Phrase-Level Localization of Inconsistency Errors in Summarization by Weak Supervision

Masato Takatsuka **Tetsunori Kobayashi** **Yoshihiko Hayashi**
Faculty of Science and Engineering, Waseda University
Waseda-machi 27, Shinjuku, Tokyo 1620042, Japan
takatsuka@pcl.cs.waseda.ac.jp
koba@waseda.jp yshk.hayashi@aoni.waseda.jp

Abstract

Although the fluency of automatically generated abstractive summaries has improved significantly with advanced methods, the inconsistency that remains in summarization is recognized as an issue to be addressed. In this study, we propose a methodology for localizing inconsistency errors in summarization. A synthetic dataset that contains a variety of factual errors likely to be produced by a common summarizer is created by applying sentence fusion, compression, and paraphrasing operations. In creating the dataset, we automatically label erroneous phrases and the dependency relations between them as "inconsistent," which can contribute to detecting errors more adequately than existing models that rely only on dependency arc-level labels. Subsequently, this synthetic dataset is employed as weak supervision to train a model called SumPhrase, which jointly localizes errors in a summary and their corresponding sentences in the source document. The empirical results demonstrate that our SumPhrase model can detect factual errors in summarization more effectively than existing weakly supervised methods owing to the phrase-level labeling. Moreover, the joint identification of error-corresponding original sentences is proven to be effective in improving error detection accuracy.

1 Introduction

The quality, particularly the fluency, of automatically generated abstractive summaries has improved significantly (Lewis et al., 2020; Zhang et al., 2020; Raffel et al., 2020) with methods that benefit from large-scale pre-trained language models. However, recent studies (Cao et al., 2018; Maynez et al., 2020) have pointed out that more than 30% of the generated summaries are inconsistent with the source documents owing to unintentionally introduced factual errors, which affect the reliability and usability of summarization systems.

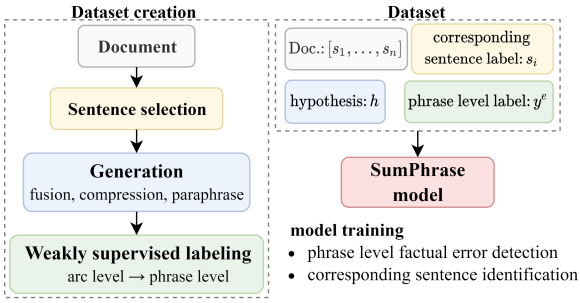

Figure 1: Overview of proposal. A synthetic dataset is created and used as weak supervision to train the SumPhrase error localization model.

Existing approaches for evaluating inconsistency in summarization can be roughly divided into two categories. One is the unsupervised approach that relies on an external natural language understanding (NLU) system, such as natural language inference (NLI) (Falke et al., 2019; Mishra et al., 2021; Laban et al., 2022) or question answering (QA) (Durmus et al., 2020; Wang et al., 2020; Scialom et al., 2021) systems, to validate a summary. Although this approach can benefit from the ever-progressing NLU technologies, the configuration and performance of an inconsistency detection system would be inevitably constrained by the underlying systems.

Therefore, we adopt another approach, namely the weakly supervised approach, which employs a synthetic dataset that contains automatically generated errors (Kryscinski et al., 2020; Goyal and Durrett, 2020). The key to the success of this approach lies in the quality and quantity of the synthetic dataset. It is necessary for such a dataset to contain a variety of errors that are likely to be produced by a common summarization system. However, as argued by Goyal and Durrett (2021), the error distributions that are produced by existing methods are considerably different from those produced by actual summarization models.

Proceedings of the 29th International Conference on Computational Linguistics, pages 6151–6164
October 12–17, 2022.

Figure 1 overviews our proposal for the localization of factual inconsistency errors in summarization. We propose to create a synthetic dataset by applying sentence fusion and compression operations in addition to paraphrasing. Thus, recurring summarization errors are expected to be reproduced. The resulting synthetic dataset is then used as weak supervision to train an error localization model called SumPhrase. This model jointly localizes errors in a summary and their corresponding sentences in the source document[1].

In the dataset creation process, a single sentence or a pair of sentences is first selected from a source document. A hypothesis sentence is then generated from the selected sentence or sentence pair, which is expected to contain various intrinsic factual errors. Thereafter, the hypothesis sentence is labeled at the phrase level, where each label is computed from dependency arc-level labels. Finally, the dataset for training the SumPhrase model is created by organizing the phrase-level labeled data.

Furthermore, the detection of factual errors in a summary can be combined with the identification of their corresponding sentences in the source document. This joint approach not only improves the usability of the error detection system but also improves the error detection performance in a multi-task learning setting (Kryscinski et al., 2020).

The contributions of this study are as follows:

- We present an improved method for generating a dataset by incorporating common means of summary generation, such as sentence fusion and compression, in addition to paraphrasing. We empirically investigate the effectiveness of these types of operations.
- We propose a model that jointly detects errors in a summary by fully using the phrase-level labels and identifies their corresponding sentences in the source document.
- We empirically demonstrate that the proposed method can localize the inconsistencies between a source document and the summary more effectively than existing weakly supervised methods.

2 Related Work

We review the approaches for inconsistency detection in summarization by classifying them into the following two categories.

2.1 Unsupervised Approach

We classify studies that exploit an external NLU system/model into this category.

A system that uses an NLI model determines that a summary is inconsistent if it cannot be entailed from the input document (Falke et al., 2019; Mishra et al., 2021). This method tends to suffer from a mismatch between the lengths of input texts: the documents to be summarized are generally longer than the usual premises collected in an NLI dataset, which makes inconsistency detection difficult (Mishra et al., 2021). Laban et al. (2022) addressed this issue by aggregating entailment scores measured between sentence pairs in the segments rendered from a source document. In contrast, a system that relies on a QA model considers a summary to be inconsistent if an external QA system fails to answer the questions related to a source document (Durmus et al., 2020; Wang et al., 2020; Scialom et al., 2021). The adequacy of a generated question, as well as the performance of the QA model, may affect the error detection performance.

