Visual Text Correction

Amir Mazaheri and Mubarak Shah

Center for Research in Computer Vision, University of Central Florida
amirmazaheri@knights.ucf.edu shah@crcv.ucf.edu

Abstract. This paper introduces a new problem, called *Visual Text Correction (VTC)*, i.e., finding and replacing an inaccurate word in the textual description of a video. We propose a deep network that can simultaneously detect an inaccuracy in a sentence, and fix it by replacing the inaccurate word(s). Our method leverages the semantic interdependence of videos and words, as well as the short-term and long-term relations of the words in a sentence. Our proposed formulation can solve the VTC problem employing an End-to-End network in two steps: (1)Inaccuracy detection, and (2)correct word prediction. In detection step, each word of a sentence is reconstructed such that the reconstruction for the inaccurate word is maximized. We exploit both Short Term and Long Term Dependencies employing respectively Convolutional N-Grams and LSTMs to reconstruct the word vectors. For the correction step, the basic idea is to simply substitute the word with the maximum reconstruction error for a better one. The second step is essentially a classification problem where the classes are the words in the dictionary as replacement options. Furthermore, to train and evaluate our model, we propose an approach to automatically construct a large dataset for the VTC problem. Our experiments and performance analysis demonstrates that the proposed method provides very good results and also highlights the general challenges in solving the VTC problem. To the best of our knowledge, this work is the first of its kind for the Visual Text Correction task.

1 Introduction

Text Correction (TC) has been a major application of Natural Language Processing (NLP). Text Correction can be in form of a single word auto-correction system, which notifies the user of misspelled words and suggests the most similar word, or an intelligent system that recommends the next word of an inchoate sentence. In this paper, we formulate a new type of Text Correction problem named *Visual Text Correction* (**VTC**). In VTC, given a video and an inaccurate textual description in terms of a sentence about the video, the task is to fix the inaccuracy of the sentence.

The inaccuracy can be in form of a phrase or a single word, and it may cause grammatical errors, or an inconsistency in context of the given video. For example, the word "car" in the sentence: "He is swimming in a car" is causing a textual inconsistency and the word "*hand*" is causing an inaccuracy in the context of the video (See Figure 1).

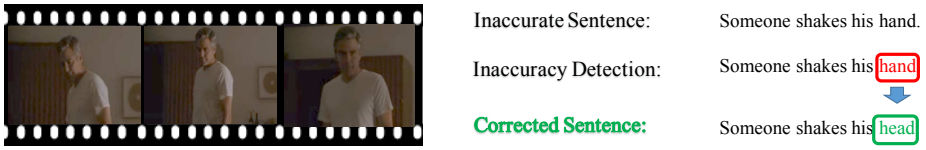

Fig. 1. One inaccurate sentence example for a given video. The VTC task is to find the inaccuracy and replace it with a correct word.

To formalize the problem, let sentence $\mathcal{S} = [w_1, w_2, ..., w_N]$ consisting of N words be an accurate description of the video $\mathcal{V}$, where $w_i \in \{0,1\}^{|V|}$, and $|V|$ is the number of words in our dictionary. For an inaccurate sentence $\widetilde{\mathcal{S}} = [\widetilde{w}_1, \widetilde{w}_2, ..., \widetilde{w}_N]$, the VTC task is to find the inaccurate word $\widetilde{w}_{t^*}$ where $1 \leq t^* \leq N$ and also to estimate the replacement word w_t. There can be several inaccurate words in a sentence; However, we train our system using sentences with just one inaccurate word. Nonetheless, we show that our trained network can be applied to sentences with multiple inaccurate words.

Our proposed formulation can solve the VTC problem employing an End-to-End network in two steps: (1)Inaccuracy detection, and (2)correct word prediction. Figure 2 shows the proposed framework of our approach. During the first step, we detect the inaccuracy by reconstruction, that is, we embed each word into a continuous vector, and reconstruct a word vector for each of the words in the sentence based on its neighboring words. A large distance between the reconstructed vector and the actual word vector implies an inaccurate word. For the second step, the basic idea is to simply substitute the word with the maximum reconstruction error for a better one. The second step is essentially a classification problem where the classes are the words in the dictionary as replacement options.

1.1 Motivations

Why Visual Text Correction? We believe that the VTC is very challenging and is a demanding problem to solve. During the last few years, the integration of computer vision and natural language processing (NLP) has received a lot of attention, and excellent progress has been made. Problems like Video Caption Generation, Visual Question Answering, etc., are prominent examples of this progress. With this paper, we start a new line of research which has many potential applications of VTC in real-world systems such as caption auto correction for video sharing applications and social networks, false tolerant text-based video retrieval systems, automatic police report validation, etc.

Why is VTC challenging? Given a large number of words in a dictionary, many different combinations of words can take place in a sentence. For example, there are $\binom{|V|}{3}$ possible triplet combinations of words from a dictionary of size $|V|$, which makes pre-selection of all possible correct combinations impractical.

