SummaC: Re-Visiting NLI-based Models for Inconsistency Detection in Summarization

Philippe Laban UC Berkeley, USA **Tobias Schnabel** Microsoft, USA **Paul N. Bennett** Microsoft, USA **Marti A. Hearst** UC Berkeley, USA*

Abstract

In the summarization domain, a key requirement for summaries is to be factually consistent with the input document. Previous work has found that natural language inference (NLI) models do not perform competitively when applied to inconsistency detection. In this work, we revisit the use of NLI for inconsistency detection, finding that past work suffered from a mismatch in input granularity between NLI datasets (sentence-level), and inconsistency detection (document level). We provide a highly effective and light-weight method called SummaC$_{\text{Conv}}$ that enables NLI models to be successfully used for this task by segmenting documents into sentence units and aggregating scores between pairs of sentences. We furthermore introduce a new benchmark called SummaC (**Summa**ry **C**onsistency) which consists of six large inconsistency detection datasets. On this dataset, SummaC$_{\text{Conv}}$ obtains state-of-the-art results with a balanced accuracy of 74.4%, a 5% improvement compared with prior work.

1 Introduction

Recent progress in text summarization has been remarkable, with ROUGE record-setting models published every few months, and human evaluations indicating that automatically generated summaries are matching human-written summaries in terms of fluency and informativeness (Zhang et al., 2020a).

A major limitation of current summarization models is their inability to remain factually consistent with the respective input document. Summary inconsistencies are diverse—from inversions (i.e., negation) to incorrect use of an entity (i.e., subject, object swapping), or hallucinations (i.e., introduction of entity not in the original document). Recent studies have shown that in some scenarios, even state-of-the-art pre-trained language models can generate inconsistent summaries in more than 70% of all cases (Pagnoni et al., 2021). This has led to accelerated research around summary inconsistency detection.

A closely related task to inconsistency detection is textual entailment, also referred to as Natural Language Inference (NLI), in which a *hypothesis* sentence must be classified as either entailed by, neutral, or contradicting a *premise* sentence. Enabled by the crowd-sourcing of large NLI datasets such as SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2018), modern architectures have achieved close to human performance at the task.

The similarity of NLI to inconsistency detection, as well as the availability of high-performing NLI models, led to early attempts at using NLI to detect consistency errors in summaries. These early attempts were unsuccessful, finding that re-ranking summaries according to an NLI model can lead to an increase in consistency errors (Falke et al., 2019), or that out-of-the-box NLI models obtain 52% accuracy at the binary classification task of inconsistency detection, only slightly above random guessing (Kryscinski et al., 2020).

In this work, we revisit this approach, showing that NLI models *can* in fact successfully be used for inconsistency detection, as long as they are used at the appropriate *granularity*. Figure 1 shows how crucial using the correct granularity as input to NLI models is. An inconsistency checker should flag the last sentence in the summary (shown right) as problematic. When treating the entire document as the premise and the summary as the hypothesis, a competitive NLI model predicts with probability of 0.91 that the summary is entailed by the document. However, when splitting the documents into sentence premise-hypothesis pairs (visualized as edges in Figure 1) the NLI model correctly determines that S_3 is not supported by any document sentence. This illustrates that working with sentence pairs is crucial for making NLI models work for inconsistency detection.

*Author emails: {phillab,hearst}@berkeley.edu, {Tobias.Schnabel,Paul.N.Bennett}@microsoft.com

Transactions of the Association for Computational Linguistics, vol. 10, pp. 163–177, 2022. https://doi.org/10.1162/tacl_a_00453
Action Editor: Shay Cohen. Submission batch: 8/2021; Revision batch: 11/2021; Published 2/2022.

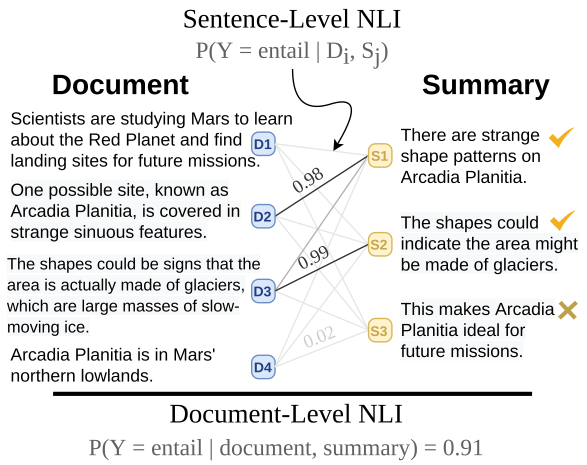
Sentence-Level NLI
P(Y = entail | D$_i$, S$_j$)
Document
Summary
Scientists are studying Mars to learn about the Red Planet and find landing sites for future missions.
D1
One possible site, known as Arcadia Planitia, is covered in strange sinuous features.
D2
The shapes could be signs that the area is actually made of glaciers, which are large masses of slow-moving ice.
D3
Arcadia Planitia is in Mars' northern lowlands.
D4
0.98
0.99
0.02
S1
There are strange shape patterns on Arcadia Planitia. ✓
S2
The shapes could indicate the area might be made of glaciers. ✓
S3
This makes Arcadia Planitia ideal for future missions. ✗
Document-Level NLI
P(Y = entail | document, summary) = 0.91

Figure 1: **Example document with an inconsistent summary.** When running each sentence pair (D_i, S_j) through an NLI model, S_3 is not entailed by any document sentence. However, when running the entire (document, summary) at once, the NLI model incorrectly predicts that the document highly entails the entire summary.

Our contributions are two-fold. First, we introduce a new approach for inconsistency detection based on the aggregation of sentence-level entailment scores for each pair of input document and summary sentences. We present two model variants that differ in the way they aggregate sentence-level scores into a single score. SummaC$_{\text{ZS}}$ performs zero-shot aggregation by combining sentence-level scores using `max` and `mean` operators. SummaC$_{\text{Conv}}$ is a trained model consisting of a single learned convolution layer compiling the distribution of entailment scores of all document sentences into a single score.

Second, to evaluate our approach, we introduce the SummaC Benchmark by standardizing existing datasets. Because the benchmark contains the six largest summary consistency datasets, it is more comprehensive and includes a broader range of inconsistency errors than prior work.

