National University of Singapore

Master's Thesis

Comparative Response Generation

Author
Saurabh JAIN

Supervisor
Prof. Min-Yen KAN

Submitted in part fulfilment of the requirements
for the degree of Master of Computing at

Web Information Retrieval / Natural Language Processing Group
(WING)
School Of Computing

Declaration of Authorship

I, Saurabh Jain, declare that this thesis titled, "Comparative Response Generation" and the work presented in it are my own.

I confirm that:

• This work was done wholly or mainly while in candidature for a research degree at this University.

• Where I have consulted the published work of others, this is always clearly attributed.

• Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.

• I have acknowledged all main sources of help.

• Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Name: Saurabh Jain
Date: January 3, 2022

Abstract

Online reviews have become a prominent medium for users to express their opinions regarding products or services. Due to the rapid growth of the e-commerce industry and its approach even to the smallest section of the society, user-generated opinionated text is increasing at a burgeoning rate. Such vast numbers of reviews are equally important for buyers, sellers, and manufacturers. It fuels the requirement for systems to automatically analyse and distil information from them.

Since reviews are opinions of experienced customers, these play an important role in making a decision regarding a product or service. Due to the vast number of reviews, it is not feasible for a potential buyer to go through all of these. Although e-commerce platforms provide information regarding a product or service and template-based functionality to compare products, these comparisons are based on information provided by sellers. These sources do not include information present in the reviews. This thesis attempts to bridge the gap by modelling reviews to generate comparative responses consisting of positive and negative experience regarding the product.

Specifically, we generate a comparative response from a given positive and a negative opinion. The main contributions of our thesis are: 1) The creation of a dataset for Comparative Response Generation from contrasting opinions regarding a product, and 2) the introduction of a neural language model to generate comparative responses. To the best of our knowledge, no previous work has been done to generate comparative responses from review opinions. We achieve a score of 0.72 SARI, a metric to evaluate output sentence against input and references, on our test dataset.

Keywords Comparative Response, Amazon Reviews, Opinion Summarization, Opinion Fusion, Sentence Fusion.

Acknowledgements

I would like to pay my sincere gratitude to Prof. Min-Yen Kan for warmly welcoming me as a member of the Web, Information Retrieval / Natural Language Processing Group (WING) and his patience and guidance alongside my master study. He helped me develop preliminary research skills and motivated my interest in research.

I would like to express my sincere thanks to Yisong Miao, who is also a member of WING. I want to thank him for his guidance alongside this project, being very thoughtful to me and for teaching me many research skills in detail with great patience. He is a very motivated and hardworking person.

I would like to say thank to my friends, Anurag and Rui Liu, for helping me to whenever required. I would also like to thank my fellow WING members for their accompaniment and support. We also express our sincere thanks to NVIDIA Corporation for gifting GPUs to our group which we use for performing experiments in this thesis.

I will always be grateful and thankful towards my parents, my sister, and the God. Due to their motivation and constant support in all difficult times, I am able to reach here and it will always encourage me to take a step ahead.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The proliferation of opinions on the Web has transformed the way users express their opinions about aspects of everyday experiences. While social media platforms, like Twitter[1] and Facebook[2], are foremost channels to express and spread information across the world quickly, personal opinions are contributed and more densely concentrated in the medium of *online user reviews.*

Online e-commerce websites, like Amazon[3], Flipkart[4], etc., or purpose-built websites, like movie content review websites *Rotten Tomatoes*[5] and IMDB[6], allow customers to express their opinions about entities of interest. This is a significant shift from the time when reviews were tailored by a small number of professional critics, and it has a huge impact on industries like entertainment [26], hotel [102], and commerce (TurnTo Report, 2017). At the same time, the number of reviews written every day is growing very fast; more than 27 million reviews were written on Yelp[7] website in 2017, a figure which tripled in less than 5 years (Yelp Report, 2017). These reviews play a crucial role in making decision to buy a product or service. However, due to such a large and continuously increasing volume of reviews, it is infeasible for a customer to skim all these sources.

[1] https://twitter.com/
[2] https://www.facebook.com/
[3] https://www.amazon.com/
[4] https://www.flipkart.com/
[5] https://www.rottentomatoes.com/
[6] https://www.imdb.com/
[7] https://www.yelp.com/

Such a large source of opinions has sparked interest in Natural Language Processing (NLP) research. As users have to navigate through a large pool of opinions to make decisions, *opinion mining* [67] — i.e., automatic analysis and summarization of opinions in text — has received a significant attention. Two questions are mainly discussed while analyzing opinions: 1) What are users talking about? and 2) How are they talking about it?

The first question is associated with the Aspect Detection task [75, 92]. Aspects are attributes of a product or service which play a crucial role in user's opinion regarding the product or service. For example, image quality and sound are two attributes of a tablet. The second question is associated with the Sentiment Analysis task [94, 68, 89, 84]. Sentiment analysis deals with classifying the polarity of a given text at the phrase [84], sentence [51], or document level [54, 101]. Sentiment analysis measures polarity of a text and takes discrete or continuous values, ranging from very positive to very negative.

E-commerce websites do provide functionality to compare products. These functionalities are template-based and compare products or services on the basis of information provided by sellers or product selling platforms. Comparative opinions from customers, who are experienced users of the product or service, are often missing from such a template-based comparison. On the other hand, QA system on reviews, like AmazonQA [40], often only tell one side of the story in the response, either positive or negative. In our opinion, while querying regarding an aspect of a product or service, both positive and negative opinions should be provided to the user. To the best of our knowledge, no such work has been done yet to provide *Comparative Response* regarding a product or service to a user. Our work in this thesis attempts to solve this problem. It takes a positive and a negative opinion regarding a product and fused these to generate a comparative response (Fig. 1.1).

