SECTION I.Introduction

Abstractive summarization aims at generating a short and concise summary from a long source text where the summary captures key information of the source text. Recent abstractive summarization models have shown high performance in automatic evaluation metrics such as ROUGE with the help of pre-trained language models [1], [2], [3]. Despite their high performance, they often generate a factually-inconsistent summary with respect to a source text. According to the previous studies [4], [5], [6], [7], about 30% of summaries generated by abstractive summarization models contain factual errors. Table 1 shows an example of an inconsistent summary generated by the BART summarizer [8], one of the widely-used models for abstractive summarization. According to this table, 147 people including 142 students are killed in the source text, but the 147 people are in critical condition in the BART-generated summary, which is factually inconsistent with the source text. Since an inconsistent summary limits the feasibility of abstractive summarization, it is significant to ensure factual consistency of generated summaries.

TABLE 1 An Example of a Factual Inconsistent Summary Generated by a BART Summarizer, the Corrected Summary Using a Entity-Level Correction Model, and the Golden Summary. The Underline in the Generated Summary Implies an Inconsistent Part

Table 1- Table 1- 
An Example of a Factual Inconsistent Summary Generated by a BART Summarizer, the Corrected Summary Using a Entity-Level Correction Model, and the Golden Summary. The Underline in the Generated Summary Implies an Inconsistent Part

An Example of a Factual Inconsistent Summary Generated by a BART Summarizer, the Corrected Summary Using a Entity-Level Correction Model, and the Golden Summary. The Underline in the Generated Summary Implies an Inconsistent Part

The previous studies have attempted to ensure the factual consistency of a generated summary by incorporating some external knowledge into an abstractive summarization model [4], [6], [10], [11], [12], correcting a generated summary by replacing its inconsistent tokens with the tokens in a source text [13], or rewriting a summary through a revision model [9], [13]. Among them, the approach of correcting inconsistent tokens has achieved state-of-the-art performance. This approach focuses on the fact that most factual inconsistencies occur at the entities such as numbers and pronouns [14], [15]. Thus, it modifies only small amount of a generated summary so that it preserves the fluency of the summary. Furthermore, it has additional advantage that it can be applied to any abstractive summarization models. However, regarding only entities as correcting candidates inevitably results in a limitation of non-entity inconsistency because factual inconsistencies do not occurs only at entities, but also at any tokens in a summary. For instance, in Table 1, even though the summary by an entity corrector modifies the number of 142 in the BART summary as 19, it is yet inconsistent with the source text. Therefore, to ensure the factual consistency of a generated summary, a correction should be made to inconsistent phrases such as “in critical condition.”

This paper proposes a generalized fact corrector for improving factual consistency in abstractive summarization. The proposed fact corrector aims to correct any tokens in a generated summary, while ordinary fact correctors rectify only the entities in the summary. To achieve this goal, the proposed corrector consists of three modules of a summary fact checker, a token fact checker, and a fact emender. The proposed fact corrector regards all tokens in a summary as correcting candidates. However, one consideration of the proposed corrector is that most tokens in a generated summary are factually consistent. Thus, rather than trying to edit all tokens in the summary, the summary fact checker first verifies if the generated summary is factually consistent with its source text. If the summary is decided not to coincide with the source text by the summary fact checker, then the token fact checker finds out inconsistent tokens in the summary and masks them out for further editing. Finally, similarly to answer-span selection in the question answering tasks [16], [17], [18], the fact emender selects a span from the source text for each mask in the masked summary and then replaces the mask with the span. Since these three modules are closely related and affect one another, they are jointly trained under a joint learning framework [19], [20] to improve the performance of each module. As a result, the proposed fact corrector corrects the inconsistent summary at the token-level without sacrificing correction efficiency.

The proposed token-level fact corrector is verified with three summarization benchmark data sets of CNN/DM, XSUM, and Gigaword. The experimental results prove that the proposed fact corrector outperforms the entity-level fact correctors in factual consistency metrics. In particular, the proposed corrector achieves 81.04 of BERTScore [21] in Gigaword data set, which is 1.03 higher than the state-of-the-art entity-level fact correction model. It is also shown that the proposed corrector generates more factually-consistent summaries even for K2019 data, a data set manually annotated for fact-consistent summarization. The performance improvement of the proposed correction is obtained from token-level corrections, since the proposed correction rectifies factually-inconsistent tokens as well as factually-inconsistent entities.