The SummaC models outperform existing inconsistency detection models on the benchmark, with the SummaC$_{\text{Conv}}$ obtaining an overall balanced accuracy of 74.4%, 5% above prior work. We publicly release the models and datasets.[1]

2 Related Work

We briefly survey existing methods and datasets for fact checking, inconsistency detection, and inconsistency correction.

2.1 Fact Checking and Verification

Fact checking is a related task in which a model receives an input claim along with a corpus of ground truth information. The model must then retrieve relevant evidence and decide whether the claim is supported, refuted, or if there is not enough information in the corpus (Thorne et al., 2018). The major difference to our task lies in the different semantics of consistency and accuracy. If a summary adds novel and accurate information not present in the original document (e.g., adding background information), the summary is accurate but inconsistent. In the summary inconsistency detection domain, the focus is on detecting any inconsistency, regardless of its accuracy, as prior work has shown that current automatic summarizers are predominantly inaccurate when inconsistent (Maynez et al., 2020).

2.2 Datasets for Inconsistency Detection

Several datasets have been annotated to evaluate model performance in inconsistency detection, typically comprising up to two thousand annotated summaries. Datasets are most commonly crowd-annotated with three judgements each, despite some work showing that as many as eight annotators are required to achieve high inter-annotator agreement (Falke et al., 2019).

Reading the entire original document being summarized is time-consuming, and to amortize this cost, consistency datasets often contain multiple summaries, generated by different models, for the same original document.

Some datasets consist of an overall consistency label for a summary (e.g., FactCC [Kryscinski et al., 2020]), while others propose a finer-grained typology with up to 8 types of consistency errors (Huang et al., 2020).

We include the six largest summary consistency datasets in the SummaC Benchmark, and describe them more in detail in Section 4.

2.3 Methods for Inconsistency Detection

Due to data limitations, most inconsistency detection methods adapt NLP pipelines from other tasks including QAG models, synthetic classifiers, and parsing-based methods.

QAG methods follow three steps: (1) question generation (QG), (2) question answering (QA) with the document and the summary, (3) matching document and summary answers. A summary is

[1] `https://github.com/tingofurro/summac/`.

considered consistent if few or no questions have differing answer with the document. A key design choice for these methods lies in the source for question generation. Durmus et al. (2020) generate questions using the summary as a source, making their FEQA method precision-oriented. Scialom et al. (2019) generate questions with the document as a source, creating a recall-focused measure. Scialom et al. (2021) unite both in QuestEval, by generating two sets of questions, sourced from the summary and document respectively. We include FEQA and QuestEval in our benchmark results.

Synthetic classifiers rely on large, synthetic datasets of summaries with inconsistencies, and use those to train a classifier with the expectation that the model generalizes to non-synthetic summaries. To generate a synthetic dataset, Kryscinski et al. (2020) propose a set of semantically invariant (e.g., paraphrasing) and variant (e.g., sentence negation) text transformations that they apply to a large summarization dataset. `FactCC-CLS`, the classifier obtained when training on the synthetic dataset, is included in our benchmark results for comparison.

Parsing-based methods generate relations through parsing and compute the fraction of summary relations that are compatible with docu ment relations as a precision measure of summary factuality. Goodrich et al. (2019) extract `(subject, relation, object)` tuples most commonly using OpenIE (Etzioni et al., 2008). In the recent DAE model, Goyal and Durrett (2020) propose to use arc labels from a dependency parser instead of relation triplet. We include the DAE model in our benchmark results.

2.4 Methods for Consistency Correction

Complementary to inconsistency detection, some work focused on the task of mitigating inconsistency errors during summarization. Approaches fall in two categories: Reinforcement Learning (RL) methods to improve models and stand-alone re-writing methods.

RL methods often rely on an out-of-the-box inconsistency detection model and use reinforcement learning to optimize a reward with a consistency component. Arumae and Liu (2019) optimize a QA-based consistency reward, and Nan et al. (2021) streamline a QAG reward by combining the QG and QA model, making it more efficient for RL training. Pasunuru and Bansal (2018) leverage an NLI-based component as part of an overall ROUGE-based reward, and Zhang et al. (2020b) use a parsing-based measure in the domain of medical report summarization.

Re-writing methods typically operate as a modular component that is applied after an existing summarization model. Cao et al. (2020) use a synthetic dataset of rule-corrupted summaries to train a post-corrector model, but find that this model does not transfer well to real summarizer errors. Dong et al. (2020) propose to use a QAG model to find erroneous spans, which are then corrected using a post-processing model.

Since all methods discussed above for consistency correction rely on a model to detect inconsistencies, they will naturally benefit from more accurate inconsistency detectors.

3 SummaC Models

We now introduce our SummaC models for inconsistency detection. The first step common to all models is to apply an out-of-the-box NLI model to generate an *NLI Pair Matrix* for a `(document, summary)` pair. The two models we present then differ in the way they process this pair matrix to produce a single consistency score for a given summary. We also describe the SummaC evaluation benchmark, a set of inconsistency detection datasets, in Section 4. In Section 5, we measure the performance of the SummaC models on this benchmark and investigate components of the models, including which NLI model achieves highest performance, which NLI categories should be used, and what textual granularity is most effective.

3.1 Generating the NLI Pair Matrix

NLI datasets are predominantly represented at the sentence level. In our pilot experiments, we found that this causes the resulting NLI models to fail in assessing consistency for documents with 50 sentences and more.

This motivates the following approach. We generate an NLI Pair Matrix by splitting a `(document, summary)` pair into sentence blocks. The document is split into M blocks, each considered a premise labeled from $D_1, \ldots, D_M$, and the summary is split into N blocks, each considered a hypothesis labeled from $S_1, \ldots, S_N$.

