(\boldsymbol{\gamma}^T\boldsymbol{h})$ – the probability of predicting the positive class. Here γ is a trainable weight vector.

Given the binary nature of the classification problem we can use the cross entropy loss:

$$\mathcal{L}_e(\Phi) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{m_i}\Big[y_{ij}\log p_{ij} + (1-y_{ij})\log\big(1-p_{ij}\big)\Big], \tag{4}$$

where Φ is the set of all trainable weights in the model, N is the number of data samples, m_i is the number of sentences in the i-th data sample, p_{ij} is the predicted label of the j-th sentence in the i-th data sample, and y_{ij} is the corresponding ground truth label.

4.3 Implementation Details

The CmntEdit model described in this section is implemented using the Pytorch[3] framework and trained on a single Tesla P100 GPU with 16GB memory. The word vectors are initialized with pre-trained Glove embeddings (Pennington et al., 2014) using the default dimensionality of 100. We set the number of training epochs to 5, the maximum length of a comment to 30 tokens and the maximum length of an edit to 300 tokens. For the Comment Ranking task, we set the batch size to 10 and consider 5 candidate comments in each data sample: one true comment and 4 distractors. We thus have 5 comment-edit pairs for each data sample and 50 pair-wise samples for each training batch. For the Edit Anchoring task, we also set the batch size to 10 and consider 5 candidate sentences, which yields an identical 50 training instances per batch.

It should be noted that while we train our model with only 5 candidate comments or sentences (for Comment Ranking or Edit Anchoring respectively), the models – once trained – can be applied to any number of candidates at test time for either task.

5 Experiment

In this section, we show evaluation results of our model on the previously collected Wikipedia dataset 3. We begin by introducing the experimental settings in Section 5.1. We then compare the performance achieved by the proposed method against several baseline models in Section 5.2. We also conduct an ablation study to evaluate the various components of our model, as well as provide some qualitative results to demonstrate it's effectiveness in practise.

[3]https://pytorch.org/

5.1 Experimental Setup

We begin by introducing the evaluation setting, metrics and baselines we use in our experiments.

5.1.1 Datasets and Labels

We use the *WikiCmnt* dataset described in Section 3 for training and evaluation. The dataset contains 786,886 data samples in total. We reserve 70% of the data for training and split the remainder between 10% for validation and 20% for testing.

For the Comment Ranking task, we always have a single true comment, but in our test we experiment with both 4 and 9 distractors, yielding a total of 5 and 10 candidate comments. Similarly for the Edit Anchoring task, we also test with both 5 and 10 candidate sentences.

5.1.2 Evaluation Metrics

We use common ranking evaluation metrics from the literature to evaluate models on the task of Comment Ranking. These include: (i) **Precision@K**. The proportion of predicted instances where the true comment appears in the ranked top-K result, with $K = 1, 3, 5$. (ii) **Mean Reciprocal Rank (MRR)**. The average multiplicative inverse of the rank of the correct answer, represented mathematically as $\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\text{rank}_i}$, where N is the number of samples and rank_i is the rank assigned to the true comment by a model. (iii) **Normalized Discounted Cumulative Gain (NDCG)**. the normalized gain of each comment based on its ranking position in the results. We set the relevance score of the true comment to one and those of the distractors to zero.

For the Edit Anchoring task, we use **Accuracy** and **F1 Score** as evaluation metrics. These are computed based on a model's classification of candidate sentences in the post-edit version of the document.

5.1.3 Baseline Methods

For the Comment Ranking task, we compare our approach to the following baselines:
(i) **Cosine-TfIdf** uses the cosine similarity between the TfIdf-weighted vectors of the comment and edit as a measure of relevance. (ii) **Cosine-InferSent** computes the cosine similarity between

	Candidates=5				Candidates=10				
	P@1	P@3	MRR	NDCG	P@1	P@3	P@5	MRR	NDCG
Cosine-TfIdf	0.266	0.519	0.470	0.597	0.137	0.291	0.444	0.296	0.453
Cosine-InferSent	0.228	0.597	0.471	0.600	0.115	0.305	0.497	0.302	0.461
RankNet	0.257	0.658	0.503	0.620	0.034	0.055	0.061	0.139	0.320
LambdaMART	0.310	0.726	0.549	0.661	0.152	0.384	0.593	0.352	0.502
Gated RNN	0.283	0.716	0.531	0.647	0.158	0.411	0.628	0.363	0.511
CmntEdit-MT	0.674	0.930	0.804	0.853	0.529	0.780	0.896	0.680	0.758
CmntEdit-CR	**0.710**	**0.944**	**0.828**	**0.871**	**0.593**	**0.830**	**0.924**	**0.730**	**0.796**