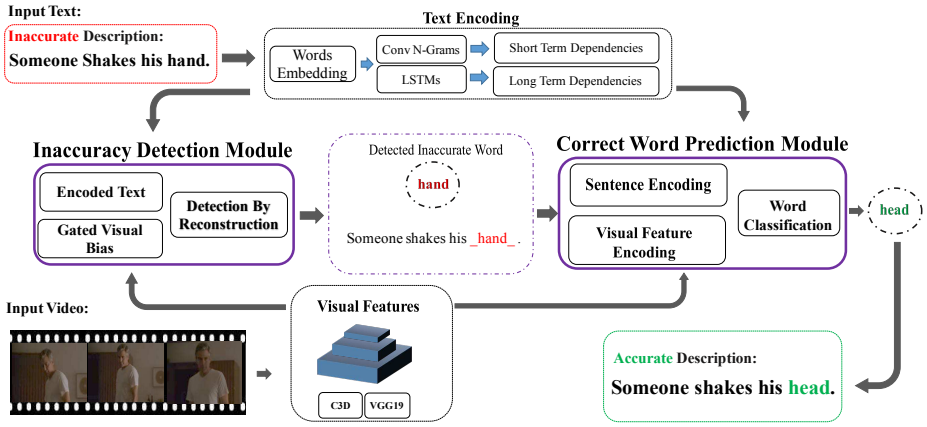

Fig. 2. Proposed framework for Visual Text Correction. The goal is to find and replace the inaccurate word in the descriptive sentence of a given video. There are two main modules: 1) The Inaccuracy Detection module finds the inaccurate word, and 2) the Correct Word Prediction module predicts an accurate word as a substitution. Both of these modules use the encoded text and visual features. The Inaccuracy Detection uses Visual Gating Bias to detect an inaccuracy and the Word Prediction Modules uses an efficient method to encode a sentence and visual features to predict the correct word.

Also, in many cases, even a meaningful combination of words may result in an incorrect or inconsistent sentence. Furthermore, sentences can vary in length, and the inaccuracy can be in the beginning, middle or at the end of a sentence. Last but not least, a VTC approach must find the inaccuracy and also choose the best replacement to fix it. The video can provide useful information in addition to text since some words of the sentence, like verbs and nouns, need to be consistent with the video semantics like objects and actions present in the video.

1.2 Contributions

The contribution of this paper is three-fold. First, we introduce the novel VTC problem. Second, we propose a principled approach to solve the VTC problem by decomposing the problem into inaccurate word detection and correct word prediction steps. We propose a novel sentence encoder and a gating method to fuse the visual and textual inputs. Third, we offer an efficient way to build a large dataset to train our deep network and conduct experiments. We also show that our method is applicable to sentences with multiple inaccuracies.

2 Related Work

In the past few years Deep Convolutional Neural Networks (CNNs) [1–4] have been demonstrated to be very useful in solving numerous Computer Vision prob-

lems like object detection [5, 6], action classification [7, 8]. Similarly, Recurrent Neural Networks (RNN) [9–11] and more specifically Long Short Term Memories(LSTM) [12] have been influential in dramatic advances in solving many Natural Language Processing (NLP) problems such as Translation [13], Paraphrasing [14], Question Answering [15–17], and etc. In addition to RNNs, several NLP works benefit from N-Grams [18, 19], and convolutional N-Grams [20, 13] to encode the neighborhood dependencies of the words in a sentence. The recent work in [13] show the superiority of N-Gram Convolutions over LSTM methods in sequence to sequence translation task. Therefore, in this paper we leverage N-Grams convolutions and Gating Linear Unit [21] in encoding the text and also incorporating visual features in our inaccuracy detection network. In addition, studies on encoding semantics of words [22, 23], phrases and documents [24, 25] into vectors have been reported. The main goal of all these studies is to represent the textual data in a way that preserves the semantic relations. In this research, we use the representation and distance learning to reconstruct each word of a sentence and find the inaccurate word based on the reconstruction error.

NLP and CV advances have motivated a new generation of problems, which are at the intersection of NLP and CV. Image/Video captioning [26–28] is to generate a description sentence about a given image/video. Visual Question Answering (VQA) [29–31, 30, 32–34] is to find the answer of a given question about a given image. In the captioning task, any correct sentence about the image/video can be acceptable, but in VQA, the question can be about specific details of the visual input. There are different types of the VQA problems, like multiple choice question answering [35], Textbook Question Answering (TQA) [36], Visual Dialogue [36], Visual Verification [37], Fill In the Blank (FIB) [38, 28, 39], etc. In addition to several types of questions in each of aforementioned works, different kinds of inputs have been used. Authors in [35] introduced a dataset of movie clips with the corresponding subtitles (conversations between actors) and questions about each clip. TQA [36] is a more recent form of VQA, where the input is a short section of elementary school textbooks including multiple paragraphs, figures, and a few questions about each. The aim of Visual Dialogue [36] is to keep a meaningful dialogue about a given photo, where a dialogue is a sequence of questions asked by a user followed by answers provided by system. Visual Knowledge Extraction [37] problem is to verify statements by a user (e.g. "Do horses fly?") from web crawled images.