Each D_i, S_j combination is run through the NLI model, which produces a probability distribution over the three NLI categories (E_{ij}, C_{ij}, N_{ij})

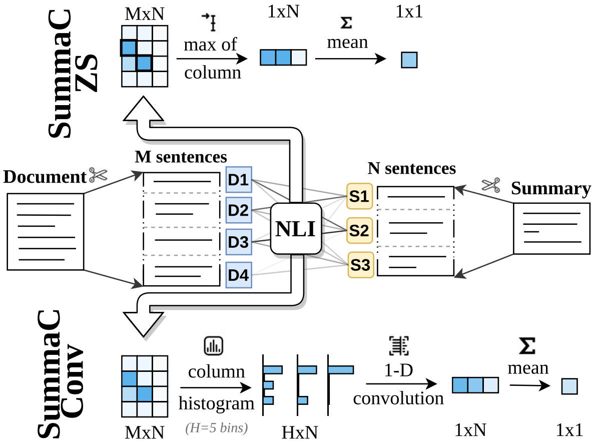

Figure 2: **Diagram of the SummaC$_{ZS}$ (top) and SummaC$_{Conv}$ (bottom) models.** Both models utilize the same NLI Pair Matrix (middle) but differ in their processing to obtain a score. The SummaC$_{ZS}$ is **Z**ero-**S**hot, and does not have trained parameters. SummaC$_{Conv}$ uses a convolutional layer trained on a binned version of the NLI Pair Matrix.

for entailment, contradiction, and neutral, respectively. If not specified otherwise, the pair matrix is an $M \times N$ matrix consisting of the entailment scores E_{ij}. In Section 5.3.3, we examine the effect of granularity by splitting texts at the paragraph level or binning two sentences at a time. In Section 5.3.2, we explore the use of the contradiction and neutral categories in our experiments.

The example in Figure 1 has $M = 4$ document sentences, and $N = 3$ summary sentences, and the corresponding NLI Pair Matrix is the following:

$$X_{pair} = \begin{bmatrix} 0.02 & 0.02 & 0.04 \\ 0.98 & 0.00 & 0.00 \\ 0.43 & 0.99 & 0.00 \\ 0.00 & 0.00 & 0.01 \end{bmatrix}$$

The pair matrix can be interpreted as the weights of a bipartite graph, which is also illustrated in Figure 1 where the opacity of each edge (i, j) represents the entailment probability E_{ij}.

The two SummaC models take as input the same NLI Pair Matrix, but differ in the aggregation method to transform the pair matrix into a score. Figure 2 presents an overview of SummaC$_{ZS}$ and SummaC$_{Conv}$.

3.2 SummaC$_{ZS}$: Zero-Shot

In the SummaC$_{ZS}$ model, we reduce the pair matrix to a one-dimensional vector by taking the maximum (`max`) value of each column. On an intuitive level, for each summary sentence, this step consists of retaining the score for the document sentence that provides the strongest support for each summary sentence. For the example in Figure 1:

$$\max(X_{pair}, \text{axis='col'}) = \begin{bmatrix} 0.98 & 0.99 & 0.04 \end{bmatrix}$$

The second step consists of taking the `mean` of the produced vector, reducing the vector to a scalar which is used as the final model score. At a high level, this step aggregates sentence-level information into a single score for the entire summary. For example, in Figure 1, the score produced by SummaC$_{ZS}$ would be 0.67. If we removed the third sentence from the summary, the score would increase to 0.985. We experiment with replacing the `max` and `mean` operators with other operators in Appendix B.

3.3 SummaC$_{Conv}$: Convolution

One limitation of SummaC$_{ZS}$ is that it is highly sensitive to extrema, which can be noisy due to the presence of outliers and the imperfect nature of NLI models. In SummaC$_{Conv}$, we reduce the reliance on extrema values by instead taking into account the entire distribution of entailment scores for each summary sentence. For each summary sentence, a learned convolutional layer is in charge of converting the entire distribution into a single score.

The first step of the SummaC$_{Conv}$ algorithm is to turn each column of the NLI Pair Matrix into a fixed-size histogram that represents the distribution of scores for that given summary sentence.

We bin the NLI scores into H evenly spaced bins (e.g., if $H = 5$, the bins are $[0, 0.2)$, $[0.2, 0.4)$, $[0.4, 0.6)$, $[0.6, 0.8)$, $[0.8, 1)$). Thus the first summary sentence of the example in Figure 1 would have the following histogram: $[2, 0, 1, 0, 1]$, because there are two values between $[0.0, 0.2]$ in the first column, one in $[0.4, 0.6]$ and one in $[0.8, 1.0]$.

By producing one histogram for each summary sentence, the binning process in the example of Figure 1 would produce:

$$\text{bin}(X_{pair}) = \begin{bmatrix} 2 & 3 & 4 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

Dataset	Size		% Positive	IAA	Source	# Summarizer	# Sublabel
	Valid.	Test					
CoGenSumm (Falke et al., 2019)	1281	400	49.8	0.65	C	3	0
XSumFaith (Maynez et al., 2020)	1250	1250	10.2	0.80	X	5	2
Polytope (Huang et al., 2020)	634	634	6.6	–	C	10	8
FactCC (Kryscinski et al., 2020)	931	503	85.0	–	C	10	0
SummEval (Fabbri et al., 2021)	850	850	90.6	0.7	C	23	4
FRANK (Pagnoni et al., 2021)	671	1575	33.2	0.53	C+X	9	7

Table 1: **Statistics of the six datasets in the SummaC Benchmark.** For each dataset, we report the validation and test set sizes, the percentage of summaries with positive (consistent) labels (**% Positive**), the inter-annotator agreement (when available, **IAA**), the source of the documents (**Source**: C for CNN/DM, X for XSum), the number of summarizers evaluated, and the number of sublabels annotated.

The binned matrix is then passed through a 1-D convolution layer with a kernel size of H. The convolution layer scans the summary histograms one at a time, and compiles each into a scalar value for each summary. Finally, the scores of each summary sentence are averaged to obtain the final summary-level score.

In order to learn the weights of the convolution layer, we train the SummaC$_{\text{Conv}}$ model end-to-end with the synthetic training data in FactCC (Kryscinski et al., 2020). The original training dataset contains one million `(document, summary)` pairs evenly distributed with consistent and inconsistent summaries. Because we are only training a small set of H parameters (we use $H = 50$), we find that using a 10,000 sub-sample is sufficient. We train the model using a cross-entropy loss, the Adam optimizer, a batch size of 32, and a learning rate of 10^{-2}. We perform hyper-parameter tuning on a validation set from the FactCC dataset.

The number of bins used in the binning process, which corresponds to the number of parameters in the convolution layer, is also a hyper-parameter we tune on the validation set. We find that performance increases until 50 bins (i.e., a bin width of 0.02) and then plateaus. We use 50 bins in all our experiments.

4 SummaC Benchmark

To rigorously evaluate the SummaC models on a diverse set of summaries with consistency judgements, we introduce a new large benchmark dataset, the SummaC Benchmark. It comprises the six largest available datasets for summary inconsistency detection, which we standardize to use the same classification task.