Table 3: Performance on Comment Ranking

comment and edit vectors generated by the state-of-the-art sentence embedding method InferSent (Conneau et al., 2017). (iii) **RankNet** (Burges et al., 2005) minimizes the number of inversions in ranking by optimizing a pair–wise loss function. We use a previous neural network implementation[4] with default settings. (iv) **LambdaMART** (Burges, 2010) leverages gradient boosted decision trees with a cost function derived from LambdaRank (Burges et al., 2007) for solving a ranking task. We use an existing python implementation[5], with a learning rate of 0.02 and 10 max leaf nodes. (v) **Gated Recurrent Neural Network** (Chung et al., 2014) models the sequence of words in comments and edits using pre-trained GloVe vectors as embedding units. Three fully-connected layers compute a final relevance score between comments and edits.

For the Edit Anchoring task, we chose the following popular classifiers as baselines: (i) **Random Forest** (Liaw et al., 2002) (ii) **Adaboost** (Rätsch et al., 2001) (iii) **Passive Aggressive** classifiers (Crammer et al., 2006) (iv) **Gated Recurrent Neural Network**. The features used in these models are based on both TfIdf scores, as well as sentence embedding features generated by InferSent. The Gated RNN model is trained with a task-specific fully connect layer for Edit Anchoring to compute the classification probability.

5.1.4 Model Variants

We train and evaluate several different variants of our neural architecture. They include: (i) **CmntEdit-CR**, a variant of our model only trained with data samples for the Comment Ranking task. (ii) **CmntEdit-EA**, a variant of our model only trained with data samples for the Edit Anchoring task. (iii) **CmntEdit-MT**, a variant of our model jointy trained on data samples of both Comment Ranking and Edit Anchoring tasks. Unless otherwise specified, all models use a standard context window size of 5 tokens[6].

	Candidates=5		Candidates=10	
	Acc	F1	Acc	F1
Passive-Aggr	0.581	0.533	0.716	0.262
RandForest	0.639	0.290	0.743	0.112
Adaboost	0.657	0.398	**0.751**	0.207
Gated RNN	0.696	0.651	0.665	0.539
CmntEdit-MT	0.635	0.587	0.619	0.468
CmntEdit-EA	**0.744**	**0.687**	0.726	**0.583**

Table 4: Performance on Edit Anchoring

5.2 Experimental Results

We now present and discuss the empirical results of our evaluation on both Comment Ranking and Edit Anchoring.

5.2.1 Comment Ranking

Table 3 summarizes results of the Comment Ranking task. Our models significantly outperform all the baselines in every setting and on all metrics. The results are statistically significant at $p < 0.01$ using the Wilcoxon signed-rank test (Smucker et al., 2007). Since the Comment Ranking task only has one true comment, the other comments being distractors, the P@1 result becomes especially important for practical applications. Our approach achieves 71% precision, which is significantly better than the 31% precision of the best baseline method (LambdaMART) with 5 candidate comments. Our model similarly outperforms the best baseline with 10 candidate comments, ob-

[4]https://github.com/shiba24/learning2rank

[5]https://github.com/jma127/pyltr

[6]We performed an extensive evaluation of context size with window sizes ranging from 0 to 50. From the result, we found the our model tends to perform best when the context window size is close to 5 tokens.

Comment Ranking	Candidates=5		Candidates=10	
	P@1	MRR	P@1	MRR
w/o Action	0.630	0.778	0.495	0.659
w/o Attention	0.625	0.775	0.488	0.655
w/o Hadamard	0.624	0.774	0.483	0.652
CmntEdit-CR	**0.710**	**0.828**	**0.593**	**0.730**
Edit Anchoring	Candidates=5		Candidates=10	
	Acc	F1	Acc	F1
w/o Action	0.713	0.668	0.684	0.556
w/o Attention	0.723	0.670	0.701	0.562
w/o Hadamard	0.722	0.660	0.706	0.558
CmntEdit-EA	**0.744**	**0.687**	**0.726**	**0.583**

Table 5: Ablation study

taining a P@1 score of 59.3%. Additionally, the higher scores on MRR and NDCG indicate that our approach consistently ranks the true comment higher than the baseline methods.

