Convex Aggregation for Opinion Summarization

Hayate Iso ♠ Xiaolan Wang ♠
Yoshihiko Suhara ♠ Stefanos Angelidis ♠ Wang-Chiew Tan ♠*

♠ Megagon Labs ♣ University of Edinburgh ♢ Facebook AI
hayate@megagon.ai, xialan@megagon.ai,
yoshi@megagon.ai, s.angelidis@ed.ac.uk, wangchiew@fb.com

Abstract

Recent advances in text autoencoders have significantly improved the quality of the latent space, which enables models to generate grammatical and consistent text from aggregated latent vectors. As a successful application of this property, unsupervised opinion summarization models generate a summary by decoding the aggregated latent vectors of inputs. More specifically, they perform the aggregation via simple average. However, little is known about how the vector aggregation step affects the generation quality. In this study, we revisit the commonly used simple average approach by examining the latent space and generated summaries. We found that text autoencoders tend to generate overly generic summaries from simply averaged latent vectors due to an unexpected $L_2$-norm shrinkage in the aggregated latent vectors, which we refer to as summary vector degeneration.

To overcome this issue, we develop C\textit{OOP}, a latent vector aggregation framework. In essence, C\textit{OOP} considers convex combinations of the latent vectors of input reviews for better summary generation. More specifically, C\textit{OOP} searches input combinations for the latent vector aggregation using input-output word overlap. Experimental results show that C\textit{OOP} successfully alleviates the summary vector degeneration issue and establishes new state-of-the-art performance on two opinion summarization benchmarks. Code is available at https://github.com/megagonlabs/coop.

1 Introduction

The unprecedented growth of online review platforms and the recent success of neural summarization techniques (Cheng and Lapata, 2016; See et al., 2017; Liu and Lapata, 2019), spurred significant interest in research on multi-document opinion summarization (Angelidis and Lapata, 2018; Chu and Liu, 2019; Bražinskas et al., 2020; Suhara et al., 2020; Amplayo and Lapata, 2020; Amplayo et al., 2021). The goal of multi-document opinion summarization is to generate a summary that represents salient opinions in the input reviews.

Research on multi-document opinion summarization is challenging because of the lack of gold-standard summaries, which are difficult to collect at scale. This is in contrast to single-document summarization, where there exists an abundant annotated datasets (Sandhaus, 2008; Hermann et al., 2015; Rush et al., 2015; Narayan et al., 2018). Consequently, the primary approach is to employ text autoencoders for unsupervised opinion summarization (Chu and Liu, 2019; Bražinskas et al., 2020). Text autoencoders, especially variational autoencoders (VAEs), are known for the ability to generate grammatical and consistent text by aggregating multiple latent vectors (Bowman et al., 2016). Unsupervised opinion summarization models leverage this property to generate a summary by first aggregating the latent vectors of input reviews via simple average, and then decoding the summary from the aggregated vector.

However, it has not been verified if the simple average is the best choice for summary generation. Furthermore, little is known about the relationship between the latent vector and the generation quality. In this paper, we report that text autoencoder models with the simple average vector aggregation tend to generate overly generic summaries, which we refer to as summary vector degeneration. For example, as shown in Figure 1, with simply averaged latent vectors, the generated summaries of two distinct entities are almost identical. We further discovered two factors that cause summary vector degeneration: (1) simply averaged latent vectors cause unexpected $L_2$-norm shrinkage, and (2) latent vectors with smaller $L_2$-norm are decoded into less informative summaries (e.g., contain only general information.)

To address the summary vector degeneration issue, we develop C\textit{OOP}, a latent vector aggregation framework. In essence, C\textit{OOP} considers convex combinations of the latent vectors of input reviews for better summary generation. More specifi-
This optimization strategy helps the model generate views, thus improving the quality of summarization better aggregated vector to generate a more specific summary. Opinion summarization uses the simple average of input review vectors for unsupervised opinion summarization models.

Our contributions are summarized as follows:

- We report that the commonly used simple average vector aggregation method causes summary vector degeneration, which makes the decoder generate less informative and overly generic summaries.
- We formalize latent vector aggregation as an optimization problem, which considers the convex combination of input review latent vectors. We propose a solution, COOP, to approximate the optimal latent vector with linear time complexity. To the best of our knowledge, this is the first work that optimizes latent vector aggregation for opinion summarization.
- We conduct comparative experiments against existing methods (Chu and Liu, 2019; Bražinskas et al., 2020), which implement more sophisticated techniques. Our experiments demonstrate that by coupling with COOP, two opinion summarization models (BiMEANVAE and Optimus) establish new state-of-the-art performance on both Yelp and Amazon datasets.

2 Preliminaries

Let us denote $\mathcal{R} = \{x_i\}_{i=1}^{\left|\mathcal{R}\right|}$ as a dataset of customer reviews of the same domain (e.g., restaurant or product), where each review is a sequence of words $x = (x_1, \ldots, x_{\left|x\right|})$ in the text space $\mathcal{X}$. Given an entity $e$ and its reviews $\mathcal{R}_e \subseteq \mathcal{R}$, the goal of the multi-document opinion summarization task is to generate an abstractive summary $s_e$ such that the salient opinions in $\mathcal{R}_e$ are included.

