Comparative Snippet Generation

Saurabh Jain, Yisong Miao, Min-Yen Kan
National University of Singapore

saurabhjain@u.nus.edu, miaoyisong@gmail.com,
kanmy@comp.nus.edu.sg

Abstract

We model product reviews to generate comparative responses consisting of positive and negative experiences regarding the product. Specifically, we generate a single-sentence, comparative response from a given positive and a negative opinion. We contribute the first dataset for this task of Comparative Snippet Generation from contrasting opinions regarding a product, and a performance analysis of a pre-trained BERT model to generate such snippets.

1 Introduction

The proliferation of opinions on the Web has transformed the way users express their opinions and experiences about aspects of products and services. *Online user reviews* contribute personal opinions, and when aggregated together, these reviews play a crucial role in purchasing decisions. However, due to the large volume of reviews, it can be infeasible for customers to skim all such sources. As users have to navigate through a large pool of opinions to make decisions, *opinion mining and summarization* grows in importance. As such, this area of work has received significant attention.

Many e-commerce platforms provide functionalities to compare products. These functionalities may be template-based and compare products on the basis of information provided by sellers. Comparative opinions from customers, who are experienced users of the product or service, are largely missing from such template-based comparison. On the other hand, question answering systems based on reviews, such as AmazonQA (Gupta et al., 2019), often only tell one side of the story in the response: either positive or negative. In our opinion, there is a demand for compact representations of both positive and negative opinions of products. Such compact textual representation could be enunciated by dialogue agents or shown as a succinct summary to drill-down on in a mobile interface. To the best of our knowledge, no such work has been done yet to provide *comparative responses* regarding a product to a user. We attempt this novel task. We take as input a positive and a negative opinion regarding a product and generate a comparative, single-sentenced fused response, which we call a comparative snippet (Fig. 1).

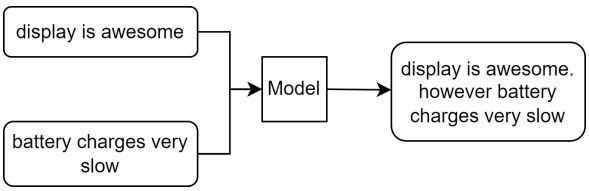

Figure 1: Comparative Snippet Generation: taking a positive and a negative opinion and generating a comparative response.

We extract single-sentence summaries of positive and negative opinions, separately, from reviews of 3,269 products mentioned by the Amazon Reviews Dataset (2018). We chose to base our corpus on this existing dataset to help spur future research on our task that can leverage existing work on the parent dataset. We then combine these positive and negative opinions to generate comparative responses. Our final dataset contains $174,394$ training instances, $19,725$ validation instances, and $21,397$ test instances[1]. We also successfully model positive and negative opinions to generate a comparative response expressing both positive and negative opinions about a target product.

2 Related Work

Sentence Sentiment Detection. Sentiment detection classifies the opinion of a sentence into two classes, *Positive* and *Negative*. Sometimes a third class, *Neutral*, is also included. Early works focus on unsupervised approaches and the use of sentiment lexicons to compute the overall sentiment of a text; e.g., (Turney, 2002). Subsequently, the convo-

[1] https://github.com/WING-NUS/comparative-snippet-generation-dataset

lutional neural network (CNN) architecture was introduced to classify the sentiment of sentences Kim (2014). Socher et al. (2011) use recursive neural networks to learn sentiment at varying granularities (i.e., words, phrases, and sentences). Many current well-performing neural models use the attention mechanism (Vaswani et al., 2017; Devlin et al., 2019) to encode a text into a vector representation.

Opinion Summarization. Opinion summarization differs from other summarization tasks in two aspects. First, it cannot rely on reference summaries for training, as it is infeasible to get such meta-reviews. To produce a reference summary for a single product, a reviewer may have to go through hundreds of reviews. Second, due to the subjectivity and conflicting nature of reviews, the notion of information importance applies differently. In this task, output summaries are based on the popularity of opinions. Moreover to be viable, approaches must be flexible with respect to input size as products can be reviewed frequently, resulting in increasing amounts of review content.

