Mingcheng Wang, Eric Xia, Trevor Jung
University of Washington
{wmingch, ericxia, tjung2}@uw.edu

Reproducibility Summary

Scope of Reproducibility

We attempt to reproduce the claim that the authors' decision-focused summarization model, `DecSum`, substantially outperforms text-only summarization methods and model-based explanation methods in terms of decision faithfulness and representativeness.

Methodology

We used the author's code, provided on their Github. We download the Yelp dataset, provided freely on the Yelp website. The author's scripts contain some code errors, and we fixed the code before it ran properly. All 15 simulations (7 were from paper and other 8 were experiments beyond content of paper) are run over two Tesla V100 GPUs. Each simulation of DecSum took 14 hours to run, and training the Longformer took 4 hours.

Results

In line with the paper, DecSum outperforms the text-only summarization methods and model-based explanation methods in terms of decision faithfulness and representativeness. The additional experiments indicate that the performance of DecSum is not sensitive to the model hyperparameters (e.g. max sequence length, summary length) and the training data used.

What was Easy

It was relatively easy to access the authors' code on Github and follow their repo's documentation. It was also easy to access and download the data used in the paper (the yelp dataset).

What was Difficult

The author's scripts contain code errors and took some time to debug them. We also had limited computing resources and running all simulations took a lot more time than we expected. We were confused by the authors' methodology in some parts of the paper, for example how sentences were chosen to compute decision representativeness, particularly for the diagram of the model prediction's PDF on page 8 of the paper.

Communication with Original Authors

We had no contact with the original authors.

1 Introduction

We attempt to reproduce the experiments described in the EMNLP 2021 paper Decision-Focused Summarization by Chao-Chun Hsu and Chenhao Tan. [1] This paper proposes a new technique called decision-focused summarization for the problem of text summarization. Traditionally, text summarization is done only using textual information. However, this has issues. For instance, a common metric used to evaluate text summaries is non-redundancy, which encourages a summary to encompass a diverse range of topics. But if a doctor wanted to analyze the risks of pancreatic cancer in a patient, it may not be useful to include information about a knee injury. Instead, with a decision-focused summarization method, one could create summaries better tailored to a task.

The authors use the following setup: they leverage a supervised decision model, which takes as text as input and makes a decision. Then, the text summarization algorithm seeks to minimize loss on the decision. The authors propose two new desiderata: decision faithfulness and decision representativeness. In addition, they also use textual non-redundancy as a third desiderata.

The authors find that decision-focused summarization outperforms text-only summarization methods and model-based explanation methods. In addition, they show that the decision-focused summarization method outperforms other methods in aiding human decision-making in a challenging classification task.

2 Scope of Reproducibility

Parts we will be able to reproduce:
1. Run DecSum and apply it to Yelp data set to find its performance, including the decision faithfulness and representativeness.
2. Compare the results of the above with the statistics provided in the paper.
3. Compare DecSum's performance to text-only summarization and model-based explanation to see if DecSum indeed has better performance in terms of decision faithfulness and representativeness.

Parts we will not be able to reproduce:
1. The paper runs text-only summarization baselines (e.g., PreSumm, BART, Random) and model-based explanations (e.g. Integrated Gradients and Attention) to compare them with DecSum. Given our time and computing resources, we may rely on the results of text-only summarization baselines and Model-based explanations shown in the paper, instead of running those simulations by ourselves.
2. The paper implemented the human evaluation by using Amazon Mechanical Turk to solicit human guesses on which restaurants would be rated higher after participants. We are not able to implement such a high-scale data collection method like Amazon Mechanical Turk.

2.1 Addressed Claims from the Original Paper

We are testing the claim from the paper that DecSum substantially outperforms text-only summarization methods and model-based explanation methods in terms of decision faithfulness and representativeness.

3 Methodology

3.1 Model Descriptions

The goal is to determine the most relevant information from a text for a particular decision to ultimately be a summary of sorts of human decision-making for a use-case. More formally, given an input text $X = \{x_s\}_{s=1}^{S=1}$, where S is the number of sentences, we wish to select a subset of sentences $\tilde{X} \subset X$ to support making a decision y. Thus, the training set, $D_{train} = \{(X_i, y_i)\}$, will be processed in such a way as to find defining traits in a text that lead to a certain decision y. The yelp datset is then defined as follows: for each restaurant, define X to be the text body of the first k reviews and let y be the average ranking of the first t reviews, where $t > k$ s.t. the problem is to predict future ratings. Here, the parameters are k and t; the researchers used $k = 10, t = 50$ s.t. their objective was to choose sentences amongst a restaurant's first 10 reviews that supported predicting its future review rating after 50 reviews.