4.1 Benchmark Standardization

We standardize the task of summary inconsistency detection by casting it as a binary classification task. Each dataset contains `(document, summary, label)` samples, where the label can either be *consistent* or **inconsistent**.

Each dataset is divided into a validation and test split, with the validation being available for parameter tuning. We used existing validation/test splits created by dataset authors when available. We did not find a split for XSumFaith, Polytope, and SummEval, and created one by putting even-indexed samples in a validation split, and odd-indexed samples in the test split. This method of splitting maintains similar class imbalance and summarizer identity with the entire dataset.

We computed inter-annotator agreement calculated with Fleiss' Kappa (Fleiss, 1971) on the dataset as an estimate for dataset quality, omitting datasets for which summaries only had a single annotator (Polytope and FactCC). Table 1 summarizes dataset statistics and properties.

4.2 Benchmark Datasets

We introduce each dataset in the benchmark chronologically, and describe the standardizing procedure.

CoGenSumm (**Co**rrectness of **Gen**erated **Summ**aries, CGS) (Falke et al., 2019) is the first introduced dataset for summary inconsistency detection, based on models trained on the CNN/DM dataset (Nallapati et al., 2016). The authors proposed that consistency detection should be approached as a ranking problem: Given a consistent and inconsistent summary for a common document, a ranking model should score the consistent summary higher. Although innovative, other datasets in the benchmark do not always have positive and negative samples for a given document. We thus map the dataset to a classification task by using all inconsistent and consistent summaries as individual samples.

XSumFaith (e**X**treme **Sum**marization **Faith**fulness, XSF) (Maynez et al., 2020) is a dataset with models trained on the XSum dataset (Narayan et al., 2018), which consists of more abstractive summaries than CoGenSumm. The authors find that standard generators remain consistent for only 20-30% of generated summaries. The authors differentiate between *extrinsic* and *intrinsic* hallucinations (which we call inconsistencies in this work). Extrinsic hallucinations, which involve words or concepts not in the original document can nonetheless be *accurate* or *inaccurate*. In order for a summarizer to generate an accurate extrinsic hallucination, the summarizer must possess external world knowledge. Because the authors found that the models are primarily inaccurate in terms of extrinsic hallucinations, we map both extrinsic and intrinsic hallucinations to a common inconsistent label.

Polytope (Huang et al., 2020) introduces a more extensive typology of summarization errors, based on the Multi-dimensional Quality Metric (Mariana, 2014). Each summary is annotated with eight possible errors, as well as a severity level for the error. We standardize this dataset by labeling a summary as inconsistent if it was annotated with any of the five accuracy errors (and disregarded the three fluency errors). Each summary in Polytope was labeled by a single annotator, making it impossible to measure inter-annotator agreement.

FactCC (Kryscinski et al., 2020) contains validation and test splits that are entirely annotated by authors of the paper, because attempts at crowd-sourced annotation yielded low inter-annotator agreement. Prior work (Gillick and Liu, 2010) shows that there can be divergence in annotations between experts and non-experts in summarization, and because the authors of the paper are NLP researchers familiar with the limitations of automatic summarizations, we expect that FactCC annotations differs in quality from other datasets. FactCC also introduces a synthetic dataset by modifying consistent summaries with semantically variant rules. We use a sub-portion of this synthetic dataset to train the SummaC$_{\text{Conv}}$ model.

SummEval (Fabbri et al., 2021) contains summarizer outputs from seven extractive models and sixteen abstractive models. Each summary was labeled using a 5-point Likert scale along four categories: coherence, consistency, fluency, and relevance by 3 annotators. We label summaries as consistent if all annotators gave a score of 5 in consistency, and inconsistent otherwise.

FRANK (Pagnoni et al., 2021) contains annotations for summarizers trained on both CNN/DM and XSum, with each summary annotated by three crowd-workers. The authors propose a new typology with seven error types, organized into semantic frame errors, discourse errors and content verifiability errors. The authors confirm that models trained on the more abstractive XSum dataset generate a larger proportion of inconsistent summaries, compared to models trained on CNN/DM. We label summaries as consistent if a majority of annotators labeled the summary as containing no error.

4.3 Benchmark Evaluation Metrics

With each dataset in the SummaC Benchmark converted to a binary classification task, we now discuss the choice of appropriate evaluation metrics for the benchmark. Previous work on each dataset in the benchmark used different evaluation methods, falling into three main categories.

First, CoGenSumm proposes a re-ranking based measure, requiring pairs of consistent and inconsistent summaries for any document evaluated; this information is not available in several datasets in the benchmark.

Second, XSumFaith, SummEval, and FRANK report on correlation of various metrics with human annotations. Correlation has some advantages, such as not requiring a threshold and being compatible with the Likert-scale annotations of SummEval, however it is an uncommon choice to measure performance of a classifier due to the discrete and binary label.

Third, authors of FactCC measured model performance using binary F1 score, and balanced accuracy, which corrects unweighed accuracy with the class imbalance ratio, so that majority class voting obtains a score of 50%.

The datasets have widely varying class imbalances, ranging from 6% to 91% positive samples. Therefore, we select balanced accuracy (Brodersen et al., 2010) as the primary evaluation metric for the SummaC Benchmark. Balanced accuracy is defined as:

$$BAcc = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \quad (1)$$

Where TP stands for true positive, FP false positive, TN true negative, and FN false negative. The choice of metric is based on the fact that accuracy is a conceptually simple, interpretable metric, and that adjusting the class imbalance out of the metric makes the score more uniform across datasets.

The balanced accuracy metric requires models to output a binary label (i.e., not a scalar score), which for most models requires the selection of a threshold in the score. The threshold is selected using the validation set, allowing for a different threshold for each dataset in the benchmark. Performance on the benchmark is the unweighted average of performance on the six datasets.

We choose Area Under the Curve of the Receiver Operating Chart (ROC-AUC) as a secondary evaluation metric, a common metric to summarize a classifier's performance at different threshold levels (Bradley, 1997).

5 Results

We compared the SummaC models against a wide array of baselines and state-of-the-art methods.