2.2 Weakly Supervised Approach

We classify inconsistency detection systems that rely on supervised learning with artificially developed datasets into this category. In this approach, a dataset must reproduce the error distribution of a common summarizer as effectively as possible. Systems that use this method can be further classified according to how they generate errors for a summary.

Rule-based text transformation systems were developed to generate errors in (Kryscinski et al., 2020; Zhao et al., 2020; Zhang et al., 2021; Cao et al., 2020). For example, in FactCCX (Kryscinski et al., 2020), errors were generated by replacing an entity name and negating a sentence. Zeng et al. (2021) further enhanced the dataset with a data augmentation technique that applies an adversarial attack mechanism.

Generative models were employed in (Goyal and Durrett, 2021, 2020) to generate erroneous texts. In particular, Goyal and Durrett (2020) proposed using a sentence with a lower posterior probability as a potentially erroneous sentence. These generated sentences were then dependency-parsed, and the dependency arcs that were only found in the hypothesis sentence were annotated as erroneous.

[1]The dataset and code are available at https://github.com/taka2946/sumphrase

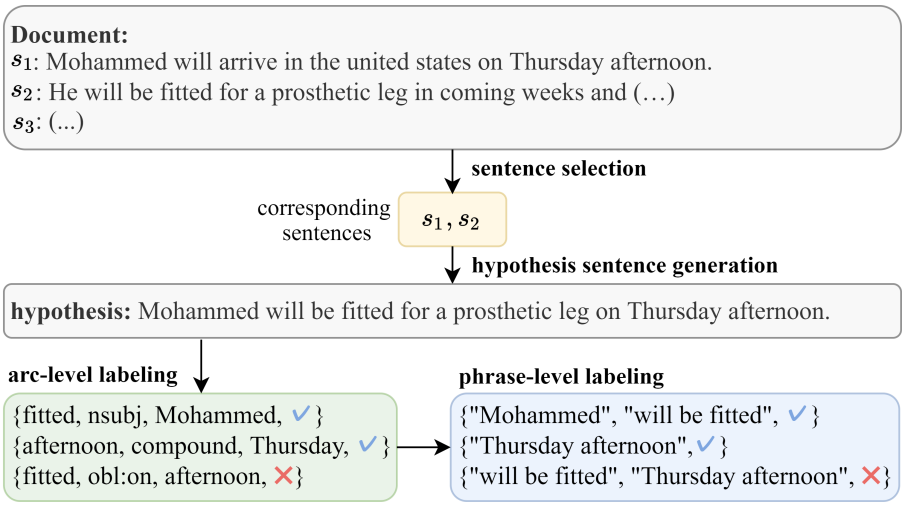

Figure 2: Process for dataset creation. In this example, two source sentences, s_1 and s_2, are selected, and the hypothesis sentence is generated by a sentence fusion operation. The selected sentences are considered error-originating if the hypothesis sentence poses an error. These source sentences are referred to as *corresponding sentences*.

We used the generative model approach, assuming that errors should be contained in the sentences generated with lower probabilities. However, this study differs from (Goyal and Durrett, 2021) in that we generate sentences not only by paraphrasing operations but also by means of sentence fusion and compression operations. This decision can be supported by the insights provided in (Lebanoff et al., 2019b) and (Lebanoff et al., 2019a). The former notes that most sentences in the reference summaries of the CNN/DailyMail dataset (Nallapati et al., 2016) are derived from sentence fusion and compression operations. The latter argues that sentence fusion tends to produce factual errors. Furthermore, we propose raising the error detection level from the dependency-arc level to the phrase level such that we can detect errors that would be overlooked by narrowly focused inspection with dependency-arc level labeling.

We jointly detect errors in summarization and localize their corresponding parts in the input document using multi-task learning, similar to the method presented in (Kryscinski et al., 2020). However, unlike their method, our method can localize more than one corresponding sentence, resolving the limitation of the method in (Kryscinski et al., 2020), which associates only single spans in the input document. This functionality is achieved by formulating the latter task as a multi-label classification problem such that an error in a summary sentence can be associated with multiple corresponding sentences in the source document.

3 Methodology

This section first details the process for creating a synthetic dataset that intentionally accommodates erroneous sentences. Our proposed model for jointly detecting errors in a summary and identifying corresponding sentences in the source document is then described.

3.1 Creation of Synthetic Dataset

3.1.1 Overview of the Creation Process

Figure 2 depicts the proposed process for creating a synthetic dataset that is employed as weak supervision. Sentence pairs ((s_1, s_2) in the example) are first **selected** from the document, and then each of them is **fused** into a sentence (indicated as *hypothesis*) by the generation model. The dependency arcs in the resulting hypothesis sentence are subsequently **labeled** as either "consistent" (blue), "inconsistent" (red), or unlabeled by comparing them with the arcs from the top-ranked hypothesis sentence, selected source sentences, and reference sentences. Thereafter, phrase-level labels (intra- or inter-phrases) are annotated by merging the arc-level labels. Although Figure 2 exemplifies the sentence fusion operation as a means of hypothesis-sentence generation, we also apply the sentence **compression** and **paraphrasing**. However, in the experiments, instead of actually applying sentence selection and paraphrasing, we used ready-made datasets. We performed the phrase-label annotation.

In the following, the dataset is represented as $\{D, h, (p_i, y_i^e)_{p_i \in h}, (p_n, p_m, y_{nm}^e)_{p_n, p_m \in h}, (s_j, y_j^s)_{s_j \in D}\}$, where D and h denote the input document and generated hypothesis sentence, respectively. Moreover, y_i^e denotes the intra-phrase consistency label for the i-th phrase p_i in h, whereas y_{nm}^e dictates the inter-phrase consistency label between p_n and p_m. Furthermore, y_j^s is a label that indicates whether the j-th sentence in D is a corresponding sentence of h.