3.1.1 The Loss Function

Loss functions were formalized from the need for the model to satisfy "decision faithfulness, decision representativeness, and textual non-redundancy", where $f : X \to y$ is the model's function.

Decision faithfulness $\mathcal{L}_F$. The sentences selected by the model should lead to the same decisions as the full input text: for selected sentences $\tilde{X} \subset X$, we want $f(\tilde{X}) \simeq f(X)$. We take the loss function to be the log absolute difference between $f(\tilde{X})$ and $f(X)$.
Decision representativeness $\mathcal{L}_R$. We want the summary of the decision distribution outputted by the model, $\hat{Y}_{\tilde{X}} = \{f(x) \mid x \in \tilde{X}\}$, to be close to the decision distribution of all sentences from the full text, $\hat{Y}_X = \{f(x) \mid x \in X\}$. Thus, we take $\mathcal{L}_R$ to be the log Wasserstein distance between $\hat{Y}_{\tilde{X}}$ and $\hat{Y}_X$.
Textual non-redundancy $\mathcal{L}_D$. As the model's objective is a summary of the reasoning for the writer's decision, we use cosine similarity to encourage dissimilar sentences in the output summary.

Final loss function. Combining these three, out loss function is

$$\begin{aligned}\mathcal{L}(\tilde{X}, X, f) &= \alpha\mathcal{L}_F(\tilde{X}, X, f) + \beta\mathcal{L}_R(\tilde{X}, X, f) + \gamma\mathcal{L}_D(\tilde{X}) \\ &= \alpha \log|f(\tilde{X}) - f(X)| + \beta \log\left(W(\hat{Y}_{\tilde{X}}, \hat{Y}_X)\right) + \gamma \sum_{x\in\tilde{X}} \max_{x'\in\tilde{X}-\{x\}} \text{cossim}(s(x), s(x')),\end{aligned}$$

where α, β, γ are hyperparameters that determine the weight of each component of the loss function.

3.1.2 The Model

DecSum is an iterative algorithm that greedily selects a sentence that minimizes the loss function: In order to select K sentences from an input X, in each step $k = \{1, \ldots, K\}$, iteratively choose a sentence from the remaining sentences $\hat{x} \in X - \tilde{X}_{k-1}$ that achieves the lowest loss $\mathcal{L}(\tilde{X}_{k-1} \cup \{\hat{x}\}, X, f)$ where $\tilde{X}_{k-1}$ is the current summary using $k-1$ sentences. When β is positive, only $\mathcal{L}_R$ is used in the first step "to encourage the algorithm to explore the full distribution rather than stalling at the sentence most faithful to $f(X)$". In practice, beam search with beam search size 4 was used to improve the greedy algorithm.

A 64%/16%/20% training/validation/test set split was employed, and Longformer [2] (Github code here) was used to fine-tune the regression models.

Input: X, f, K
Output: $\tilde{X}$
$\tilde{X} \leftarrow \emptyset,\ k \leftarrow 1;$
while $k \leq K$ **do**
  if $\beta > 0$ *and* $k = 1$ **then**
    $\hat{x} \leftarrow \text{argmin}_{\hat{x}\in X} \mathcal{L}_R(\{\hat{x}\}, X, f)$
  else
    $\hat{x} \leftarrow \text{argmin}_{\hat{x}\in X} \mathcal{L}(\tilde{X} \cup \{\hat{x}\}, X, f)$
  end
  $\tilde{X} \leftarrow \tilde{X} \cup \{\hat{x}\};$
  $X \leftarrow X - \{\hat{x}\};$
  $k \leftarrow k + 1$
end

Figure 1: DecSum Algorithm

3.2 Dataset Description and Code

We use the author's code, provided on their Github. After cloning the repo and creating a conda environment with Python 3.7, dependencies may be installed through the repo's instructions under the "Env" heading, no modifications needed. We used the Yelp dataset, provided freely on the Yelp website, and when uncompressed, is a directory of json files in the format `yelp_academic_dataset_CATEGORY.json`.