Opinion summarization can be either *abstractive* or *extractive*. In abstractive summarization, summaries are generated token-by-token to generate new sentences that articulate prevalent opinions from the inputs. These generated summaries offer a solution to the lack of reference summaries, and can be written in the style of the input reviews. However, prior work have used unrealistically small number of input reviews — 10 or fewer — to generate output summaries Suhara et al. (2020); Amplayo and Lapata (2021). Due to these shortcomings, we chose the alternative style of extractive summarization, which generates summaries by selecting phrases from the inputs. As a foundation, we base our method on Angelidis et al. (2021), who used the Vector-Quantized Variational Autoencoder (VQ–VAE) in their extractive opinion summarization. First introduced by van den Oord et al. (2017), VQ–VAE is used to learn discrete latent variables. It passes encoder output through a discretization bottleneck by lookup in the space of latent code embeddings. Specifically, we use the Quantized Transformer (*cf.* § 3.1), an unsupervised neural model inspired by VQ–VAE, to generate popularity-driven opinions. This method does not depend on vector averaging, nor does it suffer from information loss, which motivates us to use it as it easily accommodates large numbers of reviews.

Sentence Fusion. Sentence fusion combines multiple sentences, which may contain redundancies, into one coherent sentence. The output sentence not only should preserve input information but also any semantic relationships among sentences. Sentence fusion requires understanding the discourse semantics between the input sentences. Previously, feature-based approaches were used to combine sentences due to the lack of annotated data. Recently, a large-scale sentence fusion dataset, DiscoFuse Geva et al. (2019), was introduced, which has enabled the training of neural network-based models for the fusion task. The authors also train the sequence-to-sequence model to fuse the input sentences and find that the trained model succeeds in combining the sentences through structural constructions, but performs badly when fusion involves inserting discourse connectives. Recently, Rothe et al. (2020) uses a BERT-based encoder–decoder model. Although this work improves the accuracy, it struggles in detecting the semantic relationships correctly between the input sentences.

Predicting discourse markers or connecting strings is a sister task of sentence fusion. It is typically utilized as an intermediate step to improve downstream tasks. Ben-David et al. (2020) train a model to learn both the discourse relation and discourse connective together in a multi-task framework. In our work, similar to Rothe et al. (2020), we fuse two sentences together by training a model to learn the appropriate insertion of a discourse connective.

3 Dataset Generation

An instance of our dataset contains positive and negative opinions as an input and a comparative response as an output as shown in Table 2. Since no such dataset is available for reviews, we generate it from scratch. Here, dataset generation includes the tasks of opinion extraction and rule-based response generation sub-tasks. The task of opinion extraction itself includes subtasks of extraction, polarity classification, and summarization of segments.

3.1 Opinion Extraction

Segment Extraction. A sentence of a review may contain more than one opinion. For e.g., *"display was quite bland, didn't enjoy much, but speed was brilliant."* This sentence contains a positive opinion, *"but speed was brilliant"*, and a negative opinion, *"display was quite bland"*. Therefore, as suggested by Angelidis and Lapata (2018), it is

Review	In the end, take this tablet for what it is, a low end budget tablet that runs Lollipop smoothly but has a less than desirable screen resolution.
EDUs	In the end, take this tablet
	for what it is, a low end budget table
	that runs Lollipop smoothly
	but has a less than desirable screen resolution.

Table 1: A review and its extracted segments.

Input	the display is awesome. camera is not good.
Output	the display is awesome. however, camera is not good.

Table 2: An instance of our dataset.

beneficial to process phrases and discourse units extracted from review sentences compared to processing these sentences directly. Hereafter, we refer to these phrases and units as *segments*. We use work done by Feng and Hirst (2014) to extract segments from reviews (as shown in Table 1's example). After extracting segments, we perform the following five post-processing steps to improve overall quality:

1. We remove segments having less than three words, e.g. *"good product"*, *"best product"*, *"very sad"*, etc. Such short segments are not relevant to our work.

2. We remove leading and trailing punctuations; e.g., ".", "!". "," and "-".

3. We remove segments that do not contain at least one noun or pronoun and one main or auxiliary verb; e.g., *"the only problem"*, *"and was destroyed"* and *"which is annoying"*. We use Spacy[2] to extract a noun and a verb from a segment.