Fill-In-the-Blank (FIB) [38, 28, 39] is the most related to our work. FIB is a Question Answering task, where the question comes in the form of an incomplete sentence. In the FIB task, the position of the blank word in each sentence is given and the aim is to find the correct word to fill in the blank. Although FIB is somehow similar to the proposed VTC task, it is not straightforward to correct an inaccurate sentence with a simple FIB approach. In FIB problem the position of the blank is given, however in VTC it is necessary to find the inaccurate word in the sentence first and then substitute it with the correct word.

Traditional TC tasks like grammatical and spelling correction have a rich literature in NLP. For instance, the authors in [40] train a Bayesian network

to find the correct misspelled word in a sentence. Other line of works like [41, 42], try to rephrase a sentence to fix a grammatical abnormality. In contrast to works in [40, 43, 41, 41, 42], there is no misspelled word in our problem, and we solve the VTC problem even for cases when the grammatical structure of the sentence is correct. Also, reordering the words of a sentence [42] cannot be the solution to our problem, since we need to detect and replace a single word while preserving the structure of the sentence. Moreover, this is the first work to employ the videos in the Textual Correction task.

3 Approach

To formulate the VTC problem, assume $\widetilde{\mathcal{S}} = [\widetilde{w}_1, \widetilde{w}_2, ..., \widetilde{w}_N]$ is a given sentence for the video $\mathcal{V}$. Our aim is to find the index of the incorrect word, t^*, and correct it with ${w^*}_{t^*}$ as follows:

$$(t^*, w^*_{t^*}) = \underset{1 \leq t \leq N, w_t \in \beta}{\arg\max}\ p((t, w_t)|\widetilde{\mathcal{S}}, \mathcal{V}), \tag{1}$$

where $w_i \in \{0,1\}^{|V|}$ is an one-hot vector representing the $i'th$ word of the sentence, $|V|$ is the size of our dictionary and N is the length of the sentence. Also, $\beta \subseteq V$ represents the set of all potential substitution words. Since t^* and ${w^*}_{t^*}$ are sequentially dependent, we decompose the Equation 1 into two sub-tasks: Inaccurate word detection as:

$$t^* = \underset{1 \leq t \leq N}{\arg\max}\, p(t|\widetilde{\mathcal{S}}, \mathcal{V}), \tag{2}$$

and the accurate word ${w^*}_{t^*}$ prediction as:

$${w^*}_{t^*} = \underset{w \in \beta}{\arg\max}\, p(w|\widetilde{\mathcal{S}}, \mathcal{V}, t^*\ ). \tag{3}$$

3.1 Inaccuracy Detection

We propose detection by reconstruction method to find the most inaccurate word in a sentence, leveraging the semantic relationship between the words in a sentence. In our approach, each word of a sentence is reconstructed such that the reconstruction for the inaccurate word is maximized. For this purpose, we build embedded word vector $x_i \in \mathbb{R}^{d_x}$ for each corresponding word w_i using a trainable lookup table $\theta_x \in \mathbb{R}^{|V| \times d_x}$. We exploit both Short Term and Long Term Dependencies employing respectively Convolutional N-Grams and LSTMs to reconstruct the word vectors.

Short-Term Dependencies: Convolutional N-Gram networks[13] capture the *short-term* dependencies of each word surrounding. Sentences can vary in length, and a proper model should not be confused easily by long sentences. The main

advantage of N-Gram approach is its robustness to disrupting words in long sentences, since it considers just a neighboring block around each word.

Let $X = [x_1; x_2; \ldots; x_N]$ be the stacked vectors representing embedded word vectors. Since the location of each word provides extra information about the correctness of that word in a sentence, we combine it with word vectors X. We denote $p_t \in \mathbb{R}^{d_x}$ as an embedded vector associated to the t'th *position* of each sentence, which is one row of the trainable matrix, $P \in \mathbb{R}^{N \times d_x}$. We use p_t values as gates for the corresponding word vectors x_t for each sentence and get final combination I as:

$$I_t = x_t \odot \sigma(p_t), \tag{4}$$

where $\odot$ denotes element-wise multiplication, and $I \in \mathbb{R}^{N \times d_x}$ is the input to a 1-D convolution with $2d_x$ filters and receptive field size of m. We call the resulting activation vectors $C \in \mathbb{R}^{N \times 2d_x}$. Furthermore, we use Gated Linear Units (GLU) [21] as the non-linear activation function. First, we split the C matrix in half along its depth dimension:

$$\begin{aligned} [A, B] &= C, \\ \Phi &= A \odot \sigma(B), \end{aligned} \tag{5}$$

where $A, B \in \mathbb{R}^{N \times d_x}$, and $\Phi = [\phi_1; \phi_2; \ldots; \phi_N]$, and $\phi_i \in \mathbb{R}^{d_x}$. The idea is to use the B matrix as gates for the matrix A. An open gate lets the input pass, and a close gate changes the input to zero. By stacking multiple 1-D convolutions and GLU activation functions the model goes deeper and the receptive field becomes larger. The output, Φ, from each layer is the input, I, for the next layer. We call the final output Φ, from the last Convolutional N-Grams layer, $\hat{X}^C \in \mathbb{R}^{N \times d_x}$. In Figure 3, we illustrate one layer of the N-Grams encoding.