For the author's preprocessing instructions on their repo to work, we made the following modifications:
• Changed the format of all the json files to `CATEGORY.json` instead
• Line 24 of `preprocess/yelp_preprocess.py` was changed from `int(datetime(strptime()))` to `int(datetime.datetime(strptime()))`
• Made new directory `OUTPUT_DIR`

The preprocessing script effectively reads in the 52268 restaurants and associated reviews from the dataset and computes the average of the first 50 reviews. The dataset split used is 64%/16%/20% train/valid/test, so in terms of restaurants, 33452/8363/10453.

For training the Longformer model, one has to run the command in the repo with sudo, after changing `OUTPUT_DIR`, `YELP_OUTPUT_DIR`, `CAHCE_DIR` to `OUTPUT_DIR/transformers`, `OUTPUT_DIR`, `OUTPUT_DIR/transformers_cache` respectively, in `train_transformer.sh`.

For running DecSum, go to `sentence_select.sh` and change `OUTPUT_DIR` to `OUTPUT_DIR` and `MODEL_PATH` to the path of the `.ckpt` file outputted by training Longformer. Similarly, correct the paths of `OUTPUT_DIR` and `MODEL_PATH` in `single_sentence_score.sh` before running it to get the decision scores for individual sentences.

To get the Wasserstein distance between the predictions using full reviews and summaries generated by DecSum, comment out the segment of code in `sentence_select.sh` corresponding to every sentence, and uncomment the segment of code corresponding to all input. Then rename the filename of the result of `single_sentence_score.sh` to `all.csv`. Store the DecSum summaries and `all.csv` in some input directory, and run `python wasserstein.py PATH/TO/INPUT/DIRECTORY` from our Github repo, in which the above errors are also fixed.

The additional experiments we took beyond the scope of the paper may be run by modifying the parameters in `sentence_select.sh` (e.g. `num_sentences`, `num_review`) and `train_transformer.sh` (e.g. `max_seq_length`). The script that generates the pdf plot in our reproduction of the paper may also be found on our repo, and can be run with `python pdf.py`.

3.3 Hyperparameters

Longformer takes 102M parameters with half precision and makes use of the Huggingface transformers package [3] and the AdamW optimizer [4], with learning rate $5 \cdot 10^{-5}$ and a linear warmup of 500 steps for the learning rate. The model is trained for 3 epochs where the batch size is 4 and the maximum input token length is 3,000. There are 12 hidden layers, each of size 768, and there are 12 attention heads with attention window 512. The dropout is 0.1.

One hyperparameter group we used was what values to use for the three desiderata (decision faithfulness, decision representativeness, textual non-redundancy) when implementing DecSum, i.e. what values in $\{0, 1\}$ to use for the DecSum hyperparameter group (α, β, γ). We ran our experiments using the same as the ones given in Table 1 of their paper.

3.4 Experimental Setup and Computational Requirements

The authors preprocess their data and train their model on a single RTX 3090 setup. The performance of Tesla V100s lags behind an RTX 3090, so we expect that time spent on training and processing the data to be noticeably more than that of the authors. For reference, the authors spent an hour per epoch on fine-tuning the regression models with Longformer before DecSum was ran, and less than an hour on DecSum itself for every trial. Perhaps only 8 GB RAM is necessary, as the yelp dataset itself is only 13 GB. Similarly, the disk memory usage should thus be slightly above 13 GB, perhaps around 14 GB. Using a 2xTesla V100 GPU setup, we estimate the runtime of Longformer to be 4 hours and maximum 4 hours for DecSum for every (α, β, γ) permutation of the trial as well.

In reality, training Longformer took 4/3 hours for each of its 3 epochs, for a total of 4 hours. For DecSum, each trial took much longer than expected, for a total of 14 hours—10 hours for training and 4 hours for the prediction and evaluation on the test set. There were a total of 15 trials we ran, 7 of which were in the paper and 8 that were additional experiments we took beyond the scope of the paper to see the effect of certain hyperparameters. Initially, we tried

running the models with 8 GB of RAM, however this led to it crashing, so we ultimately used 16 GB of RAM to run the experiments instead. We used 15 GB of disk memory: 13 GB from the yelp dataset and the rest from the transformer and pretrained models used in the scripts wrote by the authors in the pipeline.