3.1.2 Sentence Selection

One or two sentences in the input document $D = [s_1, s_2, ..., s_n]$ are selected and fed to each generative model to generate a hypothesis sentence, which is expected to contain factual errors. A pair of sentences is used for sentence fusion, whereas a single sentence undergoes compression and paraphrasing.

In the experiments, we used ready-made datasets instead of the actual sentence selection operation: the sentence fusion dataset (Lebanoff et al., 2020b) and sentence compression dataset (Desai et al., 2020). These datasets were created for the CNN/DailyMail dataset by pairing a reference sentence with the corresponding sentences in a source document that measured the highest ROUGE scores.

3.1.3 Sentence Fusion

We used the Transformer-based Trans-LINKING (Lebanoff et al., 2020a) model to generate a hypothesis sentence. The model employed in the experiments was pre-trained on the same sentence fusion dataset used for sentence selection. As in (Lebanoff et al., 2020a), we adopt only a hypothesis sentence that shared two or more words in both of the originating sentences. In the experiments, we generated hypothesis sentences with a beam size of 10. The most probable and improbable sentences were used in the labeling process.

3.1.4 Sentence Compression

We employed CUPS (Desai et al., 2020) for sentence compression. In the experiments, we used the pre-trained model provided by the authors of CUPS. In general, sentence compression removes redundant or unimportant portions from an input sentence; that is, factual errors rarely emerge. Therefore, we annotated the dependency arcs in the compressed sentence as consistent, provided that they also existed in the input to this model.

3.1.5 Paraphrasing

Any paraphrasing model can be employed for our purpose. In the experiments, we used the dataset made available by (Goyal and Durrett, 2021), which contains paraphrased versions of the selected reference sentences in the CNN/DailyMail dataset. We computed phrase-level labels by referring to the arc-level labels already provided in this dataset.

3.1.6 Labeling

Each generated hypothesis sentence was analyzed using the Stanford CoreNLP (Manning et al., 2014) dependency parser and subsequently underwent the labeling process. The labeling process first examines each of the dependency arcs and assigns either "consistent" or "inconsistent" labels or leaves them unlabeled. This arc-level annotation step is followed by the phrase-level annotation step that assigns labels to phrases and inter-phrase dependency relations. Note that our labeling process is inspired by the method proposed in (Goyal and Durrett, 2020), which assumes that the sentence with the lowest posterior probability may contain more factual errors than sentences with higher probabilities.

We denote an input sentence[2] to the generation model as x and the set of output sentences sorted by posterior probabilities as $H = [h_1, h_2, ..., h_k]$. Furthermore, $d(h)$ denotes a set of dependency arcs of a sentence h. A reference sentence is represented by h^*, and a_i denotes the i-th arc in the corresponding set of dependency arcs.

Arc-level labeling: We follow the method proposed in (Goyal and Durrett, 2020). We select sentence h_k with the lowest probability and assign a label to each arc a_i in h_k as follows.

$$y_i^a = \begin{cases} \text{"consistent"} & \text{if } a_i \in d(x) \cup d(h^*) \\ \text{"unlabeled"} & \text{if } a_i \in d(h_1) \setminus d(x) \cup d(h^*) \\ \text{"inconsistent"} & \text{otherwise} \end{cases} \quad (1)$$

Note that h_1, the sentence with the highest probability, is used as a presumably error-free sentence.

If a_i appears in the arc set of the corresponding original sentence x or that of the reference sentence h^*, we consider it consistent. If a_i appears in $d(h_1)$ but not in x or h^*, we do not assign any labels to the arc, as it is less reliable to determine it as

[2]Remember that the sentence x originates from the source document.

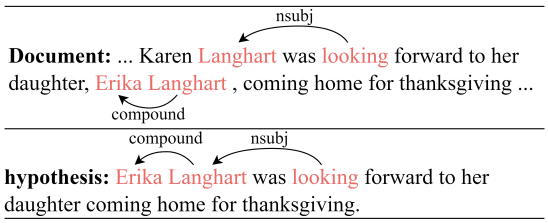

Figure 3: Motivating example for phrase-level labeling.

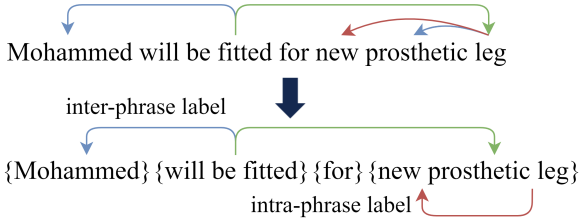

Figure 4: Example of phrase-level labels. The blue and red edges denote consistent and inconsistent labels, respectively, whereas the green edge indicates that the inter-phrase relation is unlabeled.

error-free. As arc a_i not matching either of these conditions may represent a factual error, we assign an inconsistent label to the arc.

We remind the reader that for a compressed sentence, we assign the labels as follows.

$$y_i^a = \begin{cases} \text{“consistent”} & \text{if } a_i \in d(x) \\ \text{“unlabeled”} & \text{otherwise} \end{cases} \quad (2)$$

Note that we did not assign the “inconsistent” label because a compressed sentence usually does not pose any factual error.

As in (Goyal and Durrett, 2020), we exclude particular dependency labels, such as `det` and `case`, from the annotation process, as these do not carry significant meanings.