2.1 Unsupervised Opinion Summarization

Existing unsupervised opinion summarization models (Chu and Liu, 2019; Bražinskas et al., 2020) use the autoencoder, where an encoder $E : \mathcal{X} \rightarrow \mathcal{Z}$ mapping from the text space $\mathcal{X}$ to latent space $\mathcal{Z}$, and a decoder $G : \mathcal{Z} \rightarrow \mathcal{X}$ that generates texts from latent vectors.

Encoder $E$: Given an entity $e$ and its reviews $\mathcal{R}_e$, the encoder $E$ essentially maps every review $x_i \in \mathcal{R}_e$ into the latent space: $z_i = E(x_i)$, where $z_i$ is the latent vector of review $x_i$.

Decoder $G$: The other core component is the decoder $G$, which generate a new text $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_{\left|\hat{x}\right|})$ from a given latent vector $z$: $\hat{x} = G(z)$.

Training: At the training phase, the autoencoder model is trained to generate the review. While various training methods have been proposed, the simplest approach is aimed to reconstruct the input review from the corresponding latent vector.

Generation: At the generation phase, given a set of input review latent vectors $Z_e = \{z_1, \ldots, z_{\left|\mathcal{R}_e\right|}\}$, existing opinion summarization models use simple average to create the latent vector of the summary (summary vector) $z^\text{avg}_\text{summary} = \frac{1}{\left|\mathcal{R}_e\right|} \sum_{i=1}^{\left|\mathcal{R}_e\right|} z_i$, which is then decoded into the summary. In this paper, our focus is to analyze and improve the latent vector aggregation for the summary.
2.2 Variational Autoencoders

In this study, we use variational autoencoders (VAEs) as the text autoencoder since it provides a smooth latent space, which allows to produce grammatical and consistent text from aggregated latent vectors (Kingma and Welling, 2014; Bowman et al., 2016). More specifically, we tested two VAE variations, namely BiMEANVAE and Optimus (Li et al., 2020). BiMEANVAE uses bidirectional LSTM as the encoder, LSTM as the decoder, and applies a mean pooling layer to the BiLSTM layer to obtain the latent vector. Optimus (Li et al., 2020) is a Transformer-based VAE model that uses BERT (Devlin et al., 2019) as the encoder and GPT-2 (Radford et al., 2019) as the decoder. Unlike MeanSum (Chu and Liu, 2019), both BiMEANVAE and Optimus do not use any additional objectives but the basic VAE objective (i.e., the reconstruction loss with KL regularization):

$$
\mathcal{L}(\theta, \phi) = \mathcal{L}_{\text{rec}} + \beta \mathcal{L}_{KL} \\
\mathcal{L}_{\text{rec}}(\theta, \phi) = -\mathbb{E}_{q_{\phi}(z|x)}[\log p_{\theta}(x|z)] \\
\mathcal{L}_{KL}(\phi) = D_{KL}(q_{\phi}(z|x)\|p_{\theta}(z)),
$$

where $\phi$ and $\theta$ are the parameters of the encoder $E$ and decoder $G$. $\beta$ is a hyper-parameter that controls the strength of the KL regularization $\mathcal{L}_{KL}(\phi)$. We choose the standard Gaussian distribution $\mathcal{N}(0, I)$ as the prior distribution $p_{\theta}(z)$.

3 Revisiting Simple Average Approach

In this section, we revisit the commonly used simple average approach (SimpleAvg) and examine the relations between the aggregated latent vector and the quality of generated summaries.

Taking a simple average is an intuitive way to optimize the aggregated vector in the latent space since it minimizes the total distance between input latent vectors and the aggregated vector. Thus, it appears to be a reasonable design choice and has been adopted by multiple unsupervised opinion summarization models as de-facto standard.

However, we find that only considering the total distance between the input and the aggregated latent vectors does not always render high-quality summaries. This is because SimpleAvg is completely ignorant of the decoder performance and the resulting generation. In fact, we observe that SimpleAvg tends to produce overly generic summaries (as shown in Figure 1), which we refer to as summary vector degeneration.

To gain a better understanding of the summary vector degeneration problem, we conducted further analysis and discovered two factors that cause this problem: (1) simply averaging input latent vectors causes $L_2$-norm shrinkage, and (2) latent vectors with smaller $L_2$-norm tend to be decoded into less informative generations.

3.1 $L_2$-norm Shrinkage in Latent Space

To understand how simply averaged latent vectors distribute in the latent space, we compared the $L_2$-norm of the latent vectors of input reviews and summary vectors created by SimpleAvg. We conducted analysis using BiMEANVAE on two review datasets, Yelp and Amazon.

As shown in Figure 2, the average $L_2$-norm of the summary vectors significantly shrinks from 9.97 to 4.10 on Yelp (10.15 to 4.17 on Amazon) as the number of input reviews is increased from 1 (i.e., individual reviews) to 8. The results show that simply averaging multiple latent vectors can cause $L_2$-norm shrinkage of the summary vector. As we expect each dimension in the latent space to represent a distinct semantics, $L_2$-norm shrinkage may cause some information loss in the summary vector.