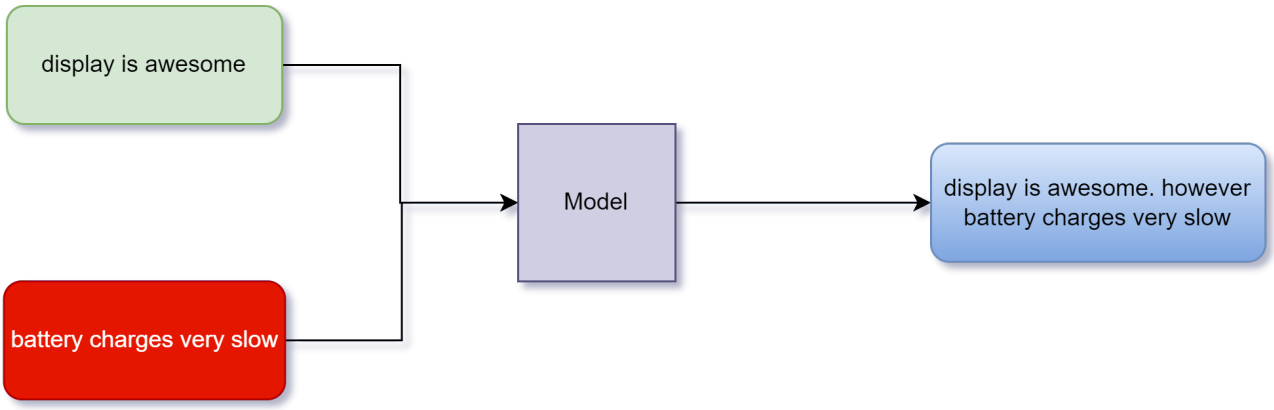

Figure 1.1: Comparative Response Generation. Our model takes a positive and a negative opinion and generate a comparative response.

1.1 Thesis Statement

In this thesis, we investigate a hypothesis related to high-quality comparative response generation, which we test through extensive evaluation and analysis of our method.

HYPOTHESIS: *A positive opinion and a negative opinion can be fused using a neural language model to generate a comparative response.*

1.2 Contributions

We extract single sentence summaries of positive and negative opinions, separately, from reviews of 74 products from the Amazon Reviews Dataset (2018). Then we combine these positive and negative opinions to generate comparative responses. Our final dataset contains 17,470 training instances, 971 each validation and test instances. We plan to open-source this dataset. We also successfully model positive and negative opinions to generate a comparative response expressing both positive and negative opinions about a target product.

Chapter 2

Background

The success of neural networks has helped the NLP community to shift from hand-crafted feature engineering and traditional pre-processing tools to the end-to-end training of models. These networks encode semantic [51, 101, 17], syntactic [16, 27], and discourse [50, 103] information into continuous vectors, which is passed to neural layers to learn task-specific parameters. Such models have set new performance standards in language understanding and generation.

To help the reader to form basic foundation to understand required NLP tasks and working of neural language models which help to understand later parts of our thesis, this chapter provides brief descriptions about key concepts and tasks in natural language processing that are relevant to our thesis, elementary discourse units, supervised sentiment analysis, attention-based models, transformer and BERT, extractive opinion summarization, and sentence fusion.

2.1 Elementary Discourse Units

Discourse parsing is the task of identifying the discourse units in a text and how these units relate to each other. Discourse segmentation determines the boundaries of discourse units, which are clause-like units that serve as building blocks of parse trees. These units are known as Elementary Discourse Units (EDUs). Here are examples of EDUs:

`[I bought this ipad for my daughter]`$_e$ `[and she is happy with it.]`$_e$

`[It has good backup,]`$_e$ `[we charge it once a day.]`$_e$

[*I bought this ipad for my daughter*] [*and she is happy with it .*] [*It has good backup ,*] [*we charge it once a day .*]

Label sequence: C C C C C C C **B** C C C C C C C **B** C C C C **B** C C C C C C

Figure 2.1: Sentences showing EDUs segmented with brackets and corresponding labels according to the sequence-based framework.

In the example, text phrases enclosed in the square brackets represent EDUs. Rhetorical Structure Theory (RST) [61] is the most widely accepted framework for discourse study. In RST, a text is represented by a hierarchical tree structure where EDUs are represented by leaves, and internal nodes represent a span of text containing multiple EDUs related by a discourse relation. EDUs identification problem is considered as boundary detection problem.

Two frameworks are popularly used for discourse segmentation. In the first, each token is processed sequentially and independently. A binary classifier is used to predict whether a boundary can be inserted before the token or not. [86, 87, 34, 49] use this framework.

The second approach frames task as a sequence labelling problem. A sequence is considered as a whole and a model predicts whether a token should be labeled as start of an EDU (Fig. 2.1) [42] uses this framework and implements sequence labelling using Conditional Random Fields (CRFs) [97]. Their model takes n-best outputs of a base segmenter and re-ranks n candidates.

Instead of considering a single token for feature extraction, [31] considers a pair of tokens adjacent to the current token to compute local features. The authors use a two-pass algorithm in which they use contextual, basic, and global features. Contextual features include feature vectors of the previous and the next tokens. Basic and global features have been shown in the Table 2.1 where T_i and T_{i+1} represent adjacent tokens. In the first pass, basic and contextual features are considered, and in the second pass, all three types of features are considered. We use this algorithm to extract EDUs from reviews.

Table 2.1: This table represents basic and global features used in [31]. Here T_i and T_{i+1} represent adjacent tokens.

Basic features	Global features
The part-of-speech tag and the lemma of T_i/T_{i+1}.	The part-of-speech tag and the lemma of the left/right neighboring EDU boundary.
Whether T_i/T_{i+1} is the beginning or the end of the sentence.	The distance to the left/ right neighboring EDU boundary.
The top syntactic tag of the largest syntactic constituent starting from or ending with T_i/T_{i+1}.	Number of syntactic constituent formed by the tokens between T_i/T_{i+1} and the left/ right neighboring EDU boundary.
The depth of the largest syntactic constituent starting from or ending with T_i/T_{i+1}.	The top syntactic tag of the lowest syntactic subtree that spans from T_i/T_{i+1} to the left-/right neighboring EDU boundary.
The top production rule of the largest syntactic constituent starting from or ending with T_i/T_{i+1}.	