The main contributions of the proposed fact corrector are as follows:

This is, to the best of our knowledge, the first attempt to correct token-level factual inconsistency of a generated summary in abstractive summarization. Since token-level fact correction is more general than entity-level fact correction from a granularity perspective, the proposed fact corrector outperforms other correctors in factual consistency performance.

The proposed fact corrector is yet efficient even if it deals with token-level granularity. Since the proposed corrector first determines whether a given summary is factually inconsistent with respect to a source text and then tries to correct only inconsistent tokens, the correction need not be made for all tokens.

The proposed fact corrector is evaluated with three popular summarization benchmarks and one manually-annotated data set. The experimental results validate the effectiveness of the proposed token-level correction.

The rest of this paper is organized as follows. Section II gives an overview of the fact correction in abstractive summarization, and Section III presents the proposed token-level fact correction. Section IV explains the details of generating a data set for the evaluation of fact correctors and how to train the proposed fact corrector. The experimental settings and results are given in Section V. Finally, Section VI draws some conclusions.

SECTION II.Related Work

The studies to improve factual consistency of abstractive summarization can be categorized into two types. The first type is to incorporate some external knowledge into an abstractive summarization model. For example, Cao et al. [4] extracted triples from a source text using an open information extractor and a dependency parser. Then, the extracted triples are incorporated into an abstractive summarization model to generate a final summary. Li et al. [12] and Falke et al. [6] utilized entailment information to produce a summary, and Zhao et al. [22] adopted a fact verification model to score summary candidates and then incorporated the candidate scores into the beam search of a summary generation model. On the other hand, Zhu et al. [10] first extracted a graph from a source text and applied a graph attention network to representing the graph. Then, they fused the represented graph into a transformer-based encoder-decoder architecture to generate a summary. The main problem of this type is that it is model-specific. That is, the methods in this type work only in conjunction with a specific summarization model.

The other type is a post-editing correction. Given a system-generated summary and a source text, a fact corrector rectifies the summary so that the summary gets factually consistent with the source text. This type is model-agnostic because it takes a generated summary as its input. According to the degree of correction, this type is further divided into two kinds of methods: a rewriting method and a span correction method. The rewriting method generates a new summary through an auto-regressive sequence-to-sequence model [13]. The input to the sequence-to-sequence model is a concatenation of a generated summary and a source text. Then, the sequence-to-sequence model generates a new summary auto-regressively at the token level. However, this method generates a new summary regardless of the consistency of the generated summary. That is, even when a given summary is already factually consistent with a source text, the method tries to generates a new summary by modifying some tokens in the given summary. As a result, the newly-generated summary may differ significantly from the given summary.

The span correction method is to correct inconsistent text spans in a generated summary by replacing the spans with correct text spans in a source text [9]. Unlike the rewriting method, this method corrects only small parts of a generated summary so that the corrected summary does not differ significantly from the given summary. Since most factual inconsistencies occur at entities, this method considers only entities as correcting candidates. That is, it first finds inconsistent entities in a generated summary and then replaces the entities with other entities in a source text. Although this method achieves a higher performance in factual consistency than the rewriting method, it tries to correct all entities without verifying the inconsistencies of the entities. Furthermore, it has limitations that inconsistent non-entities cannot be corrected and it requires a named entity recognizer to detect entities in a summary and a source text.

SECTION III.Token-Level Fact Correction in Abstractive Summarization

Given a source text X=(x1,…,xm) with m tokens and a generated summary Y′=(y′1,…,y′n) with n tokens (m≥n ), this paper aims to produce a fact-corrected summary Y using a token-level fact corrector. The proposed token-level fact corrector finds the inconsistent tokens in the generated summary and replaces them with the correct tokens in the source text, while ordinary fact correction models emend the generated summary at entity-level.