5.1 Comparison Models

We evaluated the following models on the SummaC Benchmark:

NER Overlap uses the spaCy named entity recognition (NER) model (Honnibal et al., 2020) to detect when an entity present in the summary is not present in the document. This model, adapted from Laban et al. (2021), considers only a subset of entity types as hallucinations (*PERSON, LOCATION, ORGANIZATION*, etc.)

MNLI-doc is a RoBERTa (Liu et al., 2019) model finetuned on the MNLI dataset (Williams et al., 2018). The document is used as the premise and the summary as a hypothesis, and we use the predicted probability of entailment as a score, similar to prior work on using NLI models for inconsistency detection (Kryscinski et al., 2020).

FactCC-CLS is a RoBERTa-base model finetuned on the synthetic training portion of the FactCC dataset. Although trained solely on artificially created inconsistent summaries, prior work showed the model to be competitive on the FactCC and FRANK datasets.

DAE (Goyal and Durrett, 2020) is a parsing-based model using the default model and hyper-parameters provided by the authors of the paper.[2]

FEQA (Durmus et al., 2020) is a QAG method, using the default model and hyper-parameters provided by the authors of the paper.[3]

QuestEval (Scialom et al., 2021) is a QAG method taking both precision and recall into account. We use the default model and hyper-parameters provided by the authors of the paper.[4] The model has an option to use an additional question weighter, however experiments revealed that the weighter lowered overall performance on the validation portion of the SummaC Benchmark, and we compare to the model without weighter.

5.2 SummaC Benchmark Results

Balanced accuracy results are summarized in Table 2. We find that the SummaC models achieve the two best performances in the benchmark. SummaC$_{\text{Conv}}$ achieves the best benchmark performance at 74.4%, 5 points above QuestEval, the best method not involving NLI.

Looking at the models' ability to generalize across datasets and varying scenarios of inconsistency detection provides interesting insights. For example, the FactCC-CLS model achieves strong performance on the FactCC dataset, but close to lowest performance on FRANK and XSumFaith. In comparison, SummaC model performance is strong across the board.

The strong improvement from the SummaC$_{\text{ZS}}$ to SummaC$_{\text{Conv}}$ also shines a light on the importance of considering the entire distribution of document scores for each summary sentence, instead of taking only the maximum score: The SummaC$_{\text{Conv}}$ model learns to look at the distribution and makes more robust decisions, leading to gains in performance.

The table of results with the ROC-AUC metric, the secondary metric of the SummaC Benchmark,

[2] https://github.com/tagoyal/dae-factuality.

[3] https://github.com/esdurmus/feqa.

[4] https://github.com/ThomasScialom/QuestEval.

		SummaC Benchmark Datasets							
Model Type	**Model Name**	**CGS**	**XSF**	**Polytope**	**FactCC**	**SummEval**	**FRANK**	**Overall**	**Doc./min.**
Baseline	NER-Overlap	53.0	63.3	52.0	55.0	56.8	60.9	56.8	55,900
	MNLI-doc	57.6	57.5	61.0	61.3	66.6	63.6	61.3	6,200
Classifier	FactCC-CLS	63.1	57.6	61.0	75.9	60.1	59.4	62.8	13,900
Parsing	DAE	63.4	50.8	62.8	75.9	70.3	61.7	64.2	755
QAG	FEQA	61.0	56.0	57.8	53.6	53.8	69.9	58.7	33.9
	QuestEval	62.6	62.1	**70.3***	66.6	72.5	**82.1**	69.4	22.7
NLI	SummaC$_{\text{ZS}}$	**70.4***	58.4	62.0	83.8*	78.7	79.0	72.1*	435
	SummaC$_{\text{Conv}}$	64.7	**66.4***	62.7	**89.5****	**81.7****	**81.6**	**74.4****	433

Table 2: **Performance of Summary Inconsistency Detection models on the test set of the SummaC Benchmark.** Balanced accuracy is computed for each model on the six datasets in the benchmark, and the average is computed as the overall performance on the benchmark. We obtain confidence intervals comparing the SummaC models to prior work: * indicates an improvement with 95% confidence, and ** 99% confidence (details in Section 5.2.1). The results of the throughput analysis of Section 5.2.2 are in column Doc./min (Documents per minute).

is included in Appendix A2, echoing the trends seen with the balanced accuracy metric.

5.2.1 Statistical Testing

We aim to determine whether the performance improvements of the SummaC models are statistically significant. For each dataset of the benchmark, we perform two tests through bootstrap resampling (Efron, 1982), comparing each of the SummaC models to the best-performing model from prior work. We perform interval comparison at two significance level: $p = 0.05$ and $p = 0.01$, and apply the Bonferroni correction (Bonferroni, 1935) as we perform several tests on each dataset. We summarize which improvements are significant in Table 2, and perform a similar testing procedure for the ROC-AUC results in Table A2.

SummaC models lead to a statistically significant improvement on CoGenSumm, XSumFaith, FactCC, and SummEval. QuestEval outperforms the SummaC models on Polytope at a confidence of 95%. On the FRANK dataset, QuestEval and SummaC$_{\text{Conv}}$ achieve highest performance with no statistical difference. Overall on the benchmark, both SummaC models significantly outperform prior work, SummaC$_{\text{ZS}}$ at a $p = 0.05$ significance level and SummaC$_{\text{Conv}}$ at $p = 0.01$.

5.2.2 Computational Cost Comparison

Computational cost of the method is an important practical factor to consider when choosing a model to use, as some applications such as training with a generator with Reinforcement Learning might require a minimum throughput from the model (i.e., number of documents processed by the model per unit of time).

A common method to compare algorithms is using computational complexity analysis, computing the amount of resources (time, space) needed as the size of the input varies. Computational complexity analysis is impractical in our case, as the units of analysis differ between models, and do not allow for a direct comparison. More specifically, some of the models' complexity scales with the number of sub-word units in the document (`MNLI-doc`, `FactCC-CLS`), some with the number of entities in a document (`NER-Overlap`, `DAE`, `QuestEval`), and some with number of sentences (the SummaC models).

We instead compare models by measuring throughput on a fixed dataset using a common hardware setup. More precisely, we measured the processing time of each model on the 503 documents in the test set of FactCC (with an average of 33.2 sentences per document), running a single Quadro RTX 8000 GPU. For prior work, we used implementation publicly released by the authors, and made a best effort to use the model at an appropriate batch size for a fair comparison.