Phrase-level labeling: Before describing the phrase-level labeling process, we explain its motivation using Figure 3. In this example, the hypothesis sentence contains a factual error because the agent of the verb “looking forward” is “Karen Langhart” and not “Erika Langhart.” This error may be difficult to localize with a model that only considers arc-level labels. Therefore, we assign intra-phrase labels as well as inter-phrase labels, as illustrated in Figure 2, such that factual errors beyond the word-to-word dependency relations can be detected. To initiate the phrase-level labeling process[3], we first recognize phrases by applying a set of heuristic rules that investigate the dependency structure of a sentence. This procedure recognizes a sequence of “phrases” covering the input token sequence without overlaps, where each phrase principally represents a base syntactic phrase.

Figure 4 depicts the intra-and inter-phrase labels, which are computed as follows. If all the dependency arcs that are closed in a phrase are marked as consistent, the intra-phrase label of the phrase is “consistent;” if any is marked as inconsistent, the label is “inconsistent.” An inter-phrase label is computed similarly, but by looking at each of the dependency arcs connecting a word in the head phrase and that in the dependent phrase. Note that the inter-phrase label is “inconsistent” if either of the connected phrases is marked inconsistent. In this example, the phrase “new prosthetic leg” is labeled as “inconsistent” because the particular dependency arc between “new” and “leg” is labeled as “inconsistent.”

3.2 SumPhrase: Proposed Model

Figure 5 illustrates the network architecture of the proposed SumPhrase model. The model consists of three parts: the detection of intra-phrase errors (blue), the detection of inter-phrase errors (red), and the localization of the corresponding sentences in the source document (green).

Training: The inputs to the model are twofold: the source document D and hypothesis sentence h. Tokenized tokens from each input are concatenated using the [CLS] token as the separator and fed to the pre-trained encoder, which enables a contextualized representation for each token along with the special [CLS] token that is expected to represent the hypothesis sentence. Thereafter, the representation of each sentence in D and each phrase p_i is generated using the span attention mechanism (Lee et al., 2017)[4]. The intra- and inter-phrase detection parts are trained by referring to the intra-phrase labels (y_i^e) and inter-phrase labels (y_{ij}^e). The probability of a phrase p_i being consistent is computed as follows, using its representation h_i^p:

$$p(y_i^e|p_i) = \text{softmax}(\text{FFN}_{intra}(h_i^p)). \quad (3)$$

[3]Refer to Appendix A for details of the phrase-level labeling process.

[4]The effectiveness of the span attention mechanism was experimentally confirmed, as described in Appendix B.

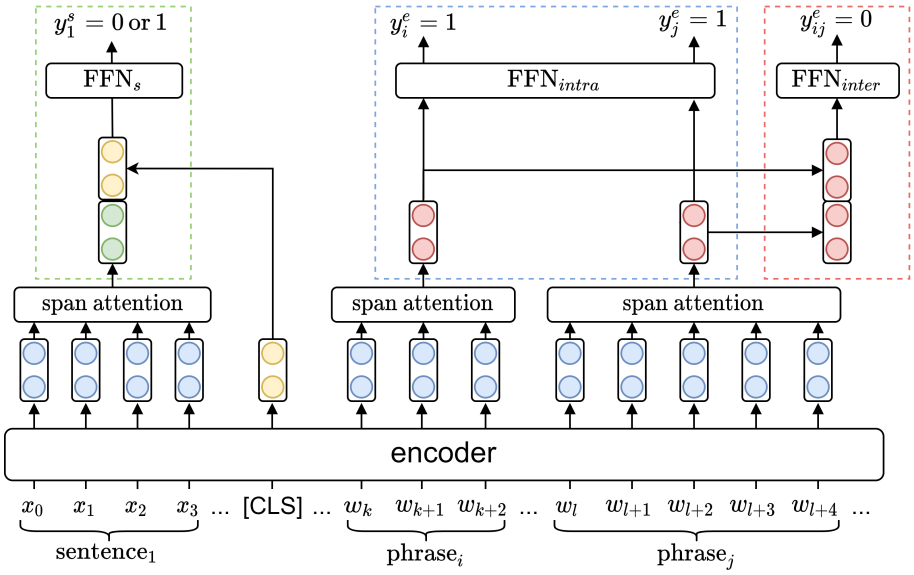

Figure 5: SumPhrase model. The blue and red frames display the intra-phrase and inter-phrase detection parts, respectively, and the green frame indicates the corresponding sentence localization part.

Similarly, the inter-phrase consistency probability is computed as follows:

$$p(y_{ij}^e|p_i, p_j) = \text{softmax}(\text{FFN}_{inter}([h_i^p; h_j^p])). \quad (4)$$

The probability of sentence s_i being a corresponding sentence of the hypothesis sentence is calculated as follows, where h_i^s and h^{cls} denote the representation of s_i and CLS token, respectively.

$$p(y_i^s|s_i) = \text{softmax}(\text{FFN}_s([h_i^s; h^{cls}])) \quad (5)$$

These logits are subsequently converted into losses (Loss_{intra}, Loss_{inter}, and Loss_s) using binary cross-entropy loss. Finally, the entire loss is defined as follows by incorporating a hyperparameter α that adjusts the impact of the corresponding sentence localization (Loss_s), which is, in a sense, introduced as an auxiliary task in the multi-task learning setting.

$$\text{Loss} = \text{Loss}_{intra} + \text{Loss}_{inter} + \alpha * \text{Loss}_s \quad (6)$$

Inference: The source document D and each sentence h in the target summary are fed to the model during the inference time. Note that each sentence in a target summary is dependency-parsed in advance. The intra- and inter-phrase consistencies are assessed using the model for phrases and phrase pairs that were extracted from the summary sentence. We flag a summary sentence as consistent only if it is predicted to exhibit no errors at any level. The corresponding sentences for a hypothesis sentence were identified using multi-label classification. Consequently, the corresponding sentences of a factual error are localized.

Source	# of data
Paraphrasing (para)	46,925
Sentence fusion (fusion)	72,093
Sentence compression (comp)	47,296
Reference sentences (ref)	107,278

Table 1: Statistics of created synthetic dataset.