3.2 Summary Vector Degeneration

To investigate the effect of $L_2$-norm shrinkage in the latent space, we further analyzed the quality of generated text for each latent vector and conducted correlation analysis against the $L_2$-norm. We used two metrics to assess the quality of generated text: (a) text length and (b) information amount. For the information amount, we trained an autoregressive model (RNN-LM) on each dataset and used negative log probabilities of generated summaries (i.e., a higher value means more amount of infor-
Figure 3: Correlation analysis of the $L_2$-norm of latent vectors $\|z\|$ and the generated text quality: (a) text length and (b) information amount.

Figure 4: COOP searches convex combinations of the latent vectors of input reviews based on the *input-output word overlap* between a generated summary and input reviews. $\times$ denotes the simply averaged vector.

4 Convex Aggregation in Latent Space

As discussed above, there are two limitations with the de-facto standard SimpleAvg. First, it causes summary vector degeneration. Second, it is ignorant of the decoder generation (in the text space $X$) for a summary vector (in the latent space $Z$).

To address the issues, we consider an optimization problem that searches for the best combination of the latent vectors of input reviews that maximizes the *alignment* between input reviews and generated summaries. We restrict the search space to the convex combinations of input review representations, so the contribution of each input review is always zero or positive. This is based on the assumption that each review in the input set should be either ignored or reflected. Hence, we refer to the latent representation aggregation problem as convex aggregation.

4.1 COOP: Convex Aggregation for Opinion Summarization

We develop a latent vector aggregation framework COOP to solve the convex aggregation problem in Figure 4. COOP optimizes for the *input-output word overlap* between a generated summary and the input reviews:

$$\text{maximize}_{z} \quad \text{Overlap}(R_e, G(z))$$

subject to

$$z = \sum_{i=1}^{|R_e|} w_i z_i,$$

$$\sum_{i=1}^{|R_e|} w_i = 1, \forall w_i \in \mathbb{R}^+.$$  

The input-output word overlap (Overlap) evaluates the consistency between input reviews and a generated summary, and it can naturally penalize hallucinated generations. Specifically, we use the ROUGE-1 F1 score as the input-output word overlap metric (Lin, 2004). Note that the method does not use gold-standard summaries or any information in the test set but uses the input reviews for calculating word overlap information.

4.2 Search Space

Following the intuition that some input reviews are useful and others are not, we narrow down the search space to the power set of an input review.

---

1. Training details are in Appendix.

2. We also tested other ROUGE scores such as ROUGE-2/L in the preliminary experiments and found that ROUGE-1 (i.e., word overlap) works most robustly, so we decided to use the most straightforward metric.
We also report two types of oracle methods. **Oracle (single)**: This oracle method selects a single input review that takes the highest ROUGE-1 F1 score on the gold-standard summary. **Oracle (comb.)**: This oracle method selects the best set of input reviews from the power set $2^{R_e} \setminus \{\emptyset\}$ of input review set $R_e$, so that it achieves the highest ROUGE-1 F1 score on the gold-standard summary when BiMEANVAE is used as the summarization model. This can also be interpreted as the upper-bound performance of BiMEANVAE.

More details about our evaluation can be found in the Appendix.

### 5 Evaluation

**Dataset:** For our experiments, we used two publicly available datasets, **Yelp** (Chu and Liu, 2019) and **Amazon** (Bražinskas et al., 2020). Besides reviews used for training, these two datasets also contain gold-standard summaries for 200 and 60 sampled entities, respectively. For both datasets, the summaries are manually created from 8 input reviews, and we used the same dev/test split, 100/100 for **Yelp** and 28/32 for **Amazon**, released by their authors for our experiments.

**Experimental settings:** We used Adam optimizer (Kingma and Ba, 2015) with a linear scheduler, whose initial learning rate is set to $10^{-3}$ ($10^{-5}$) for BiMEANVAE (Optimus.). To mitigate the KL vanishing issue, we also applied KL annealing during the training (Kingma et al., 2016; Fu et al., 2019; Li et al., 2019).

For generation, we used beam search with a size of 4. In order to generate summary-like texts, we introduce a technique, *first-person pronoun blocking*, that prohibits to generate first-person pronouns (e.g., I, my, me) during summary generations. We report the ROUGE-1/2/L F1 scores for the automatic evaluation (Lin, 2004).³

**Comparative methods:** We compared our models (i.e., BiMEANVAE and Optimus with COOP) against state-of-the-art opinion summarization models that use SimpleAvg for latent vector aggregation, namely TextVAE (Bowman et al., 2016), MeanSum (Chu and Liu, 2019). We also coupled BiMEANVAE and Optimus with SimpleAvg to verify the effectiveness of COOP. In addition, we report the performance of other abstractive, extractive or weakly-supervised opinion summarization models.

Besides the unsupervised summarization models, we also report two types of oracle methods.

**Oracle (single):** This oracle method selects a single input review that takes the highest ROUGE-1 F1 score on the gold-standard summary.

**Oracle (comb.):** This oracle method selects the best set of input reviews from the power set $2^{R_e} \setminus \{\emptyset\}$ of input review set $R_e$, so that it achieves the highest ROUGE-1 F1 score on the gold-standard summary when BiMEANVAE is used as the summarization model. This can also be interpreted as the upper-bound performance of BiMEANVAE.