The result of the throughput analysis is included in Table 2 (column Docs./min.). SummaC models are able to process around 430 documents per minute, which is much lower than some of the baselines capable of processing more than 10,000 documents per minute. However, QAG methods are more than 10 times slower than SummaC models, processing only 20-40 documents per minute.

Architecture	NLI Dataset	Performance	
		ZS	Conv
Dec. Attn	SNLI	56.9	56.4
BERT Base	SNLI	66.6	64.0
	MNLI	69.5	69.8
	MNLI+VitaminC	67.9	71.2
BERT Large	SNLI	66.6	62.4
	SNLI+MNLI+ANLI	69.9	71.7
	VitaminC	71.1	72.8
	MNLI	70.9	73.0
	MNLI+VitaminC	**72.1**	**74.4**

Table 3: **Effect of NLI model choice on SummaC models performance.** For each NLI model, we include the balanced accuracy scores of SummaC$_{ZS}$ and SummaC$_{Conv}$. BERT X corresponds to a BERT or other pre-trained models of similar size.

5.3 Further Results

We now examine how different components and design choices affect SummaC model performance.

5.3.1 Choice of NLI Model

SummaC models rely on an NLI model at their core, which consists of choosing two main components: a model architecture, and a dataset to train on. We investigate the effect of both of these choices on the performance of SummaC models on the benchmark.

Regarding model architectures, we experiment with the decomposable attention model (Parikh et al., 2016), which is a pre-Transformer architecture model that was shown to achieve high performance on SNLI, as well as Transformer base and Transformer Large architectures.

With respect to datasets, we include models trained on standard NLI datasets such as SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2018), as well as more recent datasets such as Adversarial NLI (Nie et al., 2019) and Vitamin C (Schuster et al., 2021).

Results are summarized in Table 3, and we emphasize three trends. First, the low performance of the decomposable attention model used in experiments in prior work (Falke et al., 2019) confirms that less recent NLI models did not transfer well to summary inconsistency detection.

Second, NLI models based on pre-trained Transformer architectures all achieve strong performance on the benchmark, with an average increase of 1.3 percentage points when going from a base to a large architecture.

Third, the choice of NLI dataset has a strong influence on overall performance. SNLI leads to lowest performance, which is expected as its textual domain is based on image captions, which are dissimilar to the news domain. MNLI and Vitamin C trained models both achieve close to the best performance, and training on both jointly leads to the best model, which we designate as the default NLI model for the SummaC models (i.e., the model included in Table 2).

The latter two trends point to the fact that improvements in the field of NLI lead to improvements in the SummaC models, and we can expect that future progress in the NLI community will translate to gains of performance when integrated into the SummaC model.

We relied on trained models available in HuggingFace's Model Hub (Wolf et al., 2020). Details in Appendix A.

5.3.2 Choice of NLI Category

The NLI task is a three-way classification task, yet most prior work has limited usage of the model to the use of the entailment probability for inconsistency detection (Kryscinski et al., 2020; Falke et al., 2019). We run a systematic experiment by training multiple SummaC$_{Conv}$ models that have access to varying subsets of the NLI labels, and measure the impact on overall performance. Results are summarized in Table 4. Using solely the entailment category leads to strong performance for all models. However, explicitly including the contradiction label as well leads to small boosts in performance for the ANLI and MNLI models.

With future NLI models being potentially more nuanced and calibrated, it is possible that inconsistency detector models will be able to rely on scores from several categories.

5.3.3 Choice of Granularity

So far, we've reported experiments primarily with a sentence-level granularity, as it matches the granularity of NLI datasets. One can imagine cases where sentence-level granularity might be limiting. For example, in the case of a summary performing a *sentence fusion* operation, an NLI model might not be able to correctly predict entailment of the fused sentence, seeing only one sentence at a time.

Category			SummaC$_{\text{Conv}}$ Performance		
E	N	C	VITC+MNLI	ANLI	MNLI
✓			**74.4**	69.2	72.6
	✓		71.2	55.8	66.4
		✓	72.5	69.2	72.6
✓	✓		73.1	69.6	72.6
✓		✓	74.0	**70.2**	**73.0**
	✓	✓	72.5	69.2	72.6
✓	✓	✓	74.0	69.7	**73.0**

Table 4: **Effect of NLI category inclusion on SummaC$_{\text{Conv}}$ performance.** Models had access to different subsets of the three category predictions (**E**ntailment, **N**eutral, **C**ontradiction), with performance measured in terms of balanced accuracy. Experiments were performed with 3 NLI models: Vitamic C+MNLI, ANLI, and MNLI.

		Performance			
Granularity		MNLI		MNLI + VitC	
Document	Summary	ZS	Conv	ZS	Conv
Full	Full	56.4	–	72.1	–
	Sentence	57.4	–	73.1	–
Paragraph	Full	59.8	61.8	69.8	71.2
	Sentence	65.2	64.7	72.6	74.3
Two Sent.	Full	64.0	63.8	69.7	71.3
	Sentence	71.2	**73.5**	72.5	**74.7**
Sentence	Full	58.7	61.1	68.4	69.4
	Sentence	70.3	**73.0**	72.1	**74.4**

Table 5: **Effect of granularity choice on SummaC models performance.** We tested four granularities on the document side: full, paragraph, two sentence, and sentence, and two granularities on the summary side: full and sentence. Performance of the four models is measured in balanced accuracy on the benchmark test set.

To explore this facet further, we experiment with modifying the granularity of both the document and the summary. With regard to document granularity, we consider four granularities: (1) **full text**, the text is treated as a single block, (2) **paragraph-level** granularity, the text is separated into paragraph blocks, (3) **two-sentence** granularity, the text is separated into blocks of contiguous sentences of size two (i.e., block 1 contains sentence 1-2, block 2 contains sentence 3-4), and (4) **sentence-level**, splitting text at individual sentences. For the summary granularity, we only consider two granularities: (1) **full text**, and (2) **sentence**, because other granularities are less applicable since summaries usually consist of three sentences or fewer.

We study the total of 8 `(document, summary)` granularity combinations with the two best-performing NLI models of Table 2: MNLI and Vitamin C, each included as SummaC$_{\text{ZS}}$ and SummaC$_{\text{Conv}}$ models.[5]

Results for the granularity experiments are summarized in Table 5. Overall, finer granularities lead to better performance, with `(sentence, sentence)` and `(two sent, sentence)` achieving highest performance across all four models.