Assume that a data set D={(X,Y′,Y)} is given where Y is a fact-corrected summary. The proposed token-level fact corrector generates the best fact-corrected summary Y∗ from X and Y′ by maximizing a conditional probability of X and Y′ . That is,

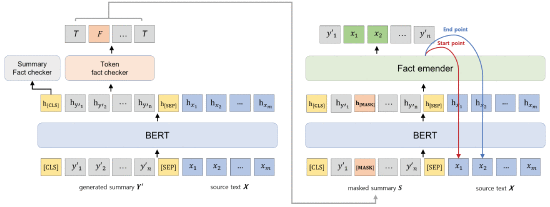
Y∗=argmaxY¯∈replace(Y′,X;θ)P(Y¯|X,Y′;θ),

View SourceRight-click on figure for MathML and additional features.

\begin{equation\*} Y^{\*} = \mathop {\mathrm {arg\,max}} \_{\bar {Y} \in replace(Y',X;\theta)} P(\bar {Y}|X, Y';\theta),\end{equation\*}

where replace(Y′,X;θ) is a set of text sequences obtained by replacing the tokens in Y′ with those in X , and θ is trainable parameters.

Figure 1 shows how the proposed token-level fact corrector works. The proposed fact corrector takes Y′ and X as its input and encodes them using an encoder. Then, a summary representation and token representations are fed to summary fact checker and token fact checker, respectively, to verify factually consistency. As a result of the token fact checker, inconsistent tokens in Y′ are masked out and the masked summary is updated to a factually-consistent summary by fact emender.

FIGURE 1. –

The overview of the proposed fact corrector for solving token-level fact correction in abstractive summarization.

FIGURE 1.

The overview of the proposed fact corrector for solving token-level fact correction in abstractive summarization.

Show All

The summary fact checker determines whether Y ’, a generated summary is factually consistent with X , a source text. The input to the summary fact checker is a concatenation of Y′ and X with special tokens of [CLS] and [SEP]. That is, the input is a sequence of ([CLS],Y′,[SEP],X) . Then, this sequence is fed into BERT [23] to obtain contextual representations: mathfonts

h[CLS],hY′,h[SEP],hX=BERT(([CLS],Y′,[SEP],X)),(1)

View SourceRight-click on figure for MathML and additional features.

\begin{equation\*} \mathbf {h}\_{\texttt {[CLS]}}, \mathbf {h}\_{Y'}, \mathbf {h}\_{\texttt {[SEP]}}, \mathbf {h}\_{X} = BERT((\texttt {[CLS]}, Y', \texttt {[SEP]}, X)), \tag{1}\end{equation\*}

where h[CLS] and h[SEP] are the contextual representations corresponding to [CLS] token and [SEP] token, respectively. hY′ and hX are the contextual token representations of the generated summary and the source text, respectively. Here, h[CLS] is used as the aggregate sequence representation for classification tasks [23]. The factual consistency of Y′ is then determined by the summary fact checker of which input is h[CLS] , and the summary fact checker is a binary classifier implemented as a one-layer MLP.

The token fact checker finds inconsistent tokens in a generated summary with respect to a source text. Thus, it takes hY′∈R|hidden|×|n| in Equation (1) as its input where hidden is the size of BERT hidden layer, and predicts inconsistent tokens in Y′ with another binary classifier. This binary classifier is also a one-layer MLP as in the summary fact checker. Note that hY′ is fed to the binary classifier so that all inconsistent tokens in Y′ are obtained independently and simultaneously.

The fact emender revises the inconsistent tokens found by the token fact checker by replacing them with the correct tokens in a source text. First, all inconsistent tokens in Y′ are masked out using a [MASK] token. If consecutive inconsistent tokens exist, they are merged into a single [MASK] token. Let S be a masked summary which has T [MASK] tokens. Note that T can be zero if there is no inconsistent tokens. The input to the fact emender is similar to the input of the summary fact checker and the token fact checker except that S is used instead of Y′ . Then, BERT is used again to obtain contextual token representations including [MASK] tokens.