4. Since we focus on working with segments with third-person narrative, we discard segments containing first-person words: "i", "me", "my", "myself", "mine", "we", "us", "our", "ourselves". While our *extractive summarization* approach (§3.1) will eventually rank such segments low, we prefer to drop these here for efficiency.

5. If a segment starts with We delete any leading occurrences of "because", "and", "before", "but", "however", "now", "of", "then", "&", "or" from the segment. As an example, we edit *"but it is not that great"* into *"it is not that great"*, by omitting the leading *"but"*.

Segment Sentiment Classification. We next classify each segment into one of two categories, positive or negative. Reviews from different domains may differ in syntactic properties — e.g., length and vocabulary — however, the underlying semantics and discourse properties remain the same. To the best of our knowledge, there are no segment-level polarity-annotated datasets that build from the Amazon Reviews. As such, we use SPOT: Sentiment Polarity Annotation Dataset[3], which contains 197 reviews taken from the Yelp Tang et al. (2015) and IMDB Diao et al. (2014) datasets, annotated with segment-level polarities for positive, neutral, or negative sentiments. AS our work only utilizes positive and negative opinions to generate a comparative response, we discard the neutral segments. We fine-tune BERT Devlin et al. (2019) for polarity classification using the SPOT dataset. Then we classify extracted segments into positive or negative class using the fine-tuned model.

Segment Summarization. Products may have a large number of reviews. In our dataset, the single most-reviewed product has a massive $10,222$ reviews, generating $90,314$ segments – completely infeasible to manually process. Also, many reviews may express the same meaning. These two characteristics strongly motivate the need for a summarization algorithm to extract popular segments. Our summarization algorithm should satisfy the following requirements: 1) it must be unsupervised, since we do not have reference summaries; 2) it must be highly scalable since reviews per product regularly exceed $1,000$ inputs; and 3) it should extract frequently-occurring segments. In the case of reviews, we observe that the popularity of a segment is generally associated with their frequency of repetition. If several reviewers talk about a specific segment, e.g., *"display is very good"*, in their reviews for a product, it becomes a popular segment.

To satisfy these requirements, we employ the

[2] https://spacy.io/

[3] https://github.com/EdinburghNLP/spot-data

Positive	Negative
it is great	battery life is lackluster
the display is awesome	camera is not good
screen is great	does have some issues with clearing memory
meets all expectations	it 's just annoying
this tablet is fantastic	eventually it refuses to turn on at all

Table 3: Extracted Summaries. This table depicts five segments each of positive and negative opinions from a extracted summary of a product.

technique of Angelidis et al. (2021). They train an embedding space consisting of latent codes. Each latent code is a randomly-initialized vector that groups semantically similar segments. Then, a later part of the algorithm extracts top segments from each code which are considered popular segments.

Our work uses a slightly different approach to determine segments to extract. While Angelidis et al. (2021) use a threshold for the total number of words in the desired output summary, our method emphasizes popularity: we select segments for the output summary which are sampled greater than a tunable threshold t times. With an overly high t (e.g., $t = 50$), too few segments are selected; but if set too low (e.g., $t = 5$), the resultant segment quality is often poor and also often syntactically invalid, semantically incomplete or repetitive. We set the threshold to $t = 18$, based on appropriate empirical tuning on our validation set. For each product, we perform summarization on positive and negative opinions separately. Table 3 illustrates a few examples of extracted summaries.

3.2 Rule-based Response Generation

After extracting popular positive and negative segments separately, our final step is to generate the contrastive snippets, conforming to the format exemplified by Table 4.

We first analyzed sampled extracted reviews to understand how users actually combine two contrasting opinions when writing their own reviews. As a result, we inventoried seven common templates that users employ to combine both positive (POS) and negative (NEG) opinions:

1. {POS} . but , {NEG} .
2. {POS} . however {NEG} .
3. {POS} . on the other hand , {NEG} .
4. although {POS} , according to a few users {NEG} .
5. {POS} . yet , some users have also mentioned that {NEG} .
6. {POS} . however , there are people who have complained that {NEG} .
7. {POS} . on the other hand , a few users have complained that {NEG} .