4 Experimental Setup

The CNN/DailyMail dataset (Nallapati et al., 2016) was used in the experiments.

4.1 Training Dataset

We created a synthetic dataset to train the SumPhrase model based on the methodology described in the previous section. Table 1 lists the number of data instances classified according to the provenance of the data. Note that "paraphrasing" refers to paraphrased data provided by (Goyal and Durrett, 2021), and "Reference sentences" means reference summary sentences obtained from the CNN/DailyMail dataset, which were added to increase the phrases labeled as consistent.

In total, this dataset contains 2,021,592 consistent labels and 191,553 inconsistency labels. More-

Model	Training data	K2020 (BA)	K2020 (F1)	Reranking (% correct)
[Sentence level]				
FactCC	text transformations	72.7	0.706	70.0%
FactCCX	text transformations	72.9	0.711	-
SumFC	text transformations	80.4	-	78.7%
FactAdv	text transformations	73.3	0.701	-
[Arc level]				
Electra-DAE	text transformations	76.7	-	-
Electra-DAE	para	72.1	-	-
Electra-DAE (ours)	para+fusion+comp+ref	82.7	0.754	85.9%
[Phrase level (ours)]				
SumPhrase	para+fusion+comp+ref	**85.3**	**0.765**	**86.0%**
SumPhrase (-multi)	para+fusion+comp+ref	85.2	0.759	84.7%

Table 2: Results of error detection. FactCC, FactCCX (Kryscinski et al., 2020), SumFC (Zhang et al., 2021), and FactAdv (Zeng et al., 2021) employ sentence-level labeling using rule-based text transformations. The Electra-DAE models (Goyal and Durrett, 2021) adopt dependency arc-level labeling. Our SumPhrase models uses phrase-level labeling, where "-multi" dictates the model trained without the auxiliary task of corresponding sentence localization. Refer to Table 1 for para, fusion, comp, and ref in the Training data column.

over, 186,028 source sentences were marked as one of the corresponding sentences of a hypothesis sentence, whereas 3,389,275 other sentences were not labeled as corresponding sentences.

4.2 Training the SumPhrase Model

The Electra-base model (Clark et al., 2020) pretrained on 3.3 billion tokens from Wikipedia and BooksCorpus (the same as the BERT-base) was employed as the encoder of the SumPhrase model in the experiments. We trained the SumPhrase model under the following conditions: number of epochs: 3, batch size: 10, optimizer: AdamW, and initial learning rate: 3e-5. The α parameter in equation-(6) was set to 0.5. We saved the checkpoint that achieved the highest BA on the validation portion of the K2020 dataset. Appendix C provides additional details of the experimental setup.

4.3 Test Datasets and Evaluation Metrics

We used two human-annotated datasets in the evaluation, which are known as K2020 (Kryscinski et al., 2020) and Reranking (Falke et al., 2019).

The K2020 dataset contains 441 consistent sentences and 62 inconsistent sentences. We measured the balanced accuracy (BA) and Macro-F1 for comparisons.

The Reranking dataset includes 373 instances, each containing a source document and two similar sentences: one is a consistent summary sentence extracted by an extractive summarizer, and the other is a potentially inconsistent summary sentence generated by an abstractive summarizer. That is, the reranking task is to correctly (re)rank consistent summary sentences higher. Thus, the portion of correctly ranked data instances is used as the evaluation metric. We also note that the Reranking dataset was used to measure the accuracy of the corresponding sentence localization task that was incorporated as an auxiliary task.

4.4 Compared Existing Models

We primarily compared our phrase-level models with dependency-arc-level Electra DAE models (Goyal and Durrett, 2021), both trained with the proposed dataset. In addition, we compared these models with existing sentence-level models including FactCC/FactCCX (Kryscinski et al., 2020), SumFC (Zhang et al., 2021), and FactAdv (Zeng et al., 2021). These models rely on text-transforming operations to generate erroneous sentences. Among these models, SumFC achieved the best results on both K2020 and Reranking[5].

5 Results and Discussion

This section discusses the efficacy of the proposed method by referring to the results. Appendix D presents and discusses some detection results.

[5]The performance of this model could be further improved by training it with our proposed dataset.

5.1 Detection of Sentence Inconsistency

Table 2 lists the results of inconsistency detection, where existing weakly supervised models that rely on a synthetic dataset are compared.

These results demonstrate that our SumPhrase model outperformed the other models in all metrics. The insights obtained from the results can be summarized as follows:

- The automatically generated training data significantly improved the detection performance (from comparisons of the Electra-DAE (ours) and SumPhrase models with other models).
- The phrase-level labeling was effective (from comparisons of the SumPhrase models with the Electra-DAE models).
- The multi-task learning strategy contributed to improving the performances (from comparisons of the SumPhrase model with the SumPhrase (-multi) model). In principle, the incrrectly localized corresponding sentences could affect the inconsistency detection performance[6]. However, the auxiliary task was accomplished with high accuracy, as shown in Table 3, and the multi-task learning strategy is assessed as adequate.
- Even without multi-task learning, the SumPhrase model exhibited better performances than most models. The exception was the result on Reranking, which was slightly inferior to that of Electra-DAE (ours).

5.2 Localization of Corresponding Sentences

We incorporated this task as an auxiliary task, assuming it would improve the performance of the main task. As shown in Table 2, this assumption is confirmed.

Table 3 summarizes the results of this auxiliary task, where BA and Macro F1 are used as the evaluation metrics. The table shows that our SumPhrase model achieved better results than the FactCCX (Kryscinski et al., 2020) and SumFC (tf-idf) (Zhang et al., 2021) models. The FactCCX model identifies evident spans in a document for a summary sentence, readily enabling the localization of the corresponding sentences. For the SumFC (tf-idf) model, we selected the original sentence that was most similar to a summary sentence by measuring the cosine similarity of the tf-idf vectors.