Long-Term Dependencies: Recurrent networks, and specifically LSTMs, have been successfully used to capture the *long-term* relations in sequences. Long-term relations are beneficial to comprehend the meaning of a text and also to find the possible inaccuracies. To reconstruct a word vector based on the rest of the sentence using LSTMs, we define a left fragment and a right fragment for each word in a sentence. The left fragment starts from the first word of the sentence to one word before the word under consideration; and the right fragment is from the last word of the sentence to one word after the word under consideration in a reverse order. We encode each of the left and right fragments with a LSTM and extract the last hidden state vector of the LSTM as the encoded fragment:

$$\hat{x}_t^R = W_c \times [u_t^l | u_t^r], \tag{6}$$

where $u_t^{l/r} \in \mathbb{R}^h$ are the encoded vectors of left/right fragments of the t'th word, and $W_c \in \mathbb{R}^{d_x \times 2h}$ is a trainable matrix to transform the $[u_t^l | u_t^r]$ into the $\hat{x}_t^R$.

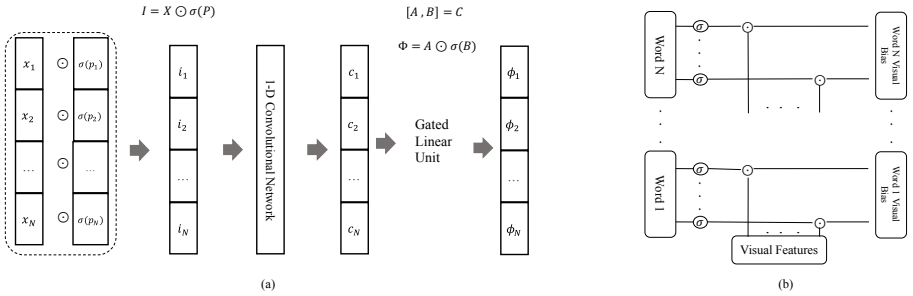

Fig. 3. (a) One layer of Convolutional Text Encoding which captures the neighboring relationships. To extend one layer to multiple layers, we simply consider the ϕ_i vectors as I_i for the next layer. (b) Our proposed Visual Gating Bias process. Given each word vector, we filter out some parts of a given visual feature through a gating process.

Detection Module: We design a module to learn the distance between an actual word vector x_t and the reconstructed $\hat{x}_t$ as explained above. This module learns to assign a larger distance to the inaccurate words and reconstruct the predictions as follows:

$$\mathcal{D}_t = W_d \times (\frac{\hat{x}_t}{\|\hat{x}_t\|} \odot \frac{x_t}{\|x_t\|}), \tag{7}$$

where $W_d \in \mathbb{R}^{1 \times d_x}$, and $\mathcal{D}_t$ is a scalar. $\hat{x}_t$ is the output of the text encoding; namely, $\hat{x}_t = \hat{x}_t^C$ for Convolutional N-Grams or $\hat{x}_t = \hat{x}_t^R$ in case of Recurrent Networks. Next, we combine both as a vector $\hat{x}_t = \hat{x}_t^R + \hat{x}_t^C$ to capture both long term and short term dependencies of a sentence. We design our distance module as a single layer network for simplicity; however, it can be a deeper network.

Visual Features as Gated Bias: Visual features can contribute in finding the inaccuracy in a video description; however, it can be very challenging since some words may not correspond to any visible form or shape (e.g. 'weather'), while some others may correspond to distinct visual appearances (e.g. 'cat'). We introduce a gating model to incorporate the visual features to measure the inconsistency of each word. The main idea is to find a dynamic vector for the visual features which changes for each word as follows (see Figure 3):

$$\Psi_{\mathcal{V}} = W_v \times \Omega(\mathcal{V}), \tag{8}$$

where $\Omega(\mathcal{V}) \in \mathcal{R}^{d_v}$ is the visual feature vector, and $W_v \in \mathcal{R}^{d_x \times d_v}$ is a transformation matrix for the visual features. We build the visual bias v_t for each word vector x_t:

$$v_t = \frac{\Psi_{\mathcal{V}}}{\|\Psi_{\mathcal{V}}\|} \odot \sigma([W_g \times x_t]), \tag{9}$$

and $W_g \in \mathcal{R}^{d_x \times d_x}$ is transformation matrix, and $\|.\|$ denotes L2-Norm of a vector. The Sigmoid ($\sigma(.)$) operator bounds its input into $(0, 1)$. It makes the model capable of refusing or accepting visual features dynamically for each word in a sentence.

The most intuitive way to incorporate the V vectors in Equation 7, is to use them as a bias term. In fact, the features which are refused by the word gates will have zero value and will act as neutral. Therefore, we use the following updated form of Equation 7 with the video contribution:

$$\mathcal{D}_t = W_d \times (\frac{\hat{x}_t}{\|\hat{x}_t\|} \odot \frac{x_t}{\|x_t\|} \oplus v_t), \tag{10}$$

where $\oplus$ denotes element-wise summation.

For the last step of the detection process, we find the word with maximum $\mathcal{D}$ value:

$$t^* = \underset{1 \leq t \leq N}{\arg\max} (\mathcal{D}_t). \tag{11}$$

Detection loss: We use the cross-entropy as detection loss function. Given the ground-truth one-hot vector $y \in \{0, 1\}^N$, which indicates the inaccurate word, and the $T^* = softmax(D)$ as probabilities, we compute the detection loss l_d.