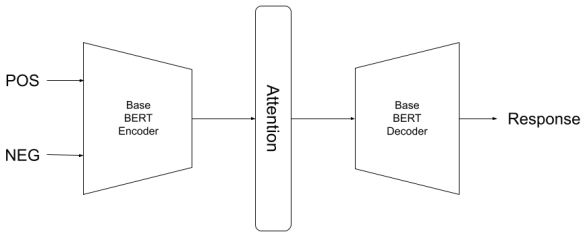

Figure 2: Model Architecture from Rothe et al. (2020).

For a given product e, let O_p and O_n represent a set of positive and negative opinions, respectively, extracted by our opinion extraction method. We combine $o_p \in O_p$, and $o_n \in O_n$ using the templates, as illustrated in Table 4, to generate an output response. Then, for each product, we combine each positive segment, o_p, with each negative segment, o_n, present in the respective extracted summaries.

4 Model Architecture

Given a positive opinion $o_p \in O_p$, and a negative opinion $o_n \in O_n$, our task is to generate a response R as shown in Table 4. We use an Encoder–Decoder based architecture similar to Rothe et al. (2020), as depicted in Fig. 2. For the encoder, we inherit BERT Transformer layer implementations which differs slightly than the canonical Transformer layer implementation Vaswani et al. (2017); as BERT replaces the standard RELU with GELU activation (Hendrycks and Gimpel, 2016). The implementation of our decoder is also similar to BERT with two modifications: First, the self-attention mechanism is modified to look only at the left context due to the unavailability of right context at the generation time. Second, an encoder–decoder attention mechanism is added as suggested by Rothe et al. (2020). We initialize both the encoder and decoder with publicly available pre-trained checkpoints from the uncased base model of BERT to learn and decode hidden representations. We join o_p and o_n, respectively, with a *full stop (.)* to make an input sequence. We use *mean cross entropy (MCE)* to compute loss. We fine-tune our model to

Input	**Positive Opinion**: it works great. **Negative Opinion**: camera is not good.
Response Format	{POS} . However , some users have also mentioned that {NEG} .
Generated Response	it works great . However , some users have also mentioned that camera is not good .

Table 4: Response generation example: a tuple of consisting of a template-based response, generated from an extracted positive and a negative opinion of a product.

generate a response fusing a positive and a negative opinion. In the next section, we describe our experimental settings and analyze results in detail.

5 Evaluation and Results

Dataset. We use reviews of products from the "Electronics" category of Amazon Reviews Dataset (2018) Ni et al. (2019). We generate our dataset in two phases, following the steps in § 3. In the first phase, we consider reviews of 74 products only and hand-curate segments in the generated summaries. We consider syntactically and semantically valid segments only. In the second phase, we scale the number of products and consider reviews of $3,269$ products. After extracting segments from these reviews, we consider only those segments for the summary generation which have part-of-speech patterns similar to hand-curated segments extracted in the first phase. Thus, we can ensure syntactic validity of the segments. Our final dataset contains $174,394$ training instances generated from reviews of $2,569$ products, $19,725$ validation instances generated from reviews of 321 products, and $21,397$ test instances generated from reviews of 379 products.

5.1 Implementation Details

We use the Transformer architecture in segment summarization model and uncased base BERT architecture for our response generation model.

Segment Summarization. As our summarization model is similar to Angelidis et al. (2021), we retain their settings in our experiments. We use a unigram LM SentencePiece vocabulary of size $32K$[4] to encode opinion segments. Our Transformer has a dimension size of 312, while its feed-forward layers are of size 512. It uses 3 layers and 4 internal heads. The input embedding layer is shared between the encoder and decoder, and $H = 8$ sentence heads are used to represent every sentence. For the quantizer, we set number of latent codes $k = 1024$ and sample $m = 30$ codes for each segment. We use the Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of 10^{-3} and a learning rate decay of 0.9. We disable segments assignments to latent codes for the first 4 epochs as warm–up steps for the Transformer. We train the model for a total of 20 epochs. At prediction time, in two-step sampling, we sample 300 latent codes, and for each code, we sample $n = 30$ segments.