2.2 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 [94, 46, 5, 99] to compute the overall sentiment of a text. [89] introduces SO-CAL, a then state-of-the-art method that combines sentiment lexicons with carefully-defined rules over syntax a tree to predict sentence sentiment.

Due to the availability of user-generated labels and large-scale crowdsourcing efforts, it became possible to train supervised learning models to predict the sentiment of a text. These approaches require a dataset D of text–label pairs $(t_i, y_i)_{i=1}^{|D|}$, where t_i is a text unit (sentence, phrase, or document) and y_i is its true sentiment label. The instance label y_i takes a value from the ordered label set, where lowest value represents a maximally negative sentiment and highest value represents maximally positive sentiment. t_i is a word sequence $w_1, w_2, ..., w_n$ of length n. Fig. 2.2 presents a toy example of training instances with positive and negative labels. These instances are used to train a supervised model, which is then used to predict an unseen text's sentiment.

A sentiment classifier, parameterized by θ, produces a probability distribution $\mathbf{p_i}$ over sentiment labels, and classifies s_i by selecting the most probable label:

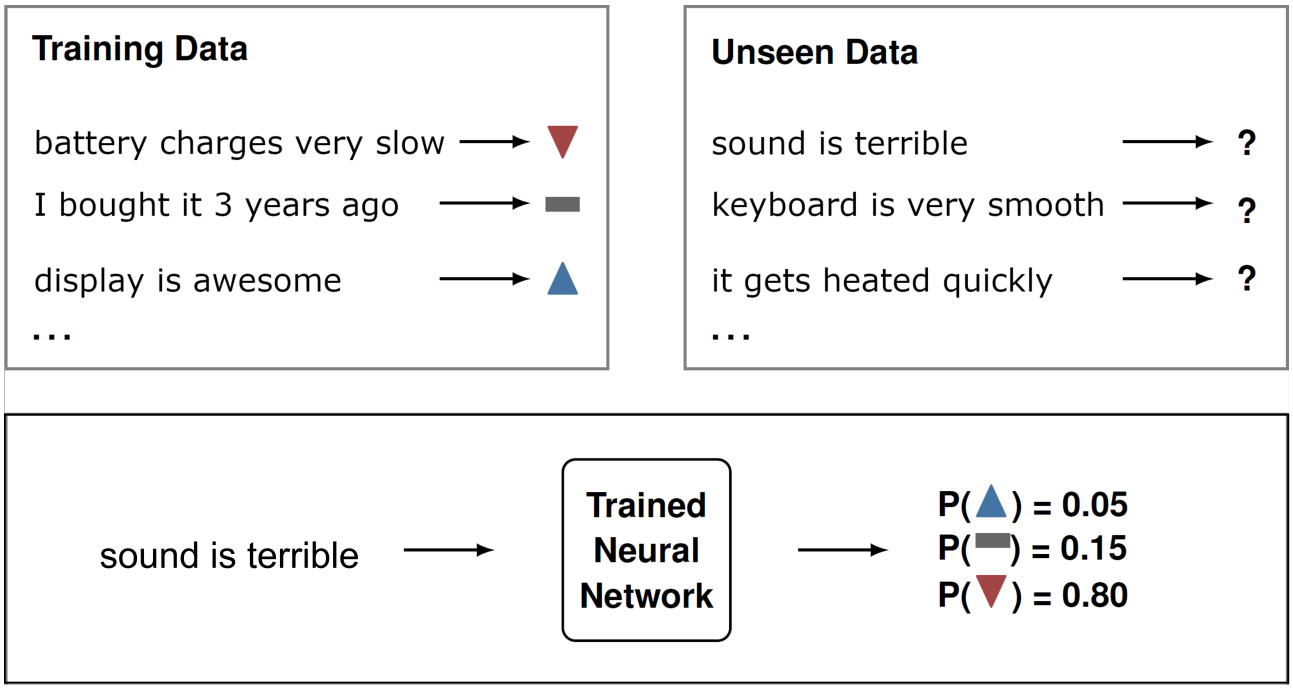

Figure 2.2: An example of supervised sentiment analysis as a supervised training task. Training instances with positive and negative labels are used to train neural model. The trained model is used to predict on unseen text.

$$\mathbf{p_i} =< p_i^{(1)}, ..., p_i^{(L)} >$$

$$p_i^c = P_\theta(y_i = c | w_1, ..., w_n)$$

In the case of a neural network, θ is defined by its deep learning architecture and learned via backpropagation. Neural network models have achieved state-of-the-art performance in the sentiment analysis. These models alleviate the need for feature engineering. [85, 84] use recursive neural networks to learn sentiment of varying granularity (i.e., words, phrases, and sentences). Later, [51] introduces the CNN architecture to classify the sentiment of sentences. With the availability of large-scale datasets [25, 90], researchers developed document-level sentiment-analysis models. Such models first build representations of sentences and then aggregate these into document feature vectors [90]. [101] presents a model which learns document representation by attending to individual parts of the text. Currently, most neural models use the attention mechanism to encode a text into a vector representation. We next describe two attention-based architectures.

2.3 Attention Based Models

With the success of neural networks, encoder–decoder architectures became popular for sequence transduction modeling. In such architectures, a neural network, named as an *encoder*, is used to encode an input text into a n-dimensional vector. This vector is fed into another neural network, a *decoder*, used to convert the vector into an output text. Recurrent neural networks, long short-term memory [43], and gated recurrent units [20] were used as basic units for such neural networks, producing state-of-the-art results for sequence modelling tasks [6, 18, 88]. These recurrent networks generate sequence of hidden states h_t as a function of previous hidden state h_{t-1} and input at position t. Their inherent sequential nature precludes parallelization. To address this shortcoming, the self-attention mechanism was used in variety of tasks including reading comprehension [17], abstractive summarization [71], textual entailment [70], and learning task-independent sentence representations [57]. [95] introduces transformer, a sequence transduction model, entirely based on self-attention to compute a representation of its input and output. We provide a brief description of it since we use this model in our work.