[6]See the second example presented in Appendix D, where extrinsic hallucination errors arise.

Model	BA	F1
FactCCX	95.9	0.945
SumFC (tf-idf)	97.0	0.970
SumPhrase	**99.2**	**0.980**

Table 3: Results of corresponding sentence localization.

	K2020		Reranking
Training data	BA	F1	%Correct
fusion	81.1	0.663	83.4%
comp	50.0	0.467	54.9%
para	73.4	0.708	78.4%
fusion+comp	82.5	0.687	84.3%
fusion+para	82.8	0.681	84.4%
fusion+comp+para	83.7	0.700	**85.1%**
ALL	**85.2**	**0.759**	84.7%

Table 4: Results of dataset ablation. "ALL" indicates the "fusion+comp+para+ref" condition.

An intriguing insight appears when we compare the SumFC results in Tables 2 and 3. Although the SumFC model achieved almost a level of accuracy (97.0 to 99.2 in BA) in this experiment close to that of our SumPhrase model, there was a significant gap in the inconsistency detection accuracy (78.7% to 86.0% with Reranking). This result indicates that the task of inconsistency detection requires a greater variety of errors in the training dataset, which cannot be achieved with the simple rule-based text transformation approach taken by SumFC and FactCCX, again demonstrating the superiority of our synthetic dataset.

5.3 Data Ablation

Table 4 presents the main results of the dataset ablation study conducted to investigate the contribution of each portion of the dataset or their combinations. Table 7 (Appendix E) provides further details of the ablation results. The SumPhrase model used in these experiments was trained with a single-task learning regime, that is, the SumPhrase (-multi) model.

The first block of Table 4 compares the results with the individual data, showing that sentence fusion data is the most promising for achieving good performances. Even with this dataset portion, our SumPhrase model achieved better results than the other models on K2020 (BA) and Reranking. In

contrast, sentence compression data alone is almost useless, as it contains only consistent labels and rarely introduces factual errors[7].

Based on the effectiveness of the sentence fusion data, the second block of the table presents the results obtained by incrementally adding other dataset portions. Apart from the performance on Reranking, the initial performance with sentence fusion data constantly improved with the addition of data, thereby demonstrating that increasing the variety of data in training is vital.

Also note that the accuracy of K2020 significantly improved by adding reference summary sentences (ALL). This result suggests that the addition of phrases labeled as "consistent" is effective. However , a careful reader may, notice that the ALL model performance on Reranking was slightly inferior to that of the fusion+comp+para model. Although a detailed investigation is required, this could be attributed to the quality of the Reranking dataset. In particular, positive and negative examples required in Reranking were generated using the FAS summarization model (Chen and Bansal, 2018). We suspect that the quality of these examples is not high compared with those generated by humans or SOTA models. Therefore, the ALL setting, which additionally incorporates human-generated reference summaries in the training data, did not yield a better result than the fusion+comp+para setting.

5.4 Limitations of the Present Work

Although the proposed method exhibits excellent performance, it focuses on intrinsic factual errors. Extrinsic hallucination errors, readily introduced with a dataset such as XSum (Narayan et al., 2018; Maynez et al., 2020), are another acute issue to be addressed. A manner of detecting extrinsic hallucination errors is to train a model using generated examples, as proposed by Utama et al. (2022). An alternative would be to incorporate another source of information, such as world knowledge, to validate the content of a summary. Before exploring this method, however, we would identify issues inherent to our approach using the XSum dataset.

6 Conclusions

We propose a neural model called SumPhrase for detecting errors in an abstractive summary. The model was empirically proven to be effective as it outperformed other weakly supervised models. This significant performance is primarily attributed to the dataset on which the model was trained. We created a synthetic dataset that contains factual errors likely to be produced by a common summarizer. These errors are labeled at the phrase level, as opposed to the dependency arc-level labels employed by existing models. The synthetic training dataset can also contribute to improving models that rely on an external NLI or QA system (Laban et al., 2022; Fabbri et al., 2021). It can also be used to fine-tune these models or to post-edit errors (Cao et al., 2020).

The model exhibited improved performance when jointly trained with the sub-task of localizing corresponding sentences in a summary sentence. This functionality may contribute to enhancing the explainability of an inconsistency error-detection system. Our method can also contribute to generating negative samples required to train a summarization model that relies on contrastive learning (Cao and Wang, 2021).

[7]Although not present in the table, the ref data exhibited almost the same results for the same reason.

Acknowledgements

This work was partially supported by JSPS KAKENHI, Grant Number 22K12723.

References

Meng Cao, Yue Dong, Jiapeng Wu, and Jackie Chi Kit Cheung. 2020. Factual error correction for abstractive summarization models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6251–6258, Online. Association for Computational Linguistics.

Shuyang Cao and Lu Wang. 2021. CLIFF: Contrastive learning for improving faithfulness and factuality in abstractive summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6633–6649, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *AAAI Conference on Artificial Intelligence*.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686, Melbourne, Australia. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Shrey Desai, Jiacheng Xu, and Greg Durrett. 2020. Compressive summarization with plausibility and salience modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6259–6274, Online. Association for Computational Linguistics.

Esin Durmus, He He, and Mona Diab. 2020. FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5055–5070, Online. Association for Computational Linguistics.

Alexander R Fabbri, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. 2021. Qafacteval: Improved qa-based factual consistency evaluation for summarization. *arXiv preprint arXiv:2112.08542*.

Tobias Falke, Leonardo F. R. Ribeiro, Prasetya Ajie Utama, Ido Dagan, and Iryna Gurevych. 2019. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2214–2220, Florence, Italy. Association for Computational Linguistics.

Tanya Goyal and Greg Durrett. 2020. Evaluating factuality in generation with dependency-level entailment. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3592–3603, Online. Association for Computational Linguistics.