The performance of CmntEdit-MT is 2% and 5.3% worse than CmntEdit-CR on NDCG and P@1, respectively. Surprisingly, this suggests that training our model in a multi-task fashion leads to slightly lower scores, and a model specifically trained for the individual task of Comment Ranking is to be preferred.

5.2.2 Edit Anchoring

Table 4 shows the results for Edit Anchoring. Our method, CmntEdit-EA, outperforms the best baseline method, Gated-RNN, by 5.5% on F1 and 6.9% on accuracy. The improvements over all the baselines are statistically significant at a p-value of 0.01. The baseline classifiers including Passive-Aggressive, Random Forest and Adaboost have high accuracies, but low F1 scores. This is because of the imbalance between positive and negative samples in our data. Specifically, the number of negative samples is 4 times greater than the number of positive samples when the size of the candidate set is 5 – and even greater when it is 10. Therefore, the baseline classifiers tend to naively predict a negative label, which artificially boosts precision to the detriment of recall. In fact, Adaboost actually outperforms our models on accuracy when the candidate set size is 10, but yields a much lower F1 score.

As with Comment Ranking, the performance of CmntEdit-MT is slightly worse than CmntEdit-EA. Within the scope of our problem space, this reinforces the finding that targeted training seems to work better than joint training.

5.2.3 Ablation Study

To verify the effectiveness of our modeling choices, we evaluate performance in the absence of each of the following model components: 1. **w/o Action**: we remove the action encoding from the trainable function $\mathcal{G}$ and instead simply use $\mathcal{G}(\boldsymbol{u}, \boldsymbol{q}) = \boldsymbol{w}^T[\boldsymbol{u}^T\boldsymbol{q}]$. 2. **w/o Attention**: we remove the edit-based attention from Equation (2). Instead, we generate the relevance vector $\boldsymbol{h}$ as follows: $\boldsymbol{h} = \left[\text{mean}_{\text{col}}(\boldsymbol{S}^\dagger); \text{mean}_{\text{col}}(\boldsymbol{S}^\ddagger)\right]^T$. 3. **w/o Hadamard**: we use the inner product instead of the Hadamard product in the trainable function $\mathcal{G}$ as follows: $\mathcal{G}(\boldsymbol{u}, \boldsymbol{q}, a) = \boldsymbol{w}^T[\boldsymbol{u}^T\boldsymbol{q}; a]$.

Results in Table 5 show that each component improves the overall performance on both Comment Ranking and Edit Anchoring tasks, across our evaluation metrics. This indicates that our modeling choices are particularly suited to tackle the inherent challenges involved in modeling comment-edit relationship.

5.2.4 Qualitative Evaluation

Table 6 provides a few output examples from our model on the Comment Ranking task, demonstrating its ability to learn abstract connections between comments and edits. Due to space constraints, only one illustrative distractor is shown.

The first example shows an edit summarized by the high-level comment “date and capitalization corrections”. This comment is correctly assigned the highest relevance score by our model, despite the fact that no words are shared between the comment and edit. Meanwhile, one of the distractors has a lower score even though it shares the lexical item “Walgreens” with the context of the edit.

In the second example an entire sentence is removed by the editor. Again, although no words are shared between the comment and the edit, our model is correctly able to identify the delete operation, possibly by learning the common Wikipedia shorthand for deletions “rm”. Meanwhile, one of the distractors contains the phrase “St Helens”, which also appears in the edit, but is still assigned a lower score.

6 Conclusion and Future Work

In this paper, we have explored the relationship between comments and edits by defining two novel tasks: Comment Ranking and Editing Anchoring. In order to model the problem we collected a dataset with over 780K comment-edit pairs. Fur-

Case 1: Grammar fixing	
Pre-Edits	In December **of** 2012, **A** judge ordered Walgreens to pay $16.57 million to settle a lawsuit claiming...
Post-Edits	In December 2012, **a** judge ordered Walgreens to pay $16.57 million to settle a lawsuit claiming...
Comment	Date and capitalization corrections [Relevance Score: 4.050]
Distractor	Dane Cook's comments about Walgreens do not belong in "Facts" [Relevance Score: 3.028]
Case 2: Sentence removal	
Pre-Edits	**[http://www.pilkington.co.uk/energikare Pilkington energiKare] - Pilkington product from St Helens helping the environment**...
Post-Edits	(whole sentence removed)
Comment	rm advertising, other link to major employer noted in text seems ok at first glance. [Relevance Score: 5.192]
Distractor	Fixing link to Kevin Brown - You know I never knew he was from St Helens [Relevance Score: 3.479]