4 Results

4.1 Result of Reproducing Experiments in Paper

We are only testing the first and only claim: that DecSum substantially outperforms text-only summarization methods and model-based explanation methods in terms of decision faithfulness and representativeness.

4.1.1 Decision Faithfulness

We are able to get similar results as the authors did, and the claim being tested was indeed true for us as well. As shown in Table 1, MSE of DecSum (1,1,1) is quite close to the result using all texts, and much better as compared to the Text-only summarization method and model-based explanation methods. Consistent with the paper, out of three hyperparameters of DecSum, the hyperparameter of decision faithfulness is the most important one for the MSE metric, while two other hyperparameters have less impact on the MSE of the results.

Note that in our DecSum experiments, (0,1,0) has the worst model performance, with MSE of 0.5718. The paper, however, shows the DecSum experiment (0,0,1) has the worst model performance, with an MSE of 0.565. This may be related to model randomness. It may also show that decision representativeness and textual non-redundancy by themselves are not good and consistent metrics for generating summaries for decision tasks.

Table 1: MSE of Model Predictions Based on Summaries of Different Methods

Method	**MSE**
All	0.1336
Text-only summarization method - random	0.475
Text-only summarization method - BART	0.502
Text-only summarization method - PreSumm	0.478
Model-based explanation methods - IG	0.565
Model-based explanation methods - Attention	0.715
DecSum (1,1,1)	0.1338
DecSum (1,1,0)	0.1336
DecSum (1,0,1)	0.1339
DecSum (0,1,1)	0.3196
DecSum (1,0,0)	0.1336
DecSum (0,1,0)	0.5718
DecSum (0,0,1)	0.2362

4.1.2 Decision Representativeness

Table 2 shows the Wasserstein distance between model predictions of the selected sentences with those of all the sentences. In line with the paper, DecSum has much lower values of Wasserstein distance (i.e., better decision representativeness) than the text-only summarization method and model-based explanation methods.

Following the paper, we plot the density distribution of model predictions to investigate model decision representativeness. Figure 2 shows the result for one selected restaurant. As shown in Figure 2, the DecSum is able to capture a sentence with a low predicted rating from reviews of IHOP, while PreSumm predicts ratings all above 3. Such a result also indicates DecSum has better decision representativeness, as compared to other summary methods (i.e. Text-only summarization and Model-based explanation methods). Dots on the plot represent selected sentences on the distribution of model predictions. Please note that in our pdf plot we arbitrarily chose the sentences to be represented by the circles and stars, as we did not understand how the authors chose their selected sentences in their density distribution plot from the paper.

Table 2: Wasserstein Distance Between Model Predictions of the Selected Sentences With Those of All the Sentences

Method	Wasserstein Distance
Text-only summarization method - random	0.398
Text-only summarization method - BART	0.423
Text-only summarization method - PreSumm	0.412
Model-based explanation methods - IG	0.62
Model-based explanation methods - Attention	0.64
DecSum (1,1,1)	0.06525
DecSum (1,1,0)	0.38331
DecSum (1,0,1)	0.00048
DecSum (0,1,1)	0.00257
DecSum (1,0,0)	0.00047
DecSum (0,1,0)	0.00257
DecSum (0,0,1)	0.14236

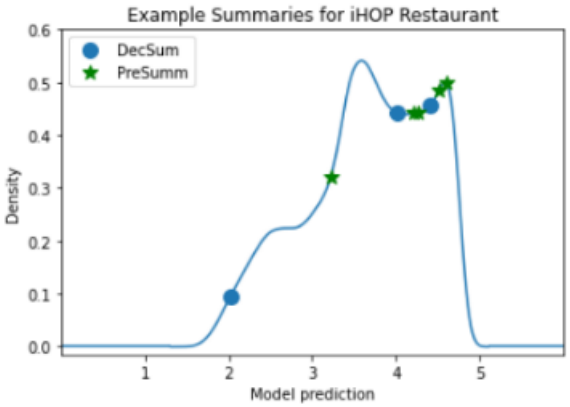

Figure 2: Example PreSumm and DecSum Summaries for the iHOP restaurant

4.2 Additional Results not Present in the Original Paper

In this subsection, we document the additional experiments we carried out to investigate the sensitivity of DecSum's performance to the hyperparameters and ablations such as dataset size.