The MNLI-only trained model achieves lowest performance when used with full text granularity on the document level, and performance steadily increases from 56.4% to 73.5% as granularity is made finer both on the document and summary side. Results for the MNLI+VitaminC model vary less with changing granularity, showcasing that the model is perhaps more robust to different granularity levels. However the `(two sent, sentence)` and `(sentence,sentence)` settings achieve highest performance, implying that finer granularity remains valuable.

For all models, performance degrades in cases where granularity on the document level is finer than summary granularity. For example the `(sentence, full)` or `(two sent, full)` combinations lead to some of the lowest performance. This is expected, as in cases in which summaries have several sentences, it is unlikely that they will fully be entailed by a single document sentence. This implies that granularity on the document side should be coarser or equal the summary's granularity.

Overall, we find that finer granularity for the document and summary is beneficial in terms of performance and recommend the use of a `(sentence, sentence)` granularity combination.

6 Discussion and Future Work

Improvements on the Benchmark. The models we introduced in this paper are just a first step towards harnessing NLI models for inconsistency detection. Future work could explore a number of improvements: combining the predictions of multiple NLI models, or combining multiple granularitiy levels—for example, through multi-hop reasoning (Zhao et al., 2019).

[5] We skip SummaC$_{\text{Conv}}$ experiments involving full text granularity on the document-side, as that case reduces the binning process to having a single non-zero value.

Interpretability of Model Output. If a model can pinpoint which portion of a summary is inconsistent, some work has shown that corrector models can effectively re-write the problematic portions and often remove the inconsistency (Dong et al., 2020). Furthermore, fine-grained consistency scores can be incorporated into visual analysis tools for summarization such as SummViz (Vig et al., 2021). The SummaC$_{ZS}$ model is directly interpretable, whereas the SummaC$_{Conv}$ is slightly more opaque, due to the inability to trace back a low score to a single sentence in the document being invalidated. Improving the interpretability of the SummaC$_{Conv}$ model is another open area for future work.

Beyond News Summarization. The six datasets in the SummaC Benchmark contain summaries from the news domain, one of the most common application of summarization technology. Recent efforts to expand the application of summarization to new domains such as legal (Kornilova and Eidelman, 2019) or scholarly (Cachola et al., 2020) text will hopefully lead to the study of inconsistency detection in these novel domains, and perhaps even out of summarization on tasks such as text simplification, or code generation.

Towards Consistent Summarization. Inconsistency detection is but a first step in eliminating inconsistencies from summarization. Future work can include more powerful inconsistency detectors in the training of next generation summarizers to reduce the prevalence of inconsistencies in generated text.

7 Conclusion

We introduce SummaC$_{ZS}$ and SummaC$_{Conv}$, two NLI-based models for summary inconsistency detection based on the key insight that NLI models require sentence-level input to work best. Both models achieve strong performance on the SummaC Benchmark, a new diverse and standardized collection of the six largest datasets for inconsistency detection. SummaC$_{Conv}$ outperforms all prior work with a balanced accuracy score of 74.4%, an improvement of five absolute percentage points over the best baseline. To the best of our knowledge, this the first successful attempt at adapting NLI models for inconsistency detection, and we believe that there are many exciting opportunities for further improvements and applications of our methods.

Acknowledgments

We would like to thank Katie Stasaski, Dongyeop Kang, and the TACL reviewers and editors for their helpful comments, as well as Artidoro Pagnoni for helpful pointers during the project. This work was supported by a Microsoft BAIR Commons grant as well as a Microsoft Azure Sponsorship.

References

Kristjan Arumae and Fei Liu. 2019. Guiding extractive summarization with question-answering rewards. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2566–2577. https://doi.org/10.18653/v1/N19-1264

Carlo E. Bonferroni. 1935. Il calcolo delle assicurazioni su gruppi di teste. *Studi in onore del professore salvatore ortu carboni*, pages 13–60.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Andrew P. Bradley. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159. https://doi.org/10.1016/S0031-3203(96)00142-2

Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M. Buhmann. 2010. The balanced accuracy and its posterior distribution. In *2010 20th International Conference on Pattern Recognition*, pages 3121–3124. IEEE. https://doi.org/10.1109/ICPR.2010.764

Isabel Cachola, Kyle Lo, Arman Cohan, and Daniel S. Weld. 2020. Tldr: Extreme summarization of scientific documents. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4766–4777. https://doi.org/10.18653/v1/2020.findings-emnlp.428

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. https://doi.org/10.18653/v1/2020.emnlp-main.506

Yue Dong, Shuohang Wang, Zhe Gan, Yu Cheng, Jackie Chi Kit Cheung, and Jingjing Liu. 2020. Multi-fact correction in abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9320–9331. https://doi.org/10.18653/v1/2020.emnlp-main.749

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. https://doi.org/10.18653/v1/2020.acl-main.454

Bradley Efron. 1982. The jackknife, the bootstrap and other resampling plans. In *CBMS-NSF Regional Conference Series in Applied Mathematics*. https://doi.org/10.1145/1409360.1409378

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. 2008. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74.

Alexander R. Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. Summeval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409. https://doi.org/10.1162/tacl_a_00373

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. https://doi.org/10.1037/h0031619

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378.

Dan Gillick and Yang Liu. 2010. Non-expert evaluation of summarization systems is risky. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 148–151.

Ben Goodrich, Vinay Rao, Peter J. Liu, and Mohammad Saleh. 2019. Assessing the factual accuracy of generated text. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 166–175. https://doi.org/10.1145/3292500.3330955

Tanya Goyal and Greg Durrett. 2020. Evaluating factuality in generation with dependency-level entailment. *arXiv preprint arXiv:2010.05478*.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python. https://doi.org/10.18653/v1/2020.findings-emnlp.322

Dandan Huang, Leyang Cui, Sen Yang, Guangsheng Bao, Kun Wang, Jun Xie, and Yue Zhang. 2020. What have we achieved on text summarization? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 446–469. https://doi.org/10.18653/v1/2020.emnlp-main.33

Anastassia Kornilova and Vladimir Eidelman. 2019. Billsum: A corpus for automatic summarization of us legislation. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 48–56.