After that, the fact emender predicts the start and the end indices of a correct text span in X for every [MASK] token in an auto-regressive fashion. Let h[MASK]t be a contextual representation for the t -th [MASK] token. For the t -th [MASK] token, hiX , the i -th token representation in X , is fused with h[MASK]t as

zti=[h[MASK]t;hiX],

View SourceRight-click on figure for MathML and additional features.

\begin{equation\*} \mathbf {z}\_{i}^{t} = [\mathbf {h}\_{\texttt {[MASK]}\_{t}};\mathbf {h}\_{X}^{i}],\end{equation\*}

where [;] denotes a vector concatenation. With this input token representation conditioned on the t -th [MASK] token, the start position (s) and the end position (e) for the mask token are predicted by

qsi=qei=pstarti=pendi=ReLU(w⊤szti+bs),ReLU(w⊤ezti+be),p(i=s)=exp(qsi)∑mj=1exp(qsj),p(i=e)=exp(qei)∑mj=1exp(qej).

View SourceRight-click on figure for MathML and additional features.

\begin{align\*} q^{s}\_{i}=&\mathbf {ReLU}(w\_{s}^{\top} \mathbf {z}\_{i}^{t} + b\_{s}), \\ q^{e}\_{i}=&\mathbf {ReLU}(w\_{e}^{\top} \mathbf {z}\_{i}^{t} + b\_{e}), \\ p^{start}\_{i}=&p(i=s) = \frac {\exp (q^{s}\_{i})}{\sum \_{j=1}^{m}\exp (q^{s}\_{j})}, \\ p^{end}\_{i}=&p(i=e) = \frac {\exp (q^{e}\_{i})}{\sum \_{j=1}^{m}\exp (q^{e}\_{j})}.\end{align\*}

Here, ws , we and bs , be are trainable parameters, and pstarti and pendi are the start and the end probabilities of i -th token in X , respectively. The span for the t -th [MASK] token is then selected based on argmax of all pstarti ’s and pendi ’s. That is,

p^start=p^end=argmax(pstart1,…,pstarti,…,pstartm),argmax(pend1,…,pend1,…,pendm),

View SourceRight-click on figure for MathML and additional features.

\begin{align\*} \hat {p}^{start}=&\mathop {\mathrm {arg\,max}} \big (p^{start}\_{1}, \ldots, p^{start}\_{i}, \ldots, p^{start}\_{m} \big), \\ \hat {p}^{end}=&\mathop {\mathrm {arg\,max}} \big (p^{end}\_{1}, \ldots, p^{end}\_{1}, \ldots, p^{end}\_{m} \big),\end{align\*}

with a constraint of p^start<p^end . Finally, the corrected summary is generated by replacing all [MASK] tokens with their predicted spans.

SECTION IV.Training Token-Level Fact Correction

A. Training Data Set Generation

There is no official data set for fact correction in abstractive summarization. Thus, this paper generates a training set for fact correction from a data set for abstractive summarization data set in a weakly-supervised way. Algorithm 1 shows how to generate the training data set for the token-level fact correction. That is, it explains how a inconsistent summary Y′ is generated from a source text X and a reference summary Y . Given a pair of X and Y in the abstractive summarization data set, it first selects a token span, TS , in Y which appears also in X . TS in X is used as an answer span for an inconsistent summary. Since it can appear multiple times in X , to specify the exact indices of TS in X , one sentence that is most similar to Y is chosen. Let X′ be a set of sentences in X which contains TS . Then, the most similar sentence s∗ is found by

s∗=argmaxsi∈X′ sim(si,Y),(2)

View SourceRight-click on figure for MathML and additional features.

\begin{equation\*} s^{\*} = \mathop {\mathrm {arg\,max}} \_{s\_{i} \in X'}~sim(s\_{i}, Y), \tag{2}\end{equation\*}

where sim(⋅,⋅) is a BERT-based sentence similarity function [23]. Then, the start and the end indices for the fact emender are obtained using index(⋅) , an index function that returns the indices for TS of s∗ in X . Finally, a factually-inconsistent summary Y′ is obtained by replacing TS in Y with another token span selected randomly from X .