More details about our evaluation can be found in the Appendix.

### 5.1 Automatic Evaluation

As shown in Table 1, COOP is able to improve both summarization models, BiMEANVAE and Optimus, by a large margin. With COOP, BiMEANVAE and Optimus obtain the new state-of-the-art performance on both benchmark datasets. Besides the summarization performance, we also show the model sizes in Table 1. Note that BiMEANVAE performs competitively well against Optimus, which is trained on top of large pre-trained language models and has approximately 20x more parameters than BiMEANVAE. We believe this is due to the simple yet important configuration in the model architecture, which uses a BiLSTM encoder (vs. unidirectional LSTM in TextVAE) and a mean-pooling layer (vs. last hidden state).

Meanwhile, BiMEANVAE and Optimus with COOP outperform Oracle (single), which selects the single review that takes the highest ROUGE score. The results indicate that our aggregation framework takes the quality of unsupervised multi-document opinion summarization to the next stage.

It is worthwhile to note that both VAE variations with the conventional simple average aggregation competitively perform well against the state-of-the-art performance on opinion summarization benchmarks as shown in Table 1. In contrast to previous study (Bražinskas et al., 2020), which showed that text VAE performs poorly on the opinion summarization, our modified configuration makes BiMEANVAE a competitive baseline for the task.

### 5.2 Human Evaluation

We conducted human evaluation to assess the quality of generated summaries. More specifically, we collected the generated summaries for entities in the **Yelp** test set with four different models, COOP (BiMEANVAE), SimpleAvg (BiMEANVAE), CopyCat and PlanSum. Then,
Table 1: ROUGE scores on the benchmarks. Bold-faced and underlined denote the best and second-best scores respectively. COOP significantly improves the performance of two summarization models, BiMEANVAE and Optimus, and achieves new state-of-the-art performance on both of the benchmark datasets. † means the results are copied from Bražinskas et al. (2020), ‡ from Amplayo et al. (2021), ‡ from Wang et al. (2020), ♯ from Angelidis et al. (2021), and ♮ from Suhara et al. (2020). Note that this study classifies OpinionDigest and PlanSum as weakly-supervised summarizers since they use additional information other than review text.

Table 2: Human evaluation on Yelp dataset. COOP outperforms the other baseline models on both informativeness (Info) and content support.

We asked three human judges to evaluate the summaries with two criteria: informativeness and content support.

We first asked human judges to evaluate the informativeness of the generated summaries by the Best-Worst Scaling (Louviere et al., 2015), which scores each summarization method with values ranging from -100 (unanimously worst) to +100 (unanimously best). We then asked human judges to evaluate the content support of the generated summaries. For each sentence in the generated summary, the judges chose an option from (a) fully supported, (b) partially supported, or (c) not.

We present the human evaluation results in Table 2. As shown, summaries generated by COOP are more informative than SimpleAvg and the other baseline models. Meanwhile, COOP also behaves well on content support as it generates more sentences with full/partial content support than the other methods. These results indicate that COOP is able to generate more informative and less hallucinated summaries. Combined with the automatic evaluation results, we conclude that COOP meaningfully improves the quality of summarization generation.

6 Analysis

In this section, we conduct a series of additional analysis to verify the effectiveness and efficiency of COOP. We also provide detailed descriptions of the setups and additional analysis in the Appendix.

4BiMEANVAE shows robust performance even combined with SimpleAvg.
6.1 Summary Vector Analysis

Does COOP avoid $L_2$-norm shrinkage? To verify if COOP alleviates the summary vector degeneration issue, we compare the $L_2$-norm distributions of the summary vectors by SimpleAvg, COOP, Oracle (comb.), and the original reviews (None.) We used BiMEANVAE for SimpleAvg and COOP as the base models.

Figure 5 shows COOP does not show severe $L_2$-norm shrinkage compared to SimpleAvg. However, the distributions of any aggregation methods, including Oracle, show smaller means of $L_2$-norm compared to individual reviews. This is expected, as customer reviews often contain irrelevant (and specific) information that is not suitable for summaries (e.g., personal experience.) Therefore, just preserving the $L_2$-norm of input latent vectors does not necessarily lead to high-quality summary vectors.

We confirm that COOP has a similar distribution to that of Oracle (comb.), which achieves the upper bound performance of COOP. The results indicate that COOP successfully excludes input reviews that contain too much irrelevant information, so it can create high-quality summary vectors without accessing any gold-standard summaries.

**How good is COOP’s summary vector?** We verify how good COOP’s summary vector selections are with respect to summary generation quality. Specifically, we sorted all summary vector candidates in power set $2^\mathcal{R}_e \setminus \emptyset$ by the ROUGE-1 score using the generated summary and gold-standard summaries. By doing so, we can use the position of COOP’s selection to evaluate the summary vector quality using ranking metrics. We iterated the process for each entity $e$ and used two metrics, namely Mean Reciprocal Rank (MRR) and normalized discounted cumulative gain (nDCG) (Schütze et al., 2008), for the evaluation.