The transformer is an encoder–decoder model where the encoder maps an input sequence of symbol representations $(x_1, ..., x_n)$ into a sequence of continuous representation $z = (z_1, ..., z_n)$, following which the decoder generates an output sequence of symbols $y_1, ..., y_m$ using the continuous representation. To generate output at each step, the model considers the output of the previous step. The transformer model computes the importance of a symbol with respect to others in a sequence. Fig. 2.3 shows basic blocks of the encoder–decoder architecture of a transformer. These basic blocks are repeated N times and mainly consist of self-attention, feed-forward, and normalization layers. An input sequence is encoded into continuous vectors using pre-trained word embeddings and positional embeddings. It employs the *sinusoidal* function to process sequences with lengths more than those encountered during training. The authors use it to encode the position of a symbol in a sequence as:

$$PE_{(pos,2i)} = \sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{2.1}$$

where *pos* represents position of the symbol in the sequence, d_{model} represents output dimension

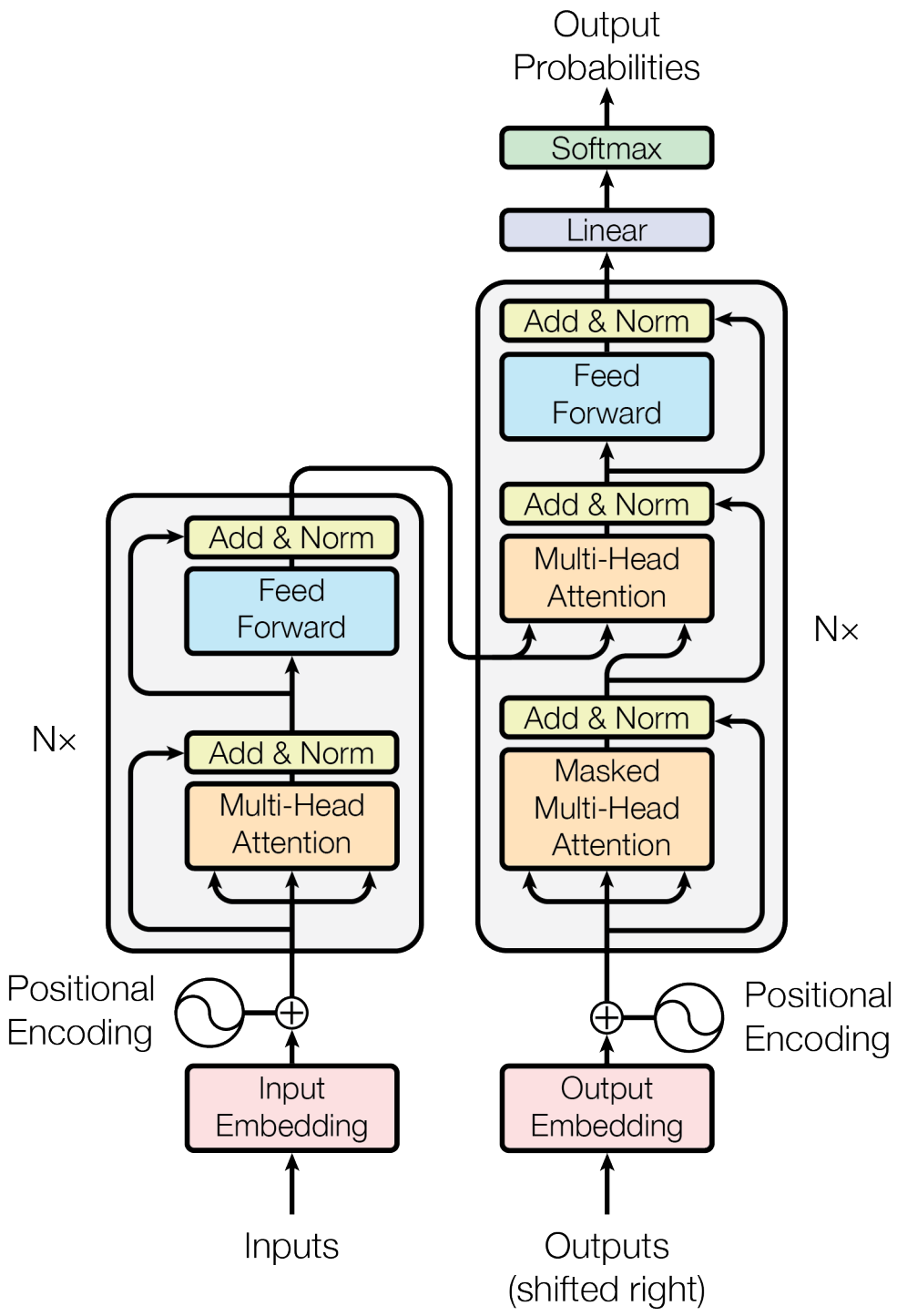

Figure 2.3: Encoder and decoder architecture of the transformer model. Figure reproduced verbatim from [95].

of embeddings, and i represents one of the dimension of d_{model}.

A self-attention layer maps a pair of query and key–value to an output where query, key, value, and output all are vectors. To compute attention, the scaled dot product (Fig. 2.4) is used:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.2}$$

where Q, K, and V are query, key, and value matrices, respectively, and d_k is the embedding dimension. This attention is performed h times so that model can attend to information from different representation subspaces at other positions. Afterward, the output of all attention computations are concatenated as follows:

$$\begin{aligned} MultiHead(Q, K, V) &= Concat(head_1, ..., head_h)W^o, \\ where\ head_i &= Attention(QW_i^Q,\ KW_i^K,\ VW_i^V) \end{aligned} \tag{2.3}$$