Algorithm 1 Training Data Set Generation for Token-Level Fact Corrector

Abstractive summarization data set Dsummary

Training data set D with start and end indices

D=ϕ

for each (X,Y)∈Dsummary do

repeat

sample a,b∼Uniform{1,m}

// choose a token span as Y[a:a+b]

TS=Y[a:a+b]

until TS∈X

// X′ is a set of sentences in X that contains TS

X′=⟨s1,…,sp⟩,s.t.TS∈si

// find the most similar sentence s∗ using Eqn. (2)

s∗=argmaxsi∈X′sim(si,Y)

// get start index and end index of TS in s∗

start,end=index(s∗,X,TS)

// generate Y′i

Y′=Y

sample c,d∼Uniform{1,m}

Y′[a:a+b]=X[c:c+d]

D=D∪{(X,Y,Y′,(start,end))}

end for

Table 2 shows an example of the training data generation for token-level fact correction. From a source text X and a reference summary Y , a token span ‘by a single bullet’ in Y is first selected. After that, it finds a sentence s∗ in X that is most similar to Y . Then, the start and the end indices that are 129 and 132 respectively are obtained from X using s∗ . The final inconsistent summary is generated by replacing ‘by a single bullet’ in Y with the token span ‘for her manslaughter’ in X .

TABLE 2 Table 2- 
An Example of Training Data Generation. The Underlined Token Spans in Both the Source Text and They Reference Summary Represent the Selected Token Spans. The Doubly-Underlined Token Span is a Corrupted Token Span

An Example of Training Data Generation. The Underlined Token Spans in Both the Source Text and They Reference Summary Represent the Selected Token Spans. The Doubly-Underlined Token Span is a Corrupted Token Span

B. Joint Learning and Inference

The proposed fact corrector consists of three modules of the summary fact checker, the token fact checker, and the fact emender. The summary fact checker is trained to minimize the binary cross entropy loss between positive and negative summaries, and the token fact checker is to minimize the binary cross entropy loss between positive and negative tokens. That is, when σ(⋅) is the sigmoid function, the loss for the summary fact checker is

Lsfc=−1N∑iN[qi⋅logσ(ci)+(1−qi)⋅log(1−σ(ci))],

View SourceRight-click on figure for MathML and additional features.

\begin{equation\*} \mathcal {L}\_{sfc} = -\frac {1}{N}\sum \_{i}^{N}\big [q\_{i} \cdot \log \sigma (c\_{i}) + (1 - q\_{i}) \cdot \log (1 - \sigma (c\_{i})) \big],\end{equation\*}

where N is the number of training examples, qi is a ground-truth summary fact label for the i -th summary instance, ci is a model output for the summary instance. Similarly, the loss for the token fact checker is

Ltfc=−1N∑iN∑jni[tij⋅logσ(oij)+(1−tij)⋅log(1−σ(oij))],

View SourceRight-click on figure for MathML and additional features.

\begin{equation\*} \mathcal {L}\_{tfc} = -\frac {1}{N}\sum \_{i}^{N}\sum \_{j}^{n\_{i}}\big [t\_{ij} \cdot \log \sigma (o\_{ij}) + (1 - t\_{ij}) \cdot \log (1 - \sigma (o\_{ij})) \big],\end{equation\*}

where tij is a ground-truth label for the j -th token in the i -th summary instance, oij is a model output for the token, and ni is the number of sampled tokens from the i -th instance.

Since most tokens in a generated summary are consistent, this skewed class distribution makes it difficult for the token fact checker to be trained. To mitigate this problem, this paper adopts a balanced sampling to make a balance between the number of consistent tokens and that of inconsistent tokens. The objective of the fact emender is formulated as the negative log likelihood of the ground-truth start and end indices. That is, the negative log likelihood loss for the fact emender is

Lfe=−1N∑iN[log(pstartxistart)+log(pendxiend)],

**View SourceRight-click on figure for MathML and additional features**.

\begin{equation\*} \mathcal {L}\_{fe} = -\frac {1}{N}\sum \_{i}^{N}\big [\log (p^{start}\_{x^{i}\_{start}}) + \log (p^{end}\_{x^{i}\_{end}}) \big],\end{equation\*}

where xistart and xiend are the ground-truth start and end indices of the i -th instance, respectively.