### Approximate Search

While we further narrowed down the search space of COOP into power set in §4.2, the brute-force search becomes intractable for a larger number of input reviews $N$. Therefore, we tested approximate search algorithms for efficient and effective search.

The simplest solution is the greedy search, which begins from a single review and progressively adds a review that offers the highest gain in the objective value. The greedy search can be easily generalized to beam search, which stores $k$ candidates for each step. We also consider the “inverse” version of the search algorithms, which begins from all input reviews and removes a review that offers the highest gain by excluding the input review step by step.
Table 4: Spearman correlation values between the input-output word overlap and ROUGE F1 scores on the test set.

|        | ROUGE-1 | ROUGE-2 | ROUGE-L |
|--------|---------|---------|---------|
| Yelp   | .3758   | .2560   | .2266   |
| Amazon | .5199   | .3903   | .4816   |

We refer to the original and the inverse versions as forward and backward, respectively.

Figure 6 reports the ROUGE-1 performance of BiMEANVAE with COOP using approximate search on the Yelp and Amazon datasets. The greedy search (beam size = 1) still outperforms the SimpleAvg baseline, and increasing the beam size further improves the performance of both search methods. Thus, we confirm that COOP framework still can provide significant gains by using approximate search when the input size is too large to conduct the exact search for the entire space.

6.3 Input-output Word Overlap

To investigate the effectiveness of the input-output word overlap as an intrinsic metric, we analyze the Spearman’s rank correlation between the input-output word overlap and the ROUGE scores on the gold-standard summaries for each summary vector in the search space $2^R \setminus \{\emptyset\}$.

Interestingly, as shown in Table 4, the two datasets show different trends. In contrast to the Amazon dataset, where the input-output word overlap shows high correlation values against the ROUGE scores, the correlation values on the Yelp dataset are much smaller. The results confirm that the effect of the input-output word overlap is not just because it is correlated with ROUGE scores between a generated summary and the gold-standard summaries.

6.4 Qualitative analysis

Figure 7 shows an example of generated summaries using BiMEANVAE with SimpleAvg and COOP for reviews about a restaurant in the Yelp dataset. This example shows that the summary generated from the SimpleAvg $z_{avg}^{summary}$ contains general opinions (e.g., “the food is delicious.”). In contrast, COOP effectively chose a subset of reviews to generate a summary vector $z_{coop}^{summary}$, which was decoded into a more specific summary.

7 Related Work

Opinion Summarization Multi-document opinion summarization uses the unsupervised approach as it is difficult to collect a sufficient amount of gold-standard summaries for training. Previously, the common approach was extractive summarization, which selects sentences based on the centrality criteria (Erkan and Radev, 2004). Due to the recent advances in neural network models, unsupervised abstractive summarization techniques have become the mainstream for opinion summarization.

Most abstractive unsupervised opinion summarization techniques use a two-stage approach that trains an encoder-decoder model based on the reconstruction objective and generates a summary from the average latent vectors of input reviews using the trained model (Chu and Liu, 2019). Amplayo and Lapata (2020) and Amplayo et al. (2021) have expanded the approach by creating pseudo review-pairs to train a summarization model.

Our study revisits this two-stage approach and develop a latent vector aggregation framework, which can be combined with a variety of opinion summarization models.

Variational Auto-Encoder The VAE is a variant of auto-encoder that learns a regularized latent space. The text VAE (Bowman et al., 2016), VAE with autoregressive decoder, has been commonly used for various NLP tasks including text generation (Ye et al., 2020), paraphrase generation (Bao et al., 2019) and text style transfer (Hu et al., 2017; John et al., 2019).

In contrast to the success of the text VAE in NLP tasks, an earlier attempt for using the text VAE for opinion summarization was not successful; Bražinskis et al. (2020) showed that the performance of text VAE was significantly lower than the other baselines. In this paper, we devise BiMEANVAE, a simple variant of the text VAE, which performs competitively well against the previous state-of-the-art methods even when coupled with SimpleAvg.

Recently, a more expressive text VAE model Optimus (Li et al., 2020), which is built on top of pre-trained BERT and GPT-2 models, has been developed. The model was originally developed for sentence generation tasks, and we are the first to combine it with a latent vector aggregation framework for unsupervised opinion summarization tasks.
8 Conclusions

In this paper, we revisit the unsupervised opinion summarization architecture and show that the commonly used simple average aggregation is suboptimal since it causes summary vector degeneration and does not consider the difference in the quality of input reviews or decoder generations.

To address the issues, we develop a latent vector aggregation framework COOP, which searches convex combinations of the latent vectors of input reviews based on the word overlap between input reviews and a generated summary. The strategy helps the model generate summaries that are more consistent with input reviews. To the best of our knowledge, COOP is the first framework that tackles the latent vector aggregation problem for opinion summarization.

Our experiments have shown that with COOP, two summarization models, BiMEANVAE and Optimus, establish new state-of-the-art performance on two opinion summarization benchmarks. The results demonstrate that our aggregation framework takes the quality of unsupervised opinion summarization to the next stage.

Acknowledgements

We thank Arthur Bražinskas for his valuable feedback and correcting the description about CopyCat.