Tanya Goyal and Greg Durrett. 2021. Annotating and modeling fine-grained factuality in summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1449–1462, Online. Association for Computational Linguistics.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, Online. Association for Computational Linguistics.

Philippe Laban, Tobias Schnabel, Paul N. Bennett, and Marti A. Hearst. 2022. SummaC: Re-visiting NLI-based models for inconsistency detection in summarization. *Transactions of the Association for Computational Linguistics*, 10:163–177.

Logan Lebanoff, Franck Dernoncourt, Doo Soon Kim, Lidan Wang, Walter Chang, and Fei Liu. 2020a. Learning to fuse sentences with transformers for summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4136–4142, Online. Association for Computational Linguistics.

Logan Lebanoff, John Muchovej, Franck Dernoncourt, Doo Soon Kim, Seokhwan Kim, Walter Chang, and Fei Liu. 2019a. Analyzing sentence fusion in abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 104–110, Hong Kong, China. Association for Computational Linguistics.

Logan Lebanoff, John Muchovej, Franck Dernoncourt, Doo Soon Kim, Lidan Wang, Walter Chang, and Fei Liu. 2020b. Understanding points of correspondence between sentences for abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 191–198, Online. Association for Computational Linguistics.

Logan Lebanoff, Kaiqiang Song, Franck Dernoncourt, Doo Soon Kim, Seokhwan Kim, Walter Chang, and Fei Liu. 2019b. Scoring sentence singletons and pairs for abstractive summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2175–2189, Florence, Italy. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Anshuman Mishra, Dhruvesh Patel, Aparna Vijayakumar, Xiang Lorraine Li, Pavan Kapanipathi, and Kartik Talamadupula. 2021. Looking beyond sentence-

level natural language inference for question answering and text summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1322–1336, Online. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gu'lçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, Jacopo Staiano, Alex Wang, and Patrick Gallinari. 2021. QuestEval: Summarization asks for fact-based evaluation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6594–6604, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Prasetya Utama, Joshua Bambrick, Nafise Moosavi, and Iryna Gurevych. 2022. Falsesum: Generating document-level NLI examples for recognizing factual inconsistency in summarization. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2763–2776, Seattle, United States. Association for Computational Linguistics.

Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020. Asking and answering questions to evaluate the factual consistency of summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5008–5020, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhiyuan Zeng, Jiaze Chen, Weiran Xu, and Lei Li. 2021. Gradient-based adversarial factual consistency evaluation for abstractive summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4102–4108, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Sen Zhang, Jianwei Niu, and Chuyuan Wei. 2021. Fine-grained factual consistency assessment for abstractive summarization models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 107–116, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zheng Zhao, Shay B. Cohen, and Bonnie Webber. 2020. Reducing quantity hallucinations in abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2237–2249, Online. Association for Computational Linguistics.

A Details of Phrase-Level Labeling

The phrase-level labels in a hypothesis sentence $h = \{w_1, ..., w_H\}$ are automatically annotated using the algorithm detailed in Algorithm 1. The first half of the algorithm creates "phrases" by inspecting particular dependency labels $L = \{l^a_1, ..., l^a_K\}$, as shown in Table 5. The second half of the algorithm assigns intra- and inter-phrase labels to each created phrase.

The inputs to the algorithm are a set of dependency labels $Y^a = \{a_1, ..., a_N\}$ and $a_n = \{w_i, w_j, l^a_n, y^a_n\}$. In this case, l^a_n denotes the dependency relation between words w_i and w_j, and y^a_n represents the corresponding dependency arc-level label assigned by the method proposed by (Goyal and Durrett, 2020).

advmod, amod, aux, compound
det, fixed, flat, goeswith, nummod
reparandum, nmod:poss, nmod:tmod

Table 5: Dependency labels used in phrase creation.

B Effectiveness of Span Attention Mechanism

Table 6 compares the performances of the SumPhrase models with and without the span attention mechanism (Lee et al., 2017) on the K2020 dataset. We used the averaged token vectors to represent phrases for the "without" condition. Given this result, we preferred the span attention mechanism over conventional average pooling, because it better captures the meaning of a semantically significant word.

Model	BA	F1
SumPhrase	**85.2**	**0.759**
SumPhrase (average)	83.8	0.746

Table 6: Effectiveness of span attention.

C Details of Experimental Setup

We used Stanford CoreNLP[8] (Manning et al., 2014) with "EnhancedDependenciesAnnotation" to dependency-parse hypothesis sentences. As the encoder for the SumPhrase model, we employed the electra-base-discriminator[9], implemented in the Huggingface Transformers library (Wolf et al., 2020). All experiments were conducted using an NVIDIA GeForce GTX TITAN X GPU with 12 GB of memory. The time required to train the SumPhrase model was approximately 40 h. For reliability, each number reported from our implementations in the present study is the average of three runs with different random seeds.

D Case Study

Figure 6 shows the results of the inconsistency detection using SumPhrase and FactCCX.

Document #1 shows a case in which the SumPhrase model correctly detects an inconsistent sentence and localizes the corresponding sentences. FactCCX failed to detect the inconsistency in the summary, although it identifyied the corresponding spans in the input. This result demonstrates that inconsistency detection is generally more complicated than sentence alignment.

Document #2 presents a typical case in which even the SumPhrase model cannot detect the inconsistency in a summary. This example poses the issue of extrinsic hallucination errors, which are difficult to detect using only the proposed methodology.

E More on Data Ablation

Table 7 lists the results for other data combinations. Again, the effectiveness of sentence fusion data compared with paraphrasing data is shown. Note that adding reference data to fusion or paraphrasing dataset portions degrades the BA, highlighting the efficacy of the ALL (fusion+comp+para+ref) combination.