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. https://doi.org/10.18653/v1/2020.emnlp-main.750

Philippe Laban, Tobias Schnabel, Paul Bennett, and Marti A. Hearst. 2021. Keep it simple: Unsupervised simplification of multi-paragraph text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6365–6378, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.498

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Valerie R. Mariana. 2014. *The Multidimensional Quality Metric (MQM) framework: A new framework for translation quality assessment*. Brigham Young University.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. *arXiv preprint arXiv:2005.00661*. https://doi.org/10.18653/v1/2020.acl-main.173

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, 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. https://doi.org/10.18653/v1/K16-1028

Feng Nan, Cicero Nogueira dos Santos, Henghui Zhu, Patrick Ng, Kathleen McKeown, Ramesh Nallapati, Dejiao Zhang, Zhiguo Wang, Andrew O. Arnold, and Bing Xiang. 2021. Improving factual consistency of abstractive summarization via question answering. *arXiv preprint arXiv:2105.04623*. https://doi.org/10.18653/v1/2021.acl-long.536

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. https://doi.org/10.18653/v1/D18-1206

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2019. Adversarial NLI: A new benchmark for natural language understanding. *arXiv preprint arXiv:1910.14599*. https://doi.org/10.18653/v1/2020.acl-main.441

Artidoro Pagnoni, Vidhisha Balachandran, and Yulia Tsvetkov. 2021. Understanding factuality in abstractive summarization with frank: A benchmark for factuality metrics. In *NAACL*.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 646–653. https://doi.org/10.18653/v1/N18-2102

Tal Schuster, Adam Fisch, and Regina Barzilay. 2021. Get your Vitamin C! Robust fact verification with contrastive evidence. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 624–643. https://doi.org/10.18653/v1/2021.naacl-main.52

Thomas Scialom, Paul-Alexis Dray, Patrick Gallinari, Sylvain Lamprier, Benjamin Piwowarski, Jacopo Staiano, and Alex Wang. 2021. Questeval: Summarization asks for fact-based evaluation. *arXiv preprint arXiv:2103.12693*. https://doi.org/10.18653/v1/2021.emnlp-main.529

Thomas Scialom, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2019. Answers unite! unsupervised metrics for reinforced summarization models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3246–3256. `https://doi.org/10.18653/v1/D19-1320`

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: A large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819. `https://doi.org/10.18653/v1/N18-1074`

Jesse Vig, Wojciech Kryscinski, Karan Goel, and Nazneen Fatema Rajani. 2021. Summvis: Interactive visual analysis of models, data, and evaluation for text summarization. *arXiv preprint arXiv:2104.07605*. `https://doi.org/10.18653/v1/2021.acl-demo.18`

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. `https://doi.org/10.18653/v1/N18-1101`

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. 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. `https://doi.org/10.18653/v1/2020.emnlp-demos.6`

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

Yuhao Zhang, Derek Merck, Emily Tsai, Christopher D. Manning, and Curtis Langlotz. 2020b. Optimizing the factual correctness of a summary: A study of summarizing radiology reports. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5108–5120. `https://doi.org/10.18653/v1/2020.acl-main.458`

Chen Zhao, Chenyan Xiong, Corby Rosset, Xia Song, Paul Bennett, and Saurabh Tiwary. 2019. Transformer-XH: Multi-evidence reasoning with extra hop attention. In *International Conference on Learning Representations*.

Appendix

A NLI Model Origin

We list the NLI models we used throughout the paper, which can be retrieved on HuggingFace's model hub.[6] BERT stands for any Pre-trained bi-directional Transformer of an equivalent size:

- `boychaboy/SNLI_roberta-base` BERT Base+SNLI
- `microsoft/deberta-base-mnli` BERT Base+MNLI
- `tals/albert-base-vitaminc-mnli` BERT Base + MNLI + VitaminC
- `boychaboy/SNLI_roberta-large` BERT Large+SNLI
- `tals/albert-xlarge-vitaminc` Bert Large+VitaminC
- `roberta-large-mnli` Bert Large+MNLI
- `tals/albert-xlarge-vitaminc-mnli` BERT Large+MNLI+VitaminC

[6] `https://huggingface.co/models.`

B SummaC$_{\text{ZS}}$ Operator Choice

Table A1 measures the effect of the choice of the two operators in the SummaC$_{\text{ZS}}$ model. We explore three options (`min`, `mean`, and `max`) for each operator. We find that the choice of `max` for Operator 1 and `mean` for Operator 2 achieves the highest performance and use these choices in our model.

C SummaC Benchmark ROC-AUC Results

Table A2 details results of models on the benchmark according to the ROC-AUC metric, confirming that the SummaC models achieve the two best accuracy results on the benchmark.

	Operator 2		
Op. 1	**Min**	**Mean**	**Max**
Min	53.1	55.7	57.4
Mean	60.5	62.8	62.0
Max	68.8	**72.1**	69.1

Table A1: **Effect of operator choice on the performance of the SummaC$_{\text{ZS}}$ model, measured in terms of balanced accuracy.** `Operator 1` reduces the row dimension of the NLI Pair Matrix, and `Operator 2` reduces the column dimension.

		SummaC Benchmark Datasets						
Model Type	**Model Name**	**CGS**	**XSF**	**Polytope**	**FactCC**	**SummEval**	**FRANK**	**Overall**
Baseline	NER-Overlap	53.0	61.7	51.6	53.1	56.8	60.9	56.2
	MNLI-doc	59.4	59.4	62.6	62.1	70.0	67.2	63.4
Classifier	FactCC-CLS	65.0	59.2	63.5	79.6	61.4	62.7	65.2
Parsing	DAE	67.8	41.3	64.1	82.7	77.4	64.3	66.3
QAG	FEQA	60.8	53.4	54.6	50.7	52.2	74.8	57.7
	QuestEval	64.4	66.4	**72.2**	71.5	79.0	**87.9**	73.6
NLI	SummaC$_{\text{ZS}}$	**73.1**	58.0	60.3	83.7	85.5	85.3	74.3
	SummaC$_{\text{Conv}}$	67.6	**70.2**	62.4	92.2**	86.0*	**88.4**	77.8**

Table A2: **Performance of Summary Inconsistency Detection models on the test portion of the SummaC Benchmark** in terms of ROC-AUC metric. The metric is computed for each model on the six datasets in the benchmark, and the average is computed as the overall performance on the benchmark. Confidence intervals comparing the SummaC models to prior work: * indicates an improvement with 95% confidence, and ** 99% confidence (details in Section 5.2.1).