4.2.1 Data Related - Training Data Size

In the original paper, %64 of the yelp dataset is used to train the model. Here, we tried halving the training data size to see its impact. As Table 3 shows, halving the training data only decreases the performance by a bit as compared to the result using the full training data.

Table 3: MSE of Model Predictions for [Default, Half] Training Data Size

Experiments	MSE
DecSum (1,1,1) with default training data	0.1338
DecSum (1,1,1) with half of training data	0.2246

4.2.2 Data Related - Number of Reviews for Computing Average Rating

In the original paper, the authors choose 50 as the number of reviews for computing average rating. Here we tried 30 and 70 as the number of reviews for computing average rating to see its impact. As expected, using a smaller number of reviews could improve the model performance, but as indicated in Table 4, the performance of DecSum is not sensitive to this hyperparameter.

Table 4: MSE of Model Predictions with [default, 30, 70] Reviews

Method	MSE
DecSum (1,1,1) with 50 reviews (default)	0.1338
DecSum (1,1,1) with 30 reviews	0.1186
DecSum (1,1,1) with 70 reviews	0.1369

4.2.3 Transformer Related - Max Sequence Length

The authors choose 3000 as the max sequence length when training the Longformer. Here we tried 1000 and 4000 as the max sequence length to see its impact. We could see the performance is much worse if we use the max sequence length 4000 for Longformer, which may be because it takes more epochs to converge for the longer sequence length. While if we choose the 1000 max sequence length for Longformer, the performance decreases only a bit.

Table 5: MSE of Model Predictions for [default, 1000, 4000] Max Sequence Length for Longformer

Experiments	MSE
DecSum (1,1,1) with max seq length 3000 (default)	0.1338
DecSum (1,1,1) with max seq length 1000	0.1456
DecSum (1,1,1) with max seq length 4000	0.3926

4.2.4 DecSum Related - Summary Length

In the paper, the authors choose 50 tokens as the summary length. Here we also tried using 10, 30, 70 tokens as the summary length to see its impact on the prediction results. As shown in Table 6, the model with higher summary length tends to have better prediction (lower MSE and higher accuracy). Such results are in our expectations, given that with a longer summary length, less information has been truncated and thus the model is able to make better predictions. But the impact of summary length is generally small and may not be statistically significant. Note that, using 70 summary length, the training time is about 1.5 times of the default version (i.e., 50 summary length).

Table 6: MSE of Model Predictions for [default, 30, 70] Summary Length

Experiments	MSE
DecSum (1,1,1) with 50 summary length (default)	0.1338
DecSum (1,1,1) with 30 summary length	0.1365
DecSum (1,1,1) with 70 summary length	0.1335

5 Discussion

Generally, we are able to reproduce the paper result. As Table 1 shows, the results of different model simulations run by ourselves are close to the paper's result, and shows that DecSum outperforms the text-only summarization methods and model-based explanation methods in decision faithfulness and representativeness.

5.1 What was Easy

The authors have uploaded their code in the Github, and the instructions to run the codes are clear. Therefore, it is relatively easy for us to run the experiments and reproduce the paper's results. The data (i.e., Yelp dataset) used in the paper is open-access and so we were able to easily obtain the required data for the experiments.

5.2 What was Difficult

In the paper's Github page, the csv files under the directory of preprocess were meant to specify the IDs of the data in the train, dev, and test sets, however, they are incorrect because they do not correspond with actual IDs. At the beginning of the project, it took us some time to figure out the particular bugs in the authors' code. The authors also used human evaluation (i.e. crowdsourcing human evaluation of model predictions using Amazon Mechanical Turk) in their paper to test the performance of DecSum, but it was not possible for us to employ this type of evaluation in our reproduction of the paper. Each DecSum trial took around 14 hours to run. In order to reproduce the paper's result, we have carried out more than 15 DecSum experiments, which took much time given the limited computing resources we had.

5.3 Recommendations for Reproducibility

The scripts provided on the repo had code errors that we had to fix before it ran properly. The GitHub documentation was incomplete (the README.md is empty for sections "Baseline methods" and "Generating Experiment Plots"). The heatmap in the paper and the pdf were hard to interpret and therefore not ideal when we were trying to replicate them.

Communication with Original Authors

We had no correspondence with the original authors.

References

[1] Chao-Chun Hsu and Chenhao Tan. Decision-focused summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 117–132, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[2] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[3] 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. 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, October 2020. Association for Computational Linguistics.

[4] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017.