where projections are parameter matrices $W_i^Q \epsilon R^{d_{model} \times d_k}$, $W_i^k \epsilon R^{d_{model} \times d_k}$, $W_i^V \epsilon R^{d_{model} \times d_v}$, and $W^o \epsilon R^{hd_v \times d_{model}}$. Here, d_{model} represents output dimension of the embedding layer, d_k represents dimension of key matrix, and d_v represents dimension of value matrix. Fig. 2.5 depicts how multi head attention is computed using several attention layers. Then the multi-head output is fed to the feed-forward network followed by a softmax layer to generate output. The transformer architecture is an essential foundation for most state-of-the-art pre-trained language models. [21, 73, 77, 45] all show the effectiveness of pre-trained language models over a wide array of natural language processing tasks. Such models show improvement not only for sentence-level tasks such as natural language inference and paraphrasing but also for token-level tasks such as named entity recognition and question answering. There are two existing strategies for applying pre-trained representations to downstream tasks: feature-based and fine-tuning. In a feature-based approach, task-specific architectures use pre-trained representations as additional features. The fine-tuning approach uses minimal task-specific parameters and trains downstream tasks by fine-tuning all trainable parameters. [77] introduces the Generative Pre-trained Transformer. It is based on fine-tuning approach and uses a unidirectional language model to learn representations.

Later, [22] introduces BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers.

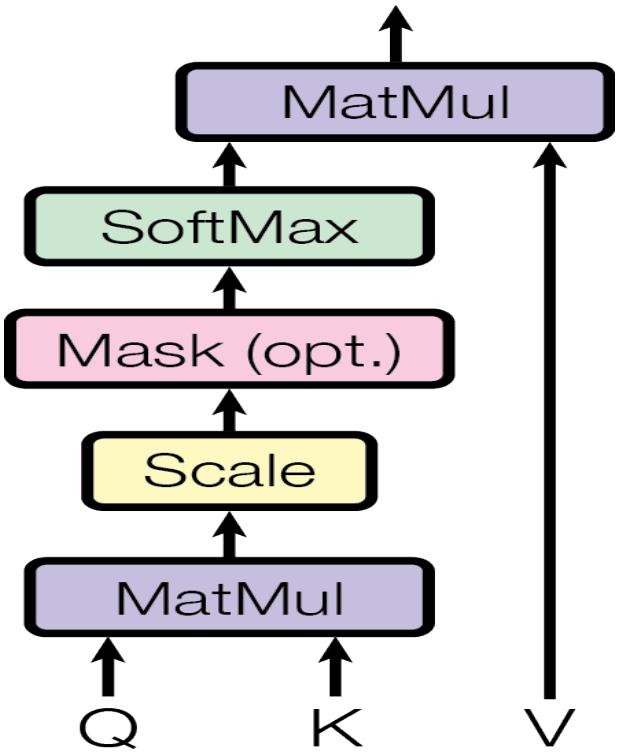

Figure 2.4: Scaled Dot Product Attention. Figure produced verbatim from [95]

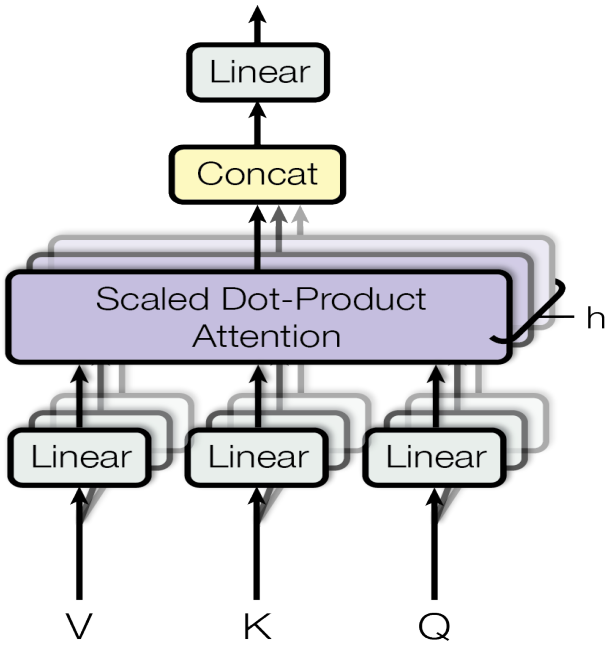

Figure 2.5: Multi-head Attention. Figure produced verbatim from [95]

BERT alleviates unidirectional constraint by introducing a masked language model (MLM) pre-training objective. The masked language model randomly masks some tokens in a sequence and learns to predict the original vocabulary of these tokens. MLM allows fusing left and right contexts to pre-train deep bidirectional transformer. The authors also pre-train a next sentence prediction task that jointly learns a representation of a pair of sequences. The pre-trained language model can be fine-tuned on a downstream task by learning from a task-specific dataset. Generally, the task-specific dataset is small compared to the dataset used to train the pre-trained model. Also, the fine-tuning process not only takes less time but also carries information learned by the pre-trained model. Our model architecture is comprised of BERT-based models. BERT takes a vector representation of input text. It uses three embeddings, namely word [38], segment, and token embeddings as shown in Fig. 2.6. The first token of every sequence is a special token $[CLS]$, The final hidden representation for this token is used as an aggregate representation of a given sequence. In the case of a pair of sequences, a special token $[SEP]$ is used to separate sequences. Fig. 2.7 (l) represents pre-training and (r) fine-tuning procedures in BERT. Except for the final layer in the fine-tuning, all other architectures are the same in both pre-training and