3.2 Correct Word Prediction

The second stage of our proposed method to solve the VTC problem is to predict a substitute word for the inaccurate word. Proposed correct word prediction consists of three sub-modules: 1- Text Encoder, 2- Video Encoder, and 3- Inference sub-modules.

Text Encoder: This sub-module must encode the input sentence in such a way that the network be able to predict the correct word for the t^*'th word. We leverage the reconstructed word vectors $\hat{x}_t$ in equation 7, since these vectors are rich enough to detect an inaccuracy by reconstruction error. We can feed the output of inaccuracy detection, t^*, to our accurate word prediction network; however, the $argmax$ operator in Equation 11 is not differentiable and prevents us to train our model End-to-End. To resolve this issue, we approximate the Equation 11 by vector $T^* = Softmax(D)$, which consists of probabilities of each of N words being incorrect in the sentence. We build the encoded text vector q_t:

$$q_t = tanh(W_q \times \hat{x}_t), \tag{12}$$

where $W_q \in \mathbb{R}^{d_q \times d_x}$ is trainable matrix. $q_t \in \mathbb{R}^{d_q}$ is in fact a hypothetical representation of the textual description. To be more specific, q_t is the encoded sentence, assuming that the word t is the incorrect word, which is to be replaced

by a blank, according to the Equation 12. Finally, the **textual representation** $u_q \in \mathbb{R}^{d_q}$, is formulated as a weighted sum over all q_t vectors:

$$u_q = \sum_{t=1}^{N} T_t^* q_t. \tag{13}$$

Note that, due to the "$tanh(.)$" operator in Equation 12, both q_t and u_q vectors have bounded values.

Video Encoding: We leverage the video information to find the accurate word for t^*'th word of a sentence. While the textual information can solely predict a word for each location, visual features can help it to predict a better word based on the video, since the correct word can have a specific visual appearance. We extract the visual feature vector $\Omega(\mathcal{V})$ and compute our video encoding using a fully-connected layer:

$$u_{\mathcal{V}} = tanh(W_{\mathcal{V}} \times \Omega(\mathcal{V})), \tag{14}$$

where $W_{\mathcal{V}} \in \mathbb{R}^{d_q \times d_v}$, and $u_{\mathcal{V}} \in \mathbb{R}^{d_q}$ is our visual representation, which has bounded values. For simplicity, we have used just one layer video encoding; however, it can be a deeper and more complicated network.

Inference: For the inference, we select the correct substitute word from the dictionary. In fact, this amounts to a classification problem, where the classes are the words and the inputs are the textual representation and the visual features:

$$w_{t^*}^* = \arg\max_{w \in \beta}(W_i \times [u_q + u_{\mathcal{V}}]), \tag{15}$$

where $W_i \in \mathbb{R}^{|\beta| \times d_q}$. Finally, we use cross-entropy to compute the correct word prediction loss, namely l_f. The total loss for our VTC method is $l = l_f + l_d$ and we train both sub-tasks together.

4 Dataset and Experiments

4.1 Dataset

In this section, we describe our visual text correction dataset and the method to generate it. The main idea behind our approach to build a dataset for the VTC task is to remove one word from each sentence and substitute it with an inaccurate word; however, there are several challenges to address in order to build a realistic dataset. Here, we list a few and also propose our approach to address those challenges.

Our goal is to build a large dataset with a variety of videos with textual descriptions. We require that the vocabulary of the dataset and the number of video samples be large enough to train a deep network; hence we choose "Large Scale Movie Description Challenge(LSMDC)" dataset [38, 44], which is one of the

largest video description datasets available. Also, LSMDC has been annotated for "Video Fill In the Blank (FIB)" task. In FIB dataset, each video description contains one or more blanks, which needs to be filled in. For the VTC problem, we introduce inaccurate word in place of the blanks in FIB dataset. If there is more than one blanks in a sentence of the FIB dataset, we generate multiple examples of that sentence.

Note that there are some important points related to selection of the replacement words, which we need to keep in mind. First, there shouldn't be a high correlation between the original and replacement words. For example, if we exchange the word "car" with "bicycle" frequently, any method will be biased and will always suggest replacing "bicycle" with "car" in all sentences. Second, we want our sentences to look natural even after the word substitution. Therefore, the replacement word should have the same "Part Of Speech" (POS) tag. For example, a singular verb is better to be replaced by another singular verb.

It is costly to manually annotate and select the replacement words for each sample, because of the significant number of videos, and the vast vocabulary of the dataset. Also, it is hard for the human annotators to prevent the correlation between the original and replacement words. We have considered all the mentioned points to build our dataset. Following we describe how we build a proper dataset for the VTC problem.