	K2020		Reranking
Training data	BA	F1	%Correct
fusion+comp	82.5	0.687	84.3%
fusion+ref	79.5	0.716	**85.2%**
para+comp	69.3	0.702	80.3%
para+ref	65.5	0.691	77.7%
ALL	**85.2**	**0.759**	84.7%

Table 7: Additional results of dataset ablation.

[8] https://nlp.stanford.edu/software/stanford-corenlp-4.1.0.zip

[9] https://huggingface.co/google/electra-base-discriminator

Algorithm 1 Phrase-level labeling

Require: (1) The set of dependency arcs and dependency arc-level labels $Y^a = \{a_1, ..., a_N\}$, $a_n = \{w_i, w_j, l_n^a, y_n^a\}$ from a hypothesis sentence h; (2) a set of dependency labels $L = \{l_1^a, ..., l_K^a\}$.
Ensure: A set of phrase-level labels Y^e.

▷ create phrases
$P \leftarrow \{\{w_1\}, ..., \{w_H\}\}$
for $a_i \in Y^a$ **do**
  if $l_i^a \in L$ **then**
    $p_n, p_m \leftarrow$ GET_PHRASE(P, a_i)
    if $p_n \neq p_m$ **then**
      $P \leftarrow$ MERGE_PHRASES(P, p_n, p_m)
    end if
  end if
end for
▷ assign phrase-level labels by merging arc-level labels
$Y^e \leftarrow \emptyset$
for $a_i \in Y^a$ **do**
  $p_n, p_m \leftarrow$ GET_PHRASE(P, a_i)
  if $p_n == p_m$ **then**
    $y_n^e \leftarrow$ GET_LABEL$(Y^e, \{p_n\})$
    $y_n^e \leftarrow$ UPDATE_LABEL(y_n^e, y_i^a)
    $Y^e \leftarrow$ UPDATE$(Y^e, \{\{p_n\}, y_n^e\})$
  else
    $y_{nm}^e \leftarrow$ GET_LABEL$(Y^e, \{p_n, p_m\})$
    $y_{nm}^e \leftarrow$ UPDATE_LABEL(y_{nm}^e, y_i^a)
    $Y^e \leftarrow$ UPDATE$(Y^e, \{\{p_n, p_m\}, y_{nm}^e\})$
  end if
end for
for $\{p, y^e\} \in Y^e$ **do**
  if LEN(p) == 2 **then**
    $y_i^e \leftarrow$ GET_LABEL$(Y^e, \{p[0]\})$
    $y_j^e \leftarrow$ GET_LABEL$(Y^e, \{p[1]\})$
    if $y_i^e ==$ inconsistent or $y_j^e ==$ inconsistent **then**
      $Y^e \leftarrow$ UPDATE$(Y^e, \{p,$ inconsistent$\})$
    end if
  end if
end for

Creating phrases: The algorithm initially assumes that each word in h individually forms a phrase and passes through the elements a_i in Y^a. If the dependency relation of arc a_i is an element presented in L, phrases p_n and p_m are retrieved by GET_PHRASE for the words in this dependency relation. If p_n differs from p_m, these phrases are merged using MERGE_PHRASES.

Assigning phrase-level labels: The algorithm passes through elements a_i in Y^a to accumulate the phrase-level labels in Y^e. First, the phrases of the words in the corresponding dependency relation, that is, p_n and p_m, are retrieved by GET_PHRASE. If these phrases denote identical phrases, an intra-phrase label y_n^e is assigned to p_n. Otherwise, the inter-phrase label y_{nm}^e is assigned to the pair p_n and p_m. Y^e is updated accordingly. Finally, the inter-phrase label is "inconsistent" if either of the connected phrases is markedly inconsistent.

The GET _LABEL, UPDATE _LABEL, and UPDATE procedures employed in the algorithm are summarized as follows:

GET_LABEL: This procedure returns the intra- or inter-phrase label from Y^e, depending on the second argument. None of the labels is returned if the corresponding label is not in Y^e.

UPDATE_LABEL: This updates and returns the label of p_n in Y^e with y_i^a if y_i^a is ranked higher than p_n in the designated priority order: inconsistent $>$ unlabeled $>$ consistent $>$ None.

UPDATE: This procedure updates the set of phrase-level labels y^e with the result of UPDATE_LABEL.

Document #1: Garissa, Kenya (CNN) Kenyan police have arrested five suspects in connection with Thursday's attack at Garissa university college (…) Thursday's attack by Al-Shabaab militants killed 147 people, including 142 students, three security officers and two university security personnel. The attack left 104 people injured, including 19 who are in critical condition, Nkaissery said. During search and recovery efforts on Friday, CNN witnessed one male who was not a student hiding under a bed. (…)
Summary: 147 people, including 142 students, are in critical condition.
FactCCX: consistent
SumPhrase: inconsistent
Human: inconsistent

Document #2: (CNN) the question : how can i know if my food is safe to eat after a specific product recall ? The answer: many of us shed a few tears over the recent sabra hummus recall (even though we are perfectly capable of making our own at home), but that sadness quickly transformed into anxiety when we looked inside our refrigerators and saw the potentially tainted culprit sitting there on the shelf. To assuage any fears, we asked John Swartzberg, m.d., a clinical professor at the university of California at Berkeley, to walk us through the process of determining if our favorite dip was still safe to eat. Related : Amy's kitchen recalls more than 70 , 000 cases of food due to fear of listeria contamination.
Summary: fda recommends anyone who has consumed a listeria-laden food should let their physician know.
FactCCX: consistent
SumPhrase: consistent
Human: inconsistent

Figure 6: Comparison of FactCCX and SumPhrase outputs. The sentence with a blue underline was identified as an error-corresponding sentence by SumPhrase, whereas the span with a red underline was localized by FactCCX. The dependency relation between the red phrases was determined as erroneous by SumPhrase.