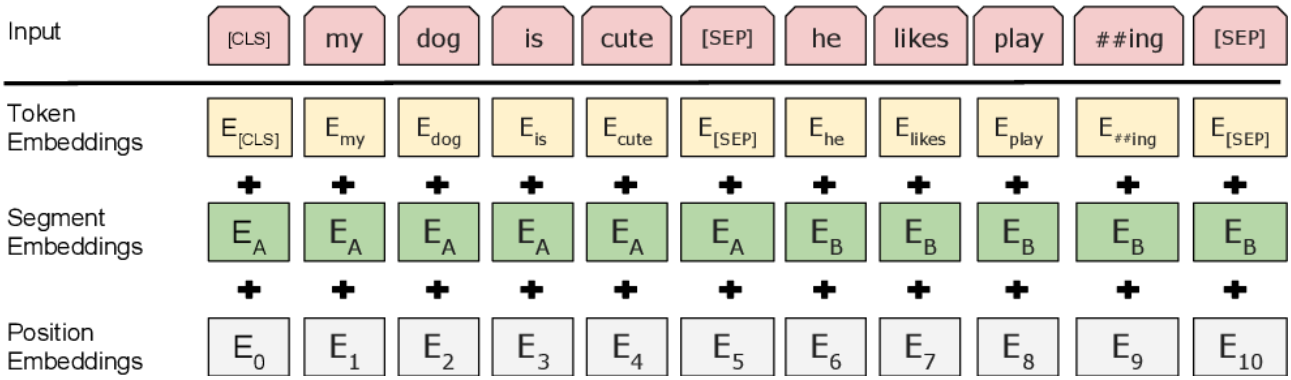

Figure 2.6: Representation of input in BERT, produced verbatim from [22]. Token, Segment, and Position embeddings are used to convert the input text into vector form.

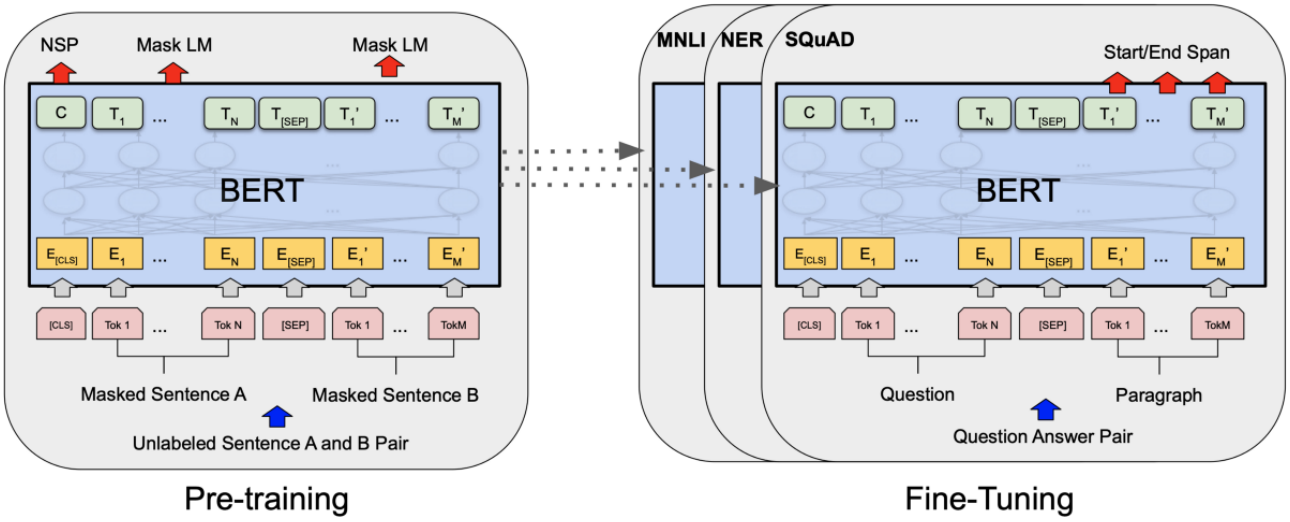

Figure 2.7: Pre-training and fine-tuning procedures in BERT, produced verbatim from [22]. The architecture of input and middle layers is uniform across both procedures. However, the output layer varies according to the downstream tasks at fine-tuning. In both procedures, two input sequences are combined using token [SEP]. The vector representation of the [CLS] token is used to represent the combined sequence.

Table 2.2: An example representing extractive and abstractive summarization. In extractive summarization, sentences are taken as it is from the review. While in abstractive summarization, model generates novel sentences.

Review
I had a very mixed experience at The Stand. The burger and fries were good. The chocolate shake was divine: rich and creamy. The drive-thru was horrible. It took us at leas 30 minutes to order when there were only four cars in front of us. We complained about the wait and got a half-hearted apology. I would go back because food is good, but my only hesitation is wait.
Extractive Summary
The burger and fries were good. The chocolate shake was divine. I would go back because the food is good. The drive-thru was horrible. It took us at least 30 minutes to order.
Abstractive Summary
The burger, fries, and chocolate were good. The shake was divine. Although I would like to go back, but the drive-through experience was not good.

fine-tuning. In our model, the base BERT model is used separately as an encoder and a decoder to generate comparative responses. Our model uses a positive and a negative opinion regarding a product to generate a comparative response. These opinions resemble popular views present in the product's reviews. In the next section, we briefly describe the opinion summarization task.

2.4 Extractive Opinion Summarization

The aforementioned related works all examine the general background for current methods in natural language tasks. We now review the two components specific to our thesis: opinion summarization and sentence fusion. We need these components to extract the opinions from the multiple input reviews in order to have text to generate such comparative sentences. For this reason, we also introduce opinion summarization techniques. Opinion summarization is different than 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 opinion summarization compared to other summarization tasks. In this task, output summaries are based on the popularity of opinions. Moreover, it should be flexible with respect to input size because products are reviewed frequently, resulting in more and more review content. Since a user may be interested in a general summary or focused on a particular aspect of a product (e.g., battery life of a laptop), our method should also generate controllable summaries. Similar to general summarization tasks, opinion summarization are also of two types, *abstractive* and *extractive*. Table 2.2 shows an example of both types of summarization. In abstractive summarization, summaries are generated token by token to generate novel sentences that articulate prevalent opinions from the inputs. Although thus generated summaries offer a solution to the lack of reference summaries, these summaries are written in the style of reviews. Such summaries can be generated from an aggregate representation of inputs and using sequence-to-sequence [83] neural models. [35] introduces Opinosis, a graph-based abstractive summarizer that explicitly models opinion popularity. They were the first to make a connection between opinion mining and text summarization. [23] adopts a hybrid approach in which important sentences are first extracted, and then templates are used to generate abstracts. [2] uses a weakly-supervised algorithm to extract important opinions according to their polarity intensity and aspect specificity. Vector averaging is one approach to model opinions. [19] introduces Meansum, an unsupervised method to generate abstractive summaries. The authors train the review decoder using a review-reconstruction task and generate summaries conditioned on averaged input vectors. [12] generates averaged vectors of reviews concerning the same entity. Then they use it to train a copy-enabled variational autoencoder trained for the review reconstruction task. However, abstractive summarization suffers from issues of text degeneration [44], hallucination [78], and first-person narratives. Apart from these issues, previous works use unrealistically small number of input reviews — 10 or less — to generate output summary. Due to these shortcomings, we use extractive summarization in our work, which aims to generate summaries by selecting few phrases from the inputs. [65] introduces VQ-VAE, Vector Quantized Variational Auto Encoder,