Random Placement: In this approach, for each annotated blank in the LSMDC-FIB dataset, we place a randomly selected word from dictionary. This approach evidently is the most straightforward and simple way to introduce the incorrect word. However, in this method, a bias towards some specific words may exist, since the selected inaccurate words may not follow the natural distribution of the words in the dictionary. For example, we have many words with less than 4 or 5 occurrences in total. By Random Placement approach, rare words and the words with high frequencies have the same chance to show up as an inaccurate word. This increases the rate of "inaccurate occurrences to accurate occurrences" for some specific words. This imbalanced dataset allows any method to detect the inaccuracy just based on the word itself not the the word in the context. Also, since replacement and original words may not take the same POS tag, Random Placement approach cannot meet one of the requirements mentioned above.

POS and Natural Distribution: Due to the weaknesses of the Random Placement, we introduce a more sophisticated approach that selects the inaccurate words from a set of words with the same tag as the original (or accurate) word. We first extract the POS tags of all the words from all the sentences using Natural Language Toolkit (NLTK) [45], resulting in 32 tags. Let S_r be the set of all the words that takes the tag r ($1 \leq r \leq 32$) at least once in the training sentences. To find a replacement for the annotated blank word w with the tag r in a sentence, we draw a sample from S_r and use it as the inaccurate word. Obviously, some tags are more common than the others in natural language and as a result the incorrect words are similarly the same.

To draw a sample from a set, we use the distribution of the words in all sentences. As a result, the words with more occurrences in the training set have more chance to be appeared as an inaccurate word. Therefore, the rate of incorrect to correct appearances of different words are close to each other. With this approach, we prevent the rare words to be chosen as the inaccurate word frequently and vice versa.

4.2 Results

Detection Experiments: In this subsection, we present our results for detection module and examine our method with various settings. The results are summarized in Table 1. Following we explain each experiment in more details.

Random guess is to select one of the words in the sentence randomly as the inaccurate word. In *Text Only Experiments* part of Table 1, we compare all the blind experiments, where no visual features are used to detect the inaccuracy. *Vanilla LSTM* uses a simple LSTM to directly produce the $\mathcal{D}_t$(Equation 7) out of its hidden state using a fully connected layer.

One-Way Long-Term Dependencies uses just u_l in Equation 6. *Long-Term Dependencies* experiment uses Recurrent Neural Networks method explained in Section 3.1. *Convolutional N-Grams w/o Position Embedding* uses just Convolutional N-Grams, however, without the contribution of the positions of each word explained in Section 3.1 while *Convolutional N-Grams* is the complete explained module in Section 3.1. These two experiments show the effectiveness of our proposed words position gating, and finally, *Convolutional N-Grams + Long-Term Dependencies* uses the combination of Convolutiona N-Grams and RNNs as mentioned in Section 3.1. The last experiment reveals the contribution of both short-term and long-term dependencies of words in a sentence for the TC task.

To further study the strength of our method to detect the wrong words, we compare our method with a *Commercial Web-App*[1]. This application can detect structural or grammatical errors in text. We provide 600 random samples from the test set to the web application and examine if it can detect the inaccuracy. In Table 1, we show the comparison between our method and the aforementioned web application. This experiment shows the superiority of our results and also the quality of our generated dataset.

In *Video and Text Experiments* part of the Table 1, we show experiments with both video and text. **Visual Gated Bias** experiment shows the capability of our proposed formulation to leverage the visual features in the detection subtask. To show the superiority of our visual gating method, we conduct *Visual Feature Concatenation* experiment. In this experiment, we combine the visual feature vector $\Omega(\mathcal{V})$ with each of the vectors x_t and $\hat{x}_t$ in Equation 7 using concatenation and a fully connected layer. For these experiments, we have used the pre-trained C3D [8] to compute the $\Omega(\mathcal{V})$.

[1] www.grammarly.com

Table 1. Detection Experiments Results. For these experiments we just evaluate the ability of different models to localize the inaccurate word.

Method	**Accuracy (%)**
Random	8.3
Text Only Experiments	
Commercial Web-App	18.8
Vanilla LSTM	28.0
One-Way Long-Term Dependencies	58.0
Long-Term Dependencies	67.2
Conv N-Grams w/o Position Embedding	66.8
Conv N-Grams	69.0
Conv N-Grams + Long-Term Dependencies	*72.5*
Video and Text Experiments	
Conv N-Grams + Long-Term Dependencies + Visual Feature Concatenation	72.8
Conv N-Grams + Long-Term Dependencies + Visual Gated Bias	**74.5**

4.3 Correction Experiments

In Table 2, we provide our results for the correction task. Note that, the correction task is composed of both inaccurate word detection and correct word predictions sub-tasks; thus, a correct answer for a given test sample must have the exact position of the inaccurate word and also the true word prediction ($(t^*,w^*_{t^*})$ in Equation 1).

Our Model - Just Text experiment demonstrates our method performance with only textual information. *Our Model With C3D Features* uses both video and text, with C3D [8] features as visual features. Similarly, *Our Model With VGG19 Features* shows the results when VGG19 [46] features are the visual input. In *Our Pre-trained Detection Model + Pre-Trained FIB* [39] experiment we use our best detection model from Table 1 to detect an inaccurate word. We remove the inaccurate word and make an incomplete sentence with one blank. Then, we use one of the pre-trained state of the art FIB methods [39], which uses two staged Bi-LSTMs (LR/RL LSTMs) for text encoding + C3D and VGG19 features + temporal and spatial attentions, to find the missing word of the incomplete sentence. We show the superiority of our method which has been trained End-to-End. In both of detection (Table 2) and correction (Table 1) tasks, there are accuracy improvements after including visual features. We also report the Mean-Average-Precision (MAP) metric, to have a comprehensive comparison. To measure the MAP, we compute $N \times |\beta|$ scores for all the possible $(t^*, w^*_{t^*})$.