References

Reinald Kim Amplayo, Stefanos Angelidis, and Mirella Lapata. 2021. Unsupervised opinion summarization with content planning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 12489–12497.

Reinald Kim Amplayo and Mirella Lapata. 2020. Unsupervised opinion summarization with noising and denoising. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1934–1945, Online. Association for Computational Linguistics.

Stefanos Angelidis, Reinald Kim Amplayo, Yoshihiko Suhara, Xiaolan Wang, and Mirella Lapata. 2021. Extractive Opinion Summarization in Quan-
tized Transformer Spaces. Transactions of the Association for Computational Linguistics, 9:277–293.

Stefanos Angelidis and Mirella Lapata. 2018. Summarizing opinions: Aspect extraction meets sentiment prediction and they are both weakly supervised. In Proceedings of the 2018 Conference onEmpirical Methods in Natural Language Processing, pages 3675–3686, Brussels, Belgium. Association for Computational Linguistics.

Charles Audet and Warren Hare. 2017. Derivative-Free and Blackbox Optimization. Springer.

Yu Bao, Hao Zhou, Shujian Huang, Lei Li, Lili Mou, Olga Vechtomova, Xin-yu Dai, and Jiajun Chen. 2019. Generating sentences from disentangled syntactic and semantic spaces. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 6008–6019, Florence, Italy. Association for Computational Linguistics.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Arthur Bražinskas, Mirella Lapata, and Ivan Titov. 2020. Unsupervised opinion summarization as copycat-review generation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5151–5169, Online. Association for Computational Linguistics.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Jennifer C. Lai, and Robert L. Mercer. 1992. An estimate of an upper bound for the entropy of English. Computational Linguistics, 18(1):31–40.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 484–494, Berlin, Germany. Association for Computational Linguistics.

Eric Chu and Peter Liu. 2019. MeanSum: A Neural Model for Unsupervised Multi-Document Abstractive Summarization. In Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pages 1223–1232, Long Beach, California, USA. PMLR.

William G Cochran. 1954. The Combination of Estimates from Different Experiments. Biometrics, 10(1):101–129.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. 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 2185–2194, Hong Kong, China. Association for Computational Linguistics.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. Journal of artificial intelligence research, 22:457–479.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. 2019. Cyclical annealing schedule: A simple approach to mitigating KL vanishing. 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 240–250, Minneapolis, Minnesota. Association for Computational Linguistics.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In proceedings of the 25th international conference on world wide web, pages 507–517.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, page 1693–1701, Cambridge, MA, USA. MIT Press.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In International Conference on Machine Learning, pages 1587–1596. PMLR.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. In International Conference on Learning Representations, ICLR.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 424–434, Florence, Italy. Association for Computational Linguistics.
summarization. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 435–445, 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.

Rong Ye, Wenxian Shi, Hao Zhou, Zhongyu Wei, and Lei Li. 2020. Variational Template Machine for Data-to-Text Generation. In International Conference on Learning Representations.
A Dataset Preparation

Source Dataset: For the Yelp dataset, we used reviews provided in the Yelp Open Dataset. For Amazon dataset, we used reviews provided in the Amazon product review dataset (He and McAuley, 2016), and we select 4 categories: Electronics; Clothing, Shoes and Jewelry, Home and Kitchen; Health and Personal Care.

Preprocessing: We restricted the character set to be ASCII printable for the experiments. We preprocessed the datasets by excluding non-ASCII characters from the reviews and by removing accents from accented characters.

Tokenizer: For BiMEANVAE, we used sentencepiece (Kudo and Richardson, 2018) to train a BPE tokenizer with a vocabulary size of 32k, a character coverage of 100%, and a max sentence length of 8,192.

For Optimus, we used the pre-trained tokenizers provided by transformers (Wolf et al., 2020). Since Optimus uses different models for the encoder and decoder, we used different tokenizers for the encoder (bert-base-cased) and the decoder (gpt2), respectively.

Training data: We used pre-defined training sets of Yelp and Amazon with additional filtering. We filtered out reviews that consist of more than 128 tokens after tokenization by the BPE tokenizer trained for BiMEANVAE. As a result, the training sets contain 3.8 million and 13.3 million reviews in Yelp and Amazon respectively. We further excluded entities that have less than 10 reviews. The basic statistics of the training data after those filtering steps are shown in Table 5.

|        | Yelp  | Amazon |
|--------|-------|--------|
| # of entity | 75,988 | 244,652 |
| # of reviews | 4,658,968 | 13,053,202 |

Table 5: Statistics of the filtered training data.

B Revisiting Simple Average Approach

B.1 RNN-LM model for information amount

We trained single-layer RNN-LMs on Yelp and Amazon datasets respectively, in 100k steps with a batch size of 256. The embedding size and the hidden size are set to 512 and the output vocabulary layer is tied with the input embedding layer (Press and Wolf, 2017; Inan et al., 2016).

B.2 Additional analysis on latent vector and input text quality

In §3, we investigated the relationship between $L_2$-norm of latent vectors and the generated text qualities, and found strong positive correlations. In addition to the generated reviews $\hat{x}$, we conduct the same analysis using input reviews $x$.