an enhanced training algorithm to learn discrete latent variables. This algorithm overcome problems of posterior collapse and large variance associated with Variational Autoencoders. It passes encoder output through a discretization bottleneck by lookup in the space of latent code embeddings. [4] uses VQ-VAE for extractive opinion summarization. We describe our approach in detail in § 3.1.3. Specifically, we use the Quantized transformer (*cf.* § 3.1.3), an unsupervised neural model inspired by Vector-Quantized Variational Autoencoder [65]. It is used to generate popularity-driven opinions. This method does not depend on vector averaging, nor does it suffer from information loss. It can easily accommodate a large number of reviews.

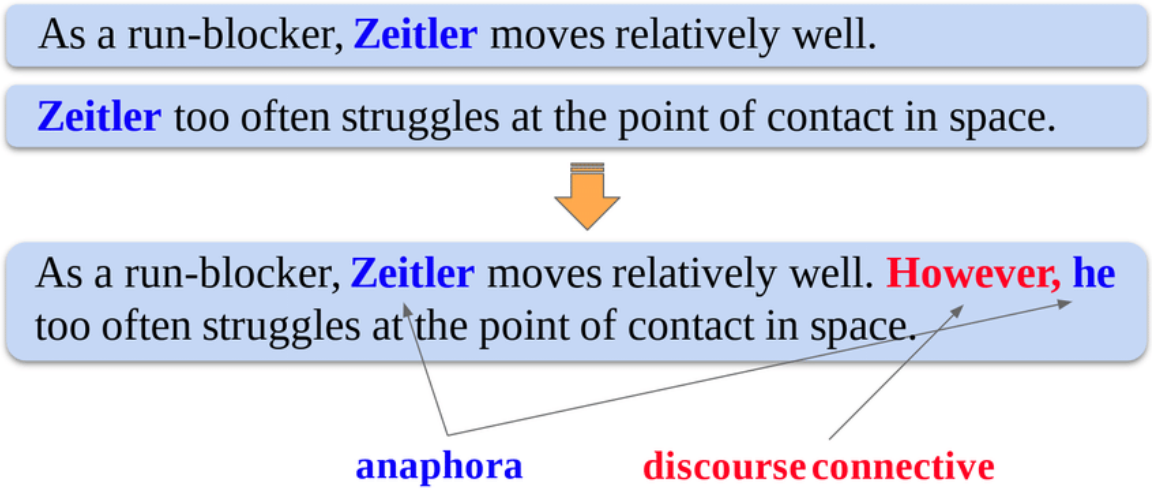

Figure 2.8: Example of Sentence Fusion, produced verbatim from [37]. We can see that two semantically contrastive senteces have been combined by discourse connector "However" and an entity "Zeitler" has been replaced by pronoun "he" in the second sentence in the output.

2.5 Sentence Fusion

The identified and extracted opinions need to be synthesized into a single sentence as a summary. Sentence fusion combines multiple sentences, which may contain overlapping information, 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. Fig. 2.8 depicts an example of sentence fusion. As shown in the figure, a fusion algorithm requires understanding that the second sentence contrasts the first one in order to insert correct discourse connective "However". In addition, the gender of entity "Zentler" needs to be inferred to insert the pronoun "he".

Previously, feature-based approaches were used to combine sentences due to the lack of datasets. [7, 93, 33, 29, 91, 10, 15] use small amounts of labelled data which is insufficient for training modern neural models. Recently, a large-scale sentence fusion dataset, DiscoFuse [37], has been introduced, which has made it possible to use data-hungry neural-network-based models to train 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. We use the balanced version of this dataset [60, 79] which contains $\sim$ 16.5 million examples. Later, [60] introduces LaserTagger and models sentence fusion as a sequence tagging problem. Recently, [79] 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 [28, 39, 59]. It is typically utilized as an intermediate step to improve downstream tasks [74, 105, 11, 76]. [58] uses connective prediction task in a multi-task learning framework to model discourse relation prediction and [47, 64] use it to compute unsupervised sentence embeddings. [8] trains a model to learn discourse relation and discourse connective together in a multi-task framework. Since, in our work, we fuse two sentences having opposite opinions, we train our model to learn to insert an appropriate discourse connective. Therefore our architecture is similar to [79].

Chapter 3

Problem Formulation and Model Architecture

Let C be a corpus of reviews on products $p_1, p_2, ...$ For every product p, we define its review set $R_p = r_1, r_2, ...$ Every review is a sequence of sentences $(x_1, x_2, ...)$ and a sentence, in turn, a sequence of words $(w_1, w_2, ...)$. For brevity, we use X_p to denote all review sentences for a product p. We formalize three sub-tasks: a) Separate review segments into two categories, positive and negative; b) Extract positive and negative opinion summaries, separately; and c) Generate a comparative response from a given positive and a negative opinion. We fine-tune BERT [22] to detect polarity of opinion sentences (*cf.* § 3.1.2). Then we fine-tune an encoder and a decoder using a segment reconstruction task to learn representation of segments and train a latent space to learn representation of latent codes (*cf.* § 3.1.3). We generate summaries for positive and negative segments, separately, by mapping input segments for a product to nearest latent codes and extracting segments representing popular codes (*cf.* § 3.1.3). Finally, we fine-tune an encoder-decoder model to generate a comparative response (*cf.* § 3.3). We make the following assumptions in our work:

1. We assume that all reviews are genuine and written by genuine buyers who have used the product. Therefore we do not handle spurious reviews separately.