Response Generation. Due to the effectiveness of BERT over Transformers in text generation tasks (Devlin et al., 2019), we use the base BERT model for our encoder and decoder. Since we initialize both the encoder and decoder with uncased base BERT pre-trained checkpoints, our experimental settings are similar to what were used while training the base BERT model. It has 12 layers, hidden size of 768, 12 attention heads, and vocabulary of $\sim 30K$ word pieces. We fine-tune this model for 5 epochs with a batch size of 32. Inputs and outputs are padded to a length of the largest available instance present in training, validation, and test sets.

5.2 Metrics

An output response should be evaluated on the basis of three aspects:

1. Preservation of input information. There should not be any change in positive and negative opinions. The semantic meaning and the syntactic structure of these opinions should be preserved. We use ROUGE-L to evaluate this aspect. It measures the longest common subsequence between an output sentence and a reference sentence. Since we do not modify the positive and negative opinions, the longest common subsequence is identical to one of the input opinions. For example, for an input sentence *"display is awesome. battery takes long time to charge."* and the corresponding output sentence *"display is awesome. however, battery takes long time to charge."*, the longest common

[4] https://github.com/google/sentencepiece

Recall	Model Based	Rule Based	Comparison Source
ROUGE-L	0.9876	1.0	Input and predicted output
ROUGE-3	0.8563	1.0	Prediction output and the most similar reference
ROUGE-4	0.7885	1.0	Prediction output and the most similar reference
ROUGE-2	0.8376	1.0	Connecting strings from the prediction output and the most similar reference
ROUGE-3	0.7884	1.0	Connecting strings from the prediction output and the most similar reference

Table 5: Comparative Snippet Generation Model Evaluation (Column 2; "Model Based"). The first row (ROUGE-L) measures input information preservation. The next two rows (ROUGE-3 and -4) measure the quality of predicted outputs, and the last two rows' entries measure the quality of the model-proposed connecting strings.

sub-sequence is *"display is awesome. battery takes long time to charge."*

2. Quality of output. In this aspect, we measure whether the order of words is correct, and connecting string is inserted at the right place. We use ROUGE-3 and ROUGE-4 to measure this aspect. These metrics measure the number of common trigrams and quadgrams between a generated output and a reference. We compare a generated output with each reference separately, and consider the score corresponding to the closest matching reference.

3. Quality of connecting string. This aspect measures whether words in connecting string are in the correct order and represent a valid sentence connector. For example, the connecting string *"on the other hand, some users have also mentioned that"* is of higher quality than the string *"on, some users have also mentioned that"*. We use ROUGE-2 to measure this aspect. Since a valid connecting string may comprise of part of two or more connecting strings, we do not use ROUGE-3 and ROUGE-4 to avoid heavy penalties. We remove those tokens from an output that are also present in the input sentence. We assume that thus remaining sub-string comprises tokens only from connecting strings. We repeat the same for all the references. Then we compute the ROUGE-2 metric between the processed output and references.

5.3 Results and Analysis

Overall Performance. We compute ROUGE-L-recall between input and a generated output to evaluate the model's performance in preserving input information. As shown in Table 5, recall of model-based generations is high: 0.9876; yet less than the perfect rule-based generation method that created the dataset. We consider recall values of ROUGE-3 and ROUGE-4 metrics to measure the quality of generated outputs with respect to the reference outputs. As shown in Table 5, recall values of ROUGE-3, and ROUGE-4 metrics for model-based generations are 0.8563, and 0.7885, respectively, which we believe is adequate. Finally, we compute the ROUGE-2 metric specifically confined specifically to the connecting string, in the manner described in (*cf.* § 5.2). As shown in Table 5, the recall of ROUGE-2 for our model-based generation is 0.8376. We believe that the connecting string quality is adequate but can definitely be improved with more careful modelling, and that the connecting string realization is a key component also contributing to overall quality in our second, overall output quality evaluation.