As shown in Figure 8, we confirm the same trends that $L_2$-norm of latent vectors is highly correlated with both metrics. Thus, we confirm that less (more) informative text tends to be embedded closer to (more distant from) the origin in the latent space.

![Figure 8: Illustrations of the relationships between $L_2$-norm of latent vectors and the input review quality: (a) text length and (b) the information content.](https://example.com/A.png)

C Evaluation

C.1 Training settings

Major hyper-parameters for training models are reported in Table 8 and 9 following the “Show-Your-Work” style suggested by Dodge et al. (2019).

Optimization: We used Adam optimizer (Kingma and Ba, 2015) with a linear scheduler that has warm-up steps. The initial learning rate was set...
to be $10^{-3}$ for BiMeanVAE and $10^{-5}$ for Optimus.

**KL annealing**: For the VAE training, we adopt the KL annealing to avoid the KL vanishing issue (Bowman et al., 2016). To be specific, we tested two KL annealing strategies to control the $\beta$ value, i.e., cyclical KL annealing (Fu et al., 2019) and pretrain-then-anneal with KL thresholding (aka FreeBits) (Kingma et al., 2016; Li et al., 2019).

The **cyclic KL annealing** repeats the monotonic annealing process of the $\beta$ parameter from 0 to 1 multiple times (Fu et al., 2019). The **pretrain-then-annealing approach** has two steps (Li et al., 2019). The first step pre-trains an autoencoder model with the $\beta$ parameter fixed to 0. The second step re-initializes the decoder parameter and trains the model with the $\beta$ parameter monotonically increased from 0 to 1.

In addition to the annealing schedule, we also searched a threshold hyper-parameter for the KL value to control the strength of the KL regularization (Kingma et al., 2016).

### C.2 Baseline Models

We considered multiple abstractive and extractive summarization models.

**TextVAE** (Bowman et al., 2016): A vanilla text VAE model that has a unidirectional LSTM layer and uses the last hidden state to calculate the parameters of the posterior distribution. The model was tested in Bražinskas et al. (2020) but performed poorly.

**MeanSum** (Chu and Liu, 2019): An unsupervised multi-document abstractive summarization method that minimizes a combination of the reconstruction and similarity loss.

**CopyCat** (Bražinskas et al., 2020): An unsupervised opinion summarization model. CopyCat incorporates an additional latent vector $c$ to model an entire review set $R_e$ in addition to latent vectors for individual reviews. This hierarchical modeling enables CopyCat to consider both global (entity-level) and local (review-level) information to calculate a latent representation.

### C.3 Performance on Development Set

We report the performance on the development set in Table 6. COOP consistently improve the performance on ROUGE scores (except for ROUGE-L on Yelp) on the development set.

### C.4 Baselines for summary vector degeneration

In this paper, we develop a latent vector aggregation framework based on the input-output word-overlap to address the summary vector degeneration problem. As alternative and reasonable solutions, we tested the following methods and confirm that none of them consistently outperforms SimpleAvg, as shown in Table 7.

**Extractive** This method uses an extractive summarization technique to select $k$ input reviews that best represent the input review set. In the analysis, we used LexRank and set $k = 4$, which was chosen based on the Oracle (comb.) performance on the dev set. We used the simple average for the latent representation aggregation.

**Inverse-Variance Weighting** Weighted average based on importance scores of input reviews is an alternative way for latent representation aggregation. Specifically, we consider the variance parame-
We describe the details of the ranking metrics used in §6.1.

D Analysis

D.1 Ranking Metrics

We describe the details of the ranking metrics used in §6.1.
Figure 10: The inference runtime of BiMEANVAE and Optimus with different batch sizes.

D.2 Approximate Search

In addition to the ROUGE-1 scores shown in the main paper, we show the approximated search performance for ROUGE-2/L scores in Figure 9. We observe that the ROUGE-2 score shows that backward search is better for the Yelp dataset and forward search is better for the Amazon dataset, similar to the ROUGE-1 score. In contrast to ROUGE-1/2, the ROUGE-L score shows that backward search is always better.

D.3 Runtime Analysis

We report the inference runtime of BiMEANVAE and Optimus in Figure 10. For the Yelp/Amazon data, BiMEANVAE can generate 220.17/173.59 reviews/sec on average, while Optimus can generate only 0.68/1.16 reviews/sec on average. Optimus is a huge model that uses BERT as the encoder and GPT-2 as the decoder, which makes it more computationally expensive than BiMEANVAE. Nevertheless, the inference time is still acceptable for running summarizers in batch processing.

However, due to the GPU memory size limitation, it becomes infeasible for Optimus to take a batch size of more than 8, while BiMEANVAE can process much larger batches within a reasonable time.
Input Reviews:

I usually wear size 37, but found a 38 feels better in this sandal. I absolutely love this sandal. So supportive and comfortable, although at first I did get a blister on my big toe. Do not let this be the deciding factor. It stretched out and is now fabulous. I love it so much that I bought it in three colors. 

This is a really cute shoe that feels very comfortable on my high arches. The strap on the instep fits my feet very well, but I have very slim feet. I can see how it would be uncomfortably tight on anyone with more padding on their feet.