Note that three modules in the proposed corrector are jointly trained with the three loss functions. Thus, the final loss for the proposed fact corrector is

L=w1Lsfc+w2Ltfc+w3Lfe,(3)

**View SourceRight-click on figure for MathML and additional features.**

\begin{equation\*} \mathcal {L} = w\_{1}\mathcal {L}\_{sfc} + w\_{2}\mathcal {L}\_{tfc} + w\_{3}\mathcal {L}\_{fe}, \tag{3}\end{equation\*}

where wi is a relative weight for the i -th loss.

When a new source text and its corresponding inconsistent summary are given, the proposed token-level fact corrector applies the summary fact checker, the token fact checker and the fact emender sequentially. Table 3 shows how the inconsistent summary changes to a consistent summary by the proposed token-level fact corrector. This example is derived from CNN/DM data set where the summary generated by BART is factually inconsistent with the source text. First of all, the summary fact checker determines that the given summary is negative, which implies that the summary is factually inconsistent. After that, the token fact checker determines whether each token in the summary is positive or negative. In this example, two token spans of ‘a Merseyside’ and ‘vomiting and diarrhoea’ are predicted negative so that they are replaced with [MASK] token. Then, the fact emender revises the masked summary by changing each [MASK] token to a proper token in the source text. That is, the first [MASK] is replaced with ‘Arrowe Pa’ and the second is with ‘vomiting, stomach cramps, fever and diarrhoea.’

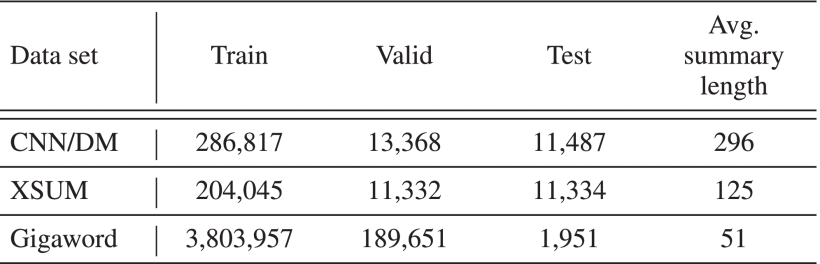
TABLE 3Table 3- 
An Example of the Inference Step Using the Proposed Token-Level Fact Correction

An Example of the Inference Step Using the Proposed Token-Level Fact Correction

SECTION V.Experiments

A. Experimental Settings

Experiments are conducted with three abstractive summarization benchmark data sets: CNN/DM [24], XSUM [25], and Gigaword [26]. CNN/DM contains over 300,000 news articles written by CNN and Daily Mail journalists. XSUM is a data set collected from BBC articles (2010 to 2017) that cover several domains including news, politics, sports, and so on. Gigaword is a data set designed to predict an entailment between an article and its headline. This data set can be used as a summarization data set by regarding an article as a source text and a headline as a reference summary, respectively. These three data sets are widely-used for evaluating abstractive summarization methods. Table 4 summarizes the simple statistics on these data sets.

TABLE 4

A Simple Statistics on the Data Sets Used for the Experiments

Note that every abstractive summarization data set contains the pairs of a source text and a reference summary. This paper uses a BART summarizer [8] to produce a system-generated summary from a source text, since the BART summarizer has shown superior performance in abstractive summarization. The BART summarizer is finetuned for each abstractive summarization data set.

The proposed fact corrector is mainly compared with two baselines: An entity corrector (Entity) [9] and a rewriting model (Rewrite) [13]. The entity corrector first selects only entities in a system-generated summary, and then these selected entities are iteratively replaced with the entities in a source text. The rewriting model generates a new correct summary from a system-generated summary and a source text through a sequence-to-sequence model. Since all tokens are regenerated, it is equivalent to an autoregressive fact corrector for the token level.

For the evaluation of the proposed corrector and its baselines, this paper follows the evaluation protocol proposed by Dong et al [9]. The corrected summary is evaluated from two perspectives: correctness and factual consistency. For correctness, the corrected summary is compared with the reference summary and the performance is measured with ROUGE. For factual consistency, three automatic factual inconsistency evaluation protocols of BERTScore [21], QAGS [27], and FactCC [14] are used.