4.4 Multiple Inaccuracies

Here, we show that our method is capable of to be generalized to sentences with more than one inaccurate words. We conduct a new experiment with multiple inaccuracies in the test sentences and show the results in Table 3. In fact, we replace all the annotated blank words in the LSMDC-FIB test sentences with an inaccurate word. We assume that the number of inaccuracies, k, is given

Table 2. Text Correction Experiments Results. For the correction task, a model needs to successfully locate the inaccurate word and provides the correct substitution.

Method	**Accuracy (%)**	**MAP (%)**
Random	0.04	≃0
Vanilla LSTM - Just Text	17.2	17.7
Our Model - Just Text	35.2	36.9
Our Pre-trained Detection Model + Pre-Trained FIB [39]	36.0	38.6
Our Model With C3D Features	38.6	39.8
Our Model With VGG19 Features	38.8	40.1
Our Model With VGG19 + C3D Features	**38.9**	**40.7**

Table 3. Detection and Correction results for sentences with multiple inaccuracies. Two types of Accuracy evaluations are provided. (1) Word-Based (WB) Accuracy: All correctly fixed incorrect words are counted independently. (2) Sentence-Based (SB) Accuracy: All inaccurate words in a sentence must be fixed correctly. Similarly, two types of MAP is reported: (1) WB-MAP, in which, one AP per each incorrect word is computed. (2) SB-MAP, in which, one AP per each sentence, including all the k incorrect words, is computed. k represents the number of inaccuracies in each sentence.

k =	1	2	3	4	All	1	2	3	4	All
# Of Test Samples	1805	4856	5961	520	30349	1805	2428	1987	130	9575
Detection	WB-Acc. (%)					SB-Acc. (%)				
Vanilla LSTM - Just Text	59	63	67	68	66	59	37	27	18	36
Our Method - Just Text	80	81	80	80	80	80	65	48	37	59
Our Method - Text + Video	**85**	**83**	**83**	**82**	**83**	**85**	**68**	**54**	**39**	**63**
Correction	WB-Acc. (%)					SB-Acc. (%)				
Our Method - Just Text	19	12	12	11	3	19	2	≃ 0	≃ 0	5
Our Method - Text + Video	**24**	**18**	**17**	**17**	**18**	**24**	**4**	≃ 0	≃ 0	**7**
Correction	WB-MAP (%)					SB-MAP (%)				
Our Method - Just Text	30	14	10	**8**	12	30	15	11	9	17
Our Method - Text + Video	**35**	**17**	**11**	7	**14**	**35**	**18**	**12**	**10**	**19**

for each test sample, but the model needs to locate them. To select the inaccuracies in each sentence, we use the LSMDC-FIB dataset annotations. Note that in training we use sentences that contain just one inaccurate word, similar to previous experiments. During the test time, we modify the Equation 11 to $t^*_{i=1,..,k} = arg\,kmax(\mathcal{D}_t)$, where $arg\,kmax$ returns the top k inaccurate word candidates. Number of inaccurate words in our test set sentences reaches up to 10 words. However, in Table 3, we show the detection results for sentences with each $k \leq 4$ value separately, and also the overall accuracy for all the k values.

4.5 Qualitative Results

We show a few VTC examples in Figure 4. For each sample, we show frames of a video and corresponding sentence with an inaccuracy. We provide the qualitative results for each example using our "Just Text" and "Text + Video" methods. We

Fig. 4. Here we show four samples of our test results. For each sample, we show a video and an inaccurate sentence, the detected inaccurate word, and the predicted accurate word for substitution. The green color indicates a correct result while the red color shows a wrong result.

show two columns for the detection and correct word prediction. The green and red colors respectively indicate true and false outputs. Note that, for the VTC task, just a good detection or prediction is not enough. Both of these sub-tasks are needed to solve the VTC problem. For example, the left bottom example in Figure 4 shows a failure case for both "Just Text", and "Text + Video", although the predicted word is correct using "Text + Video".

5 Conclusion

We have presented a new formulation of text correction problem, where the goal is to find an inaccuracy in a video description, and fix it by replacing the inaccurate word. We propose a novel approach to leverage both textual and visual features to detect and fix the inaccurate sentences, and we show the superior results are obtained our approach. Moreover, we introduce an approach to generate a suitable dataset for VTC problem. Our proposed method provides a strong baseline for inaccuracy detection and correction tasks for sentences with one or multiple inaccuracies. We believe that our work is a step forward in the research related to intersection of Natural Language Processing and Computer Vision. We hope that this work lead to more exciting future researches in VTC.