I love these sandals. The fit is perfect for my foot, with perfect arch support. I don’t think the leather is cheap, and the sandals are very comfortable to walk in. They are very pretty, and pair very well with pants and dresses.

My wife is a nurse and wears dansko shoes. We were excited to try the new crimson sandal and normally order 39 sandal and 40 closed toe. Some other reviews were right about a narrow width and tight toe box. We gave them a try and passed a great pair of shoes to our daughter with her long narrow feet, and she loves them...

Finally, a Dansko sandal that’s fashion forward! It was love at first sight! This is my 4th Dansko purchase. Their sizing, quality, and comfort is very consistent. I love the styling of this sandal and I’m pleased they are offering bolder colors.

I really love these sandals, my only issue is after wearing them for a while my feet started to swell as I have a high instep and they were a little tight across the top. I’m sure they will stretch a bit after a few wears.

I have several pairs of Dansko clogs that are all size 39 and fit perfectly. So I felt confident when I ordered the Tasha Sandal in size 39. I don’t know if a 40 would be too large but the 39 seems a little small. Otherwise, I love them. They are very cushiony and comfortable!

I own many Dansko shoes and these are among my favorites. They have ALL the support that Dansko offers in its shoes plus they are very attractive. I love the the heel height and instant comfort. They look great with slacks and dresses, dressed up or not...

SimpleAvg avg:

This is a great shoe. It is very comfortable, and the fit is perfect. The only issue is that it’s a little big on the toe area, but it’s not a problem. It is very comfortable to wear and it’s very comfortable.

COOP coop:

This is a very nice sandal that is comfortable and supportive. The only problem is that the straps are a little tight in the toe area, but it’s not a problem. They are very comfortable and look great with a pair of shoes and dress shoes. Love them!

Figure 11: Example of summaries generated by BrMeanVAE with SimpleAvg and COOP for reviews about a product on the Amazon dataset. The colors denote the corresponding opinions, and struck-through reviews in gray were not selected by COOP for summary generation (Note that SimpleAvg uses all the input reviews.) Terms that are more specific to the entity are underlined. Red and struck-through text denotes hallucinated content that has the opposite meaning compared to the input.
**Computing infrastructure**  
TITAN V

**Training duration**  
Yelp: 15 hours, Amazon: 12 hours

**Search strategy**  
Manual tuning

**Model implementation**  
https://github.com/megagonlabs/coop

| Hyperparameter                  | Search space | Best assignment |
|---------------------------------|--------------|-----------------|
| number of training steps        | 100,000      | 100,000         |
| batch size                      | 256          | 256             |
| tokenizer                       | sentencepiece| sentencepiece   |
| vocabulary size                 | 32000        | 32000           |
| embedding size                  | 512          | 512             |
| encoder                         | BiLSTM       | BiLSTM          |
| hidden size of encoder          | 256          | 256             |
| pooling                         | choice[last, max, mean, self-attn] | mean |
| number of layers                | choice[1, 2] | 1               |
| prior distribution              | $\mathcal{N}(0, I)$ | $\mathcal{N}(0, I)$ |
| size of latent code             | 512          | 512             |
| decoder                         | LSTM         | LSTM            |
| hidden size of decoder          | 512          | 512             |
| free bits                       | choice[0.0, 0.1, 0.25, 0.5, 1.0, 2.0] | 0.25 |
| KL annealing strategy           | choice[Cyclic, Pretrain+Anneal] | Pretrain+Anneal |
| learning rate scheduler         | linear schedule with warmup | linear schedule with warmup |
| learning rate optimizer         | Adam         | Adam            |
| Adam $\beta_1$                 | 0.5          | 0.5             |
| Adam $\beta_2$                 | 0.999        | 0.999           |
| learning rate                   | choice[1e-5, 1e-4, 1e-3] | 1e-3 |
| gradient clip                   | 5.0          | 5.0             |

Table 8: BiMEANVAE search space and the best assignments on Yelp and Amazon datasets.
| Hyperparameter          | Search space | Best assignment |
|-------------------------|--------------|-----------------|
| number of training steps| 500,000      | 500,000         |
| batch size              | 4            | 4               |
| encoder                 | bert-base-cased | bert-base-cased |
| decoder                 | gpt2         | gpt2            |
| prior distribution      | \( \mathcal{N}(0, I) \) | \( \mathcal{N}(0, I) \) |
| size of latent code     | 512          | 512             |
| free bits               | \( choice[0.0, 0.1, 0.25, 0.5, 1.0, 2.0] \) | 2.0             |
| KL annealing strategy   | \( choice[\text{Cyclic, Pretrain+Anneal}] \) | Cyclic          |
| learning rate scheduler | linear schedule with warmup | linear schedule with warmup |
| learning rate optimizer | Adam         | Adam            |
| Adam \( \beta_1 \)     | 0.5          | 0.5             |
| Adam \( \beta_2 \)     | 0.999        | 0.999           |
| learning rate           | \( choice[1e-5, 1e-4, 1e-3] \) | 1e-5            |
| gradient norm           | 1.0          | 1.0             |

Table 9: Optimus search space and the best assignments on Yelp and Amazon datasets.