Study on Connective Prediction. We examine the connective prediction in more detail as this is the key aspect that is variable in the generation task. The model's prediction outputs exactly match with one of the references in 13.09% test cases. Table 6 shows distribution of connecting strings corresponding to these exact matches. Our model not only learns to generate comparative responses by fusing positive and negative opinions but also learns to generate new connecting strings by fusing words of two separate connecting strings or by appending a punctuation symbol with a connecting string. Our model fuses an additional 13.94% test cases with newly-generated connecting strings. Table 7 shows the distributional analysis of such newly-generated connecting strings. We can see that *"however, some users have also mentioned that"* occurs the maximum number of times in such test cases and has been generated by fusing *"however"*, and *"some users have also mentioned that"*. The second most-frequent generated string is *"however,"* in which a comma (*","*) has been appended to a connecting string.

Our model also generates incorrectly fused sentences. Table 8 shows main types of error. The

Connecting String	%age
but,	0.64%
however	44.68%
on the other hand,	33.65%
yet, some users have also mentioned that	0.86%
although [positive opinion], according to a few users	0.57%
on the other hand, a few users have complained that	10.79%
however, there are people who have complained that	8.82%

Table 6: Distribution of connecting strings for which a prediction output matches with one of the references. This distribution is with respect to the total number of exact matches.

New Connecting String	Percent.
yet, there are people who have complained that	0.07%
but, there are people who have complained that	0.07%
however, some users have also mentioned that	48.74%
but, some users have also mentioned that	0.23%
however,	48.34%
yet,	1.51%
but	0.10%
yet	0.94%

Table 7: Distribution of new connecting strings. This distribution is with respect to the total number of newly-generated connecting strings.

Error Type	Percent.
Incorrect mixing	48.22%
Missing although word	17.82%
Single word "on" insertion	14.65%
Input information modification	19.26%

Table 8: Common generation errors from our pre-trained BERT model's output.

maximum number of failure cases occur due to incorrect mixing of parts of different connecting strings. In such cases, either an extra word is inserted, or more words are missing from the connecting string. Table 9 depicts top incorrect mixing patterns. In the four patterns, we can see that the first word of negative opinion is inserted in between the connecting string. Our analysis shows that it happens due to the ambiguity present in our training dataset. For example, our training dataset contains connecting strings *"however"*, and *"however, there are people who have complained that"*. In the case of the former, just after the connecting string *"however"*, the model inserts words from the negative opinion. While in the latter case, words from a connecting string are inserted after "however". Therefore, we assume that at prediction time probability of inserting the first word of the negative opinion after *"however"* becomes highest, thus resulting in an incorrect mixing of connecting strings. A similar argument exists for the incorrect cases containing string *"on the other hand"*. Table 10 shows an example of incorrectly mixed connecting string.

The second most common failure case is associated with input information modification. Ideally, a comparative output sentence must contain positive and negative opinions without modification. But our model, sometimes, generates an output sentence that either deletes or repeats one or more words from the input or replaces a word with its synonym or base form (Table 10). As mentioned in the section (*cf.* § 3.2), except one, in all other templates connecting string is inserted between the positive and negative opinions. In case of *although POS, according to a few users NEG*, we also prepend a word *"although"* in the output sentence. Since most of the training instances insert a connecting string only in between, our model does not learn properly to prepend a word *"although"* and thus gives rise to the third most occurring failure cases in which the first word of the positive opinion is repeated instead of prepending a word *"although"*. An example of such a case has been shown in Table 10. The last most occurring errors are associated with single word insertion *"on"* between the positive and negative opinions, as shown in Table 10.

Why not use rules to generate responses if these give better performance? As shown in our results, rule-based generations outperform model-based generations. Therefore, an obvious question arises on the use of model-based generations. Templates used for generating rule-based generations have been manually selected from a random analysis of reviews. But, in the future, we want our model to automatically learn styles of comparative response generations from the given dataset and use these styles to fuse positive and negative opinions. Therefore, we prefer to use model-based generations and improve their accuracy.