ACKNOWLEDGMENTS This material is based upon work supported by the National Science Foundation under Grant No. 1741431. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012)
2. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
3. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR. (2015)
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. (2009)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778
7. Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402 (2012)
8. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: ICCV. (2015)
9. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing **45**(11) (1997)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8) (1997)
11. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
12. Malinowski, M., Rohrbach, M., Fritz, M.: Ask your neurons: A neural-based approach to answering questions about images. In: CVPR. (2015)
13. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. arXiv preprint arXiv:1705.03122 (2017)
14. Chen, D.L., Dolan, W.B.: Collecting highly parallel data for paraphrase evaluation. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, Association for Computational Linguistics (2011) 190–200
15. Bordes, A., Usunier, N., Chopra, S., Weston, J.: Large-scale simple question answering with memory networks. arXiv preprint arXiv:1506.02075 (2015)
16. Kumar, A., Irsoy, O., Su, J., Bradbury, J., English, R., Pierce, B., Ondruska, P., Gulrajani, I., Socher, R.: Ask me anything: Dynamic memory networks for natural language processing. arXiv preprint arXiv:1506.07285 (2015)
17. Weston, J., Chopra, S., Bordes, A.: Memory networks. arXiv preprint arXiv:1410.3916 (2014)
18. Zhang, H., Chiang, D.: Kneser-ney smoothing on expected counts.
19. Chen, S.F., Goodman, J.: An empirical study of smoothing techniques for language modeling. In: Proceedings of the 34th annual meeting on Association for Computational Linguistics, Association for Computational Linguistics (1996) 310–318
20. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188 (2014)
21. Dauphin, Y.N., Fan, A., Auli, M., Grangier, D.: Language modeling with gated convolutional networks. arXiv preprint arXiv:1612.08083 (2016)

22. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
23. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS. (2013) 3111–3119
24. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of the 31st International Conference on Machine Learning (ICML-14). (2014) 1188–1196
25. Dai, A.M., Olah, C., Le, Q.V.: Document embedding with paragraph vectors. arXiv preprint arXiv:1507.07998 (2015)
26. Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: A neural image caption generator. In: CVPR. (2015)
27. Johnson, J., Karpathy, A., Fei-Fei, L.: Densecap: Fully convolutional localization networks for dense captioning. arXiv preprint arXiv:1511.07571 (2015)
28. Yu, Y., Ko, H., Choi, J., Kim, G.: End-to-end concept word detection for video captioning, retrieval, and question answering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 3165–3173
29. Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence Zitnick, C., Parikh, D.: Vqa: Visual question answering. In: ICCV. (2015)
30. Ren, M., Kiros, R., Zemel, R.: Exploring models and data for image question answering. In: NIPS. (2015)
31. Malinowski, M., Fritz, M.: A multi-world approach to question answering about real-world scenes based on uncertain input. In: NIPS. (2014)
32. Agrawal, A., Lu, J., Antol, S., Mitchell, M., Zitnick, C.L., Batra, D., Parikh, D.: Vqa: Visual question answering. arXiv preprint arXiv:1505.00468 (2015)
33. Xiong, C., Merity, S., Socher, R.: Dynamic memory networks for visual and textual question answering. arXiv preprint arXiv:1603.01417 (2016)
34. Zhang, P., Goyal, Y., Summers-Stay, D., Batra, D., Parikh, D.: Yin and yang: Balancing and answering binary visual questions. arXiv preprint arXiv:1511.05099 (2015)
35. Tapaswi, M., Zhu, Y., Stiefelhagen, R., Torralba, A., Urtasun, R., Fidler, S.: Movieqa: Understanding stories in movies through question-answering. In: CVPR. (2016)
36. Nadeem, F., Ostendorf, M.: Language based mapping of science assessment items to skills. In: Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications. (2017) 319–326
37. Sadeghi, F., Divvala, S.K., Farhadi, A.: Viske: Visual knowledge extraction and question answering by visual verification of relation phrases. In: CVPR. (2015)
38. Maharaj, T., Ballas, N., Courville, A., Pal, C.: A dataset and exploration of models for understanding video data through fill-in-the-blank question-answering. arXiv preprint arXiv:1611.07810 (2016)
39. Mazaheri, A., Zhang, D., Shah, M.: Video fill in the blank using lr/rl lstms with spatial-temporal attentions. In: ICCV 2017. (2017)
40. Mays, E., Damerau, F.J., Mercer, R.L.: Context based spelling correction. Information Processing & Management **27**(5) (1991) 517–522
41. Wu, C.H., Liu, C.H., Harris, M., Yu, L.C.: Sentence correction incorporating relative position and parse template language models. IEEE Transactions on Audio, Speech, and Language Processing **18**(6) (2010) 1170–1181
42. Wagner, R.A.: Order-n correction for regular languages. Communications of the ACM **17**(5) (1974) 265–268

43. Suhm, B., Myers, B., Waibel, A.: Multimodal error correction for speech user interfaces. ACM transactions on computer-human interaction (TOCHI) **8**(1) (2001) 60–98
44. Rohrbach, A., Rohrbach, M., Tandon, N., Schiele, B.: A dataset for movie description. In: CVPR. (2015)
45. Loper, E., Bird, S.: Nltk: The natural language toolkit. In: Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1, Association for Computational Linguistics (2002) 63–70
46. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014)