This paper uses the BERT-large-cased for the encoder in Equation (1). That is, it has 24 layers (transformer blocks), 16 attention heads per layer, and 1,024 hidden dimensions. The data set to train the proposed fact corrector is generated from the training set of CNN/DM using Algorithm 1. Adam optimizer [28] with default settings is used to optimize the proposed corrector and the baselines, and the learning rate is set to 1e-6. The weights wi in Equation (3) are set as the same value of one. All experiments in this paper are done with one Telsa V100 GPU.

B. Experimental Results

1) Performance of Fact Correction

Table 5 presents the comparison of the proposed corrector with the baselines on CNN/DM, XSUM, and Gigaword data sets. BART in this table is a BART summarizer without any correction procedure. The ROUGE scores of BART are 40.18, 34.51, and 33.24 on CNN/DM, XSUM, and Gigaword, respectively, which are higher than the scores of the fact correction models including the proposed corrector. However, its performance for the factual consistency metrics is lower than correction models. This result implies that the summaries generated by BART are fluent but unfaithful.

TABLE 5

Table 5- 
Performance Measured With BERTScore, QAGS, FactCC and ROUGE on CNN/DM, XSUM, and Gigaword Data Sets Performance Measured With BERTScore, QAGS, FactCC and ROUGE on CNN/DM, XSUM, and Gigaword Data Sets

Table 5-

Performance Measured With BERTScore, QAGS, FactCC and ROUGE on CNN/DM, XSUM, and Gigaword Data Sets

Among the fact correction models, the proposed corrector outperforms the baselines for all data sets. It boosts up all factual consistency measures by a large margin. Especially the largest margin is gained on the Gigaword data set, where its BERTScore is 81.04 which is 1.03 higher than that of Entity, the entity corrector. Since the summaries in Gigaword is short and highly abstractive [26], inconsistencies frequently occur at the token level. Therefore, the proposed token-level correction is more suitable than the baselines especially for this data set. As a result, its performance is superior to the baselines in this data set for all factual consistency metrics. In addition, even if the proposed corrector achieves 0.95, 1.36, and 0.77 lower ROUGE scores than BART, its reduction rate is smaller than those of the baselines. These results show that the proposed corrector can improve the factual consistency of system-generated summaries without sacrificing correctness.

2) Ablation Study

The proposed token-level fact corrector sequentially applies the summary fact checker, the token fact checker, and the fact emender. Among these sub-modules, the summary fact check is somewhat independent from others in that it works prior to other modules. That is, if the token fact checker predicts all tokens of a given summary as positive (consistent), the summary can be regarded as a consistent summary.

Table 6 shows the results of an ablation study over XSUM data set. ‘– Summary fact checker’ indicates the exclusion of the summary fact checker from the proposed model. Without the summary fact checker, BERTScore and ROUGE-1 of the proposed fact corrector decrease 0.86 and 0.62, respectively. This is because some correct summaries are modified unnecessarily by the token fact checker and the fact emender. Furthermore, a BERT encoder is shared by all submodules, and thus it is jointly trained by them. As a result, the summary fact checker helps the performance of other modules be improved through the encoder.

TABLE 6 Ablation Study of the Proposed Fact Corrector on XSUM Data Set

Table 6- Table 6- 
Ablation Study of the Proposed Fact Corrector on XSUM Data Set

Ablation Study of the Proposed Fact Corrector on XSUM Data Set

3) Human Evaluation on K2019 Data Set

This paper conducts human evaluation with K2019 data set [14] for consistency checking and error correction. This data set is designed to show the factual consistency of abstractive summarization, and it consists of 503 triples of a source text, a system-generated summary, and a manually-annotated label about whether the summary is consistent or inconsistent. There are 441 consistent and 62 inconsistent summaries. The evaluation of the proposed fact corrector and the baselines for this data set is made by allowing the correction models to rectify the system-generated summaries regardless of the factual consistency of the summaries. Then, the numerical evaluation of each model is done by following the process of Cao et al [13]. That is, the correctness of the revised summaries by each model is judged by three human annotators. If all annotators agree that a revised summary is consistent, then the summary is determined to be consistent. Otherwise, the summary is inconsistent.