6 Conclusion and Future Work

We introduced a novel task of generating a comparative response (or "snippet") regarding a product that combines positive and negative opinions together in a single sentence. As such comparative responses are not easily found in natural review environments, we generate such comparative re-

Incorrect Mixing Pattern	Distribution
however the other hand, [first word from the negative opinion] few users have complained that	9.85%
however [first word from the negative opinion] there are people who have complained that	7.83%
on the other hand, [first word from the negative opinion] few users have complained that	5.83%
on [first word from the negative opinion] there are people who have complained that	7.38%
on, some users have also mentioned that	8.15%
however the other hand,	6.40%

Table 9: Top incorrect mixing patterns. Here percentage is w.r.t. all the failure cases.

Incorrect mixing	Expected	the entire set is comfortable. on the other hand, a few users have complained that right side slides down.
	Predicted	the entire set is comfortable. on the other hand, right few users have complained that right side slides down.
Missing "although"	Expected	although the retractil system works fine, according to a few users the pads are sort of squarish.
	Predicted	the the retractil system works fine, according to a few users the pads are sort of squarish.
Insertion of "on"	Expected	the 415's are a great upgrade from the oem earbuds. but, it is super uncomfortable.
	Predicted	the 415's are a great upgrade from the oem earbuds. on, it is super uncomfortable.
Information modification	Expected	sound is pretty good. but, the movement is actually more like a saw.
	Predicted	sound is pretty good. however, the movement is actually more like a see.

Table 10: Examples of top errors.

sponses through extractive summaries of product reviews using an unsupervised approach. To spur future research in this area, we have also made our dataset public and leveraged the prior Amazon reviews corpus, popular with the research community. Throughout our work we assume that all reviews are genuine and have been written by buyers who have used the product. We investigate and benchmark a baseline model for this task that combines state-of-the-art text representation (BERT) in an encoder–decoder architecture to generate a comparative response. Our analysis of the output results shows that even such a state-of-the-art pre-trained model does not generate perfect responses.

There are limitations of our work that we hope to address in the future. Currently, positive and negative opinions in a generated response may or may not be related to the same aspect. As such, improvements to better generate more naturalistic responses may restrict generation to opinions where both the positive and negative discuss the same product aspect. In future we would also like to quantify the veracity of the opinions and weight them accordingly.

Acknowledgements

We acknowledge the support of NVIDIA Corporation for their donation of the Titan X GPU that facilitated this research.

References

Reinald Kim Amplayo and Mirella Lapata. 2021. Informative and controllable opinion summarization. In *EACL*.

Stefanos Angelidis, Reinald Kim Amplayo, Yoshihiko Suhara, Xiaolan Wang, and Mirella Lapata. 2021. Extractive opinion summarization in quantized transformer spaces. *Transactions of the Association for Computational Linguistics*, 9:277–293.

Stefanos Angelidis and Mirella Lapata. 2018. Multiple instance learning networks for fine-grained sentiment analysis. *Transactions of the Association for Computational Linguistics*, 6:17–31.

Eyal Ben-David, Orgad Keller, Eric Malmi, Idan Szpektor, and Roi Reichart. 2020. Semantically driven sentence fusion: Modeling and evaluation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1491–1505, Online. Association for Computational Linguistics.

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.

Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alex Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and

post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 511–521, Baltimore, Maryland. Association for Computational Linguistics.

Mor Geva, Eric Malmi, Idan Szpektor, and Jonathan Berant. 2019. DiscoFuse: A large-scale dataset for discourse-based sentence fusion. 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 3443–3455, Minneapolis, Minnesota. Association for Computational Linguistics.

Mansi Gupta, Nitish Kulkarni, Raghuveer Chanda, Anirudha Rayasam, and Zachary Chase Lipton. 2019. Amazonqa: A review-based question answering task. In *IJCAI*.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *ArXiv*, abs/1606.08415.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. 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 188–197, Hong Kong, China. Association for Computational Linguistics.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Yoshihiko Suhara, Xiaolan Wang, Stefanos Angelidis, and Wang-Chiew Tan. 2020. OpinionDigest: A simple framework for opinion summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5789–5798, Online. Association for Computational Linguistics.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432, Lisbon, Portugal. Association for Computational Linguistics.

Peter Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 417–424, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. In *NIPS*.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.