Table 6: Example of the edits and comments matched by the proposed model with one more deceptive distractor for each case. The scores for each comment/distractor are presented after the comment/distractor. The addition and removal in edit are highlighted in bold.

ther we proposed a hierarchical multi-layer neural network capable of tackling both our proposed tasks by encoding specific edit actions, such as additions and deletions, as well as document context. In our experiments we show that our approach outperforms several baselines by significant margins on both tasks, yielding a best score of 71% precision@1 for Comment Ranking and 74.4% accuracy for Edit Anchoring.

In future work we plan to explore sequences of revisions through the lifecycle of a document from creation to completion, with the ultimate goal of modeling document evolution. We also hope to apply our modeling approach to practical downstream applications, including: i) detecting completed to-dos based on related edits; ii) localizing the paragraphs that could be edited to address a given comments; iii) summarizing document revisions.

Acknowledgement

We thank the anonymous reviewers and members of the Knowledge Technologies and Intelligent Experiences group in Microsoft Research for helpful feedback. We also thank Ryen W. White for the thoughtful feedback on the statistical hypothesis testing.

References

Ablimit Aji, Yu Wang, Eugene Agichtein, and Evgeniy Gabrilovich. 2010. Using the past to score the present: Extending term weighting models through revision history analysis. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 629–638. ACM.

Ion Androutsopoulos and Prodromos Malakasiotis. 2010. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research*, 38:135–187.

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*, EACL '12, pages 356–366, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM.

Christopher J Burges, Robert Ragno, and Quoc V Le. 2007. Learning to rank with nonsmooth cost functions. In *Advances in neural information processing systems*, pages 193–200.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585.

Stephen E Fischer. 2013. Automated document revision markup and change control. US Patent 8,381,095.

Viola Ganter and Michael Strube. 2009. Finding hedges by chasing weasels: Hedge detection using wikipedia tags and shallow linguistic features. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 173–176. Association for Computational Linguistics.

Tovi Grossman, Justin Frank Matejka, and George Fitzmaurice. 2013. Hierarchical display and navigation of document revision histories. US Patent 8,533,593.

Andy Liaw, Matthew Wiener, et al. 2002. Classification and regression by randomforest. *R news*, 2(3):18–22.

Aurélien Max and Guillaume Wisniewski. 2010. Mining naturally-occurring corrections and paraphrases from wikipedia's revision history. In *LREC*.

Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. 2016. Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 83–84. International World Wide Web Conferences Steering Committee.

Sérgio Nunes, Cristina Ribeiro, and Gabriel David. 2008. Wikichanges: exposing wikipedia revision activity. In *Proceedings of the 4th International Symposium on Wikis*, page 25. ACM.

Sérgio Nunes, Cristina Ribeiro, and Gabriel David. 2011. Term weighting based on document revision history. *Journal of the American Society for Information Science and Technology*, 62(12):2471–2478.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Gunnar Rätsch, Takashi Onoda, and K-R Müller. 2001. Soft margins for adaboost. *Machine learning*, 42(3):287–320.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Robert Shaver. 2016. Document comment management. US Patent 9,418,054.

Mark D. Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 623–632, New York, NY, USA. ACM.

Elif Yamangil and Rani Nelken. 2008. Mining wikipedia revision histories for improving sentence compression. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 137–140. Association for Computational Linguistics.

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.

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: Unsupervised extraction of lexical simplifications from wikipedia. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 365–368. Association for Computational Linguistics.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

Fabio Massimo Zanzotto and Marco Pennacchiotti. 2010. Expanding textual entailment corpora fromwikipedia using co-training. In *Proceedings of the 2nd Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, pages 28–36.

Torsten Zesch. 2012. Measuring contextual fitness using error contexts extracted from the wikipedia revision history. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 529–538. Association for Computational Linguistics.

Xiaofeng Zhu, Diego Klabjan, and Patrick N. Bless. 2017. Semantic document distance measures and unsupervised document revision detection. In *IJCNLP*.