Table 7Table 7- 
Fact Correction Results on K2019 Data Set

summarizes the performance of the proposed corrector and the baselines on K2019 data set. ‘→ Consistent’ at the ‘Consistent’ column implies that a consistent system-generated summary still remains as a consistent summary after correction. On the other hand, ‘→ Inconsistent’ represents that a consistent summary becomes inconsistent after correction, which indicates that a correction model revises a given summary in a wrong way. Thus, the value of ‘→ Consistent’ should be maximized, and that of ‘→ Inconsistent’ should be minimized. Similarly, the value of ‘→ Consistent’ at the ‘Inconsistent’ column should be maximized, and that of ‘→ Inconsistent’ should be minimized.

TABLE 7 Fact Correction Results on K2019 Data Set

The proposed model achieves higher performance than the baselines even for this data set. Among the 441 consistent system-generated summaries, the proposed fact corrector wrongly rectifies only eight summaries. This is much less than the values of Rewrite and Entity. Furthermore, the proposed fact corrector converts inconsistent summaries into consistent summaries more accurately. The number of correctly converted summaries by the proposed corrector is 15, and this is 7% improvement over Rewrite. The improvement mainly comes from the facts that the proposed corrector is designed to perform token-level fact correction and it tries to revise only inconsistent tokens.

4) Summary Correction Results

Table 8 shows the examples of how a summary generated by the BART summarizer changes to a correct one by the proposed fact correcor and the baselines. The first example is derived from CNN/DM data set, where the BART summarizer generates an inconsistent summary which includes a wrong phrase “in the Scottish Cup semi-final,” where “in the Scottish Cup Final” is correct accoring to the source text. Since ‘semi-final’ is not an entity, Entity cannot revise the wrong phrase. The proposed corrector, however, correctly replaces ‘semi-final’ with ’final’ because it distinguishes the inconsistent tokens from consistent ones. The second example is also derived from CNN/DM data set. The summary generated by the BART summarizer includes an inconsistent token ‘GQ’. This token should be ‘Andy Jordan’ and Entity correctly edits the token to ‘Andy Jordan’ correctly. The proposed fact correct replaces this token with ’Chelsea star Andy Jordan,’ w which decreases the ROUGE score slightly even if the noun phrase is correct. The third example is obtained from K2019 data set. Similar to the first example, the given generated summary is factually inconsistent with respect to the source text, and the inconsistent token span is not an entity. As a result, Entity fails in revising this summary, but the proposed corrector rectifies it correctly.

TABLE 8Table 8- 
Examples of Generated and Fact Corrected Summaries. The Underline in the Summaries Implies an Inconsistent Token Span and the Bold Tokens are Consistent Ones

Examples of Generated and Fact Corrected Summaries. The Underline in the Summaries Implies an Inconsistent Token Span and the Bold Tokens are Consistent Ones

SECTION VI.Conclusion

This paper has proposed a token-level fact correction model in abstractive summarization. The proposed fact corrector consists of three modules: the summary fact checker, the token fact checker, and the fact emender. These modules are applied sequentially to a given inconsistent summary to generate a consistent summary. The sentence fact checker first checks whether the given summary is factually consistent with respect to the source text or not. Then, the token fact checker finds the factual inconsistencies at the token-level if the given summary is determined to be inconsistent. Finally, the fact emender corrects only the factually-inconsistent tokens by replacing them with the consistent tokens in the source text. These three modules are trained through a joint learning which shares an encoder to represent the summary and the source text. Since the proposed fact corrector rectifies an inconsistent summary at token-level, it becomes a general fact corrector in that it does not require any external tools such as a named entity recognizer.

The experimental results on three benchmark data sets have shown that the proposed token-level correction achieves higher performance than the previous fact correction models. Moreover, it is also shown that the proposed fact corrector generates more factually-consistent summaries even when evaluated on a manually-annotated data set. Through these experiments, the token-level fact correction is proven to be a practical and general approach to correcting inconsistent summaries in abstractive summarization.