2. Equal weight to both opposing opinions. We do not consider the frequency of occurrence of opinions while generating a comparative response.

3.1 Opinion Extraction

We consider products having at least 1000 reviews and belong to *Computers & Tablets* sub-category. There are a total of 74 such products. Amazon reviews are based on a five-star scale, and each one is defined by 11 attributes, of which we use only *reviewText* in our work. We also do not consider reviews containing HTML tags in our research.

3.1.1 EDUs Extraction

Inspecting positive and negative opinions from reviews is essential for product comparison. Contrasting opinions may be present in a review sentence, e.g., "display was quite bland, didn't enjoy much, but speed was brilliant." Therefore, as suggested by [1], it is beneficial to process phrases and discourse units extracted from review sentences compared to processing review sentences directly. So we transform each review into Elementary Discourse Units (Table 3.1). Hereafter, these units are referred to as *segments*. We use work done by [30] to extract segments from reviews.

After extracting segments, we manually inspect randomly-selected segments and perform the following five post-processing steps:

1. We remove segments having less than three words, e.g. "good product", "best product", "very sad", etc. Such short segments are unlikely to carry relevant information to our work.

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

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", "which is annoying", etc. We use Spacy[1] syntax tree to extract a noun and a verb from a segment.

4. Since we are focused to work with segments with third-person narrative, we filter out segments containing the following first person words: "i", "me", "my", "myself", "mine", "we", "us", "our", "ourselves" (although our *Extractive Summarization* approach, defined in the next section 3.1.3, gives lower rank to such segments, we prefer to drop these here.).

[1] https://spacy.io/

Table 3.1: A review and its EDUs. This table shows a review and its corresponding EDUs.

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.

5. If a segment starts with "because", "and", "before", "but", "however", "now", "of", "then", "&", "or", we delete very first occurrence of these words from the segment. As an example, we convert "but it is not that great" into "it is not that great", "and it is defective" into "it is defective", and "because the sound becomes distorted" into "the sound becomes distorted".

3.1.2 Segment Polarity Detection

After extracting EDUs, we classify each segment into one of two categories, positive and 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. Therefore, due to the unavailability of a segment-level polarity-annotated dataset for the Amazon Reviews, we use SPOT: Sentiment Polarity Annotation Dataset[2]. It contains 197 reviews taken from Yelp [90] and IMDB [25] datasets, annotated with segment-level polarities positive, neutral, or negative, represented by $+$, 0, and $-$, respectively. These annotations are gathered at two levels of granularity: 1) sentence, and 2) Elementary Discourse Units (segments). Table 3.2 represents format of segments of a review in the SPOT dataset.

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

Table 3.2: SPOT dataset instance. This table represents an example from SPOT dataset.

4 0316705
– Jerry and the lion is not one of the better tom and jerry cartoons ,
+ but it is light years away from being the worst .
0 If i were to categorise the cartoon
+ it would be somewhat strange yet very enjoyable .
– The story is n't great to be honest ,
+ but the art work is gorgeous
+ and the music is a lot of fun .
⟨*empty line*⟩

The first line contains an overall review rating followed by a single character space and its review ID. Subsequent lines represent a segment and its polarity, where the first character of each line represents the polarity followed by a ⟨*tab*⟩, and the segment itself. Since our work combines only positive and negative opinions to generate a comparative response, we only use positive and negative segments in our research. We fine-tune BERT [22] for polarity classification using the above SPOT dataset. Then we pass the preprocessed product review EDU segments to the classification model, separating these into negative and positive ones.

3.1.3 Segment Summarization

A single product can have a large number of reviews, and many reviews may express the same semantic meaning. For example, the product having the most number of reviews — $10,222$ — alone produces $90,314$ segments. Most of these segments are redundant since they express the same meaning. Therefore, we need 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 scalable since per product reviews vary from $1,000$ to $10,222$; and 3) it should extract frequently-occurring segments. In the case of reviews, popular segments are generally associated with their frequency of occurrence. 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 use work done by [3]. 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.

Let C be a corpus of segments from reviews on products $p_1, p_2, ...$ These segments may present any number of relevant aspects $a_1, a_2, ...$, like "battery", "display", "processor", etc. For every product, we define its segment set $S_p = b_1, b_2, ...$ and a segment s is a sequence of words $(w1, w2, ...)$. Our task is to generate general summaries from S_p for each product across all aspects. We train latent codes using a segment reconstruction task. A segment is passed to an encoder that produces a multi-head representation. A head is a vector which is supposed to carry information related to a segment, either syntactic or semantic. Then we map the segments onto their nearest latent codes and extract the top segments for every popular latent code.

[3] uses a variant of VQ-VAEs ([65, 80]) which consists of: (a) a transformer encoder that encodes a segment s into a multi-head representation $s_1, s_2, ..., s_H$, where $s_i \in R^D$ and H is total number of heads; (b) a latent space that maps each head of a segment to a discrete set of latent codes, and uses an average of representation of these codes to reconstruct head vectors as $q_1, q_2, ..., q_H$, where $q_i \in R^D$; (C) a transformer decoder that uses quantized vectors to generate reconstructed segment $\hat{s}$.

Latent Space Training

To receive segment-level vector representation, we prepend a special token $[SNT]$ to each segment. Each segment is passed to the transformer encoder [95] to generate token-level representations. Then, the vector representation of token $[SNT]$, $s_{[SNT]} \in R^D$, is used to generate the multi-head representation of a segment s by splitting $s_{[SNT]}$ into H sub-vectors, $\overline{s_1}, \overline{s_2}, ..., \overline{s_H}$, where $\overline{s_i} \in R^{D/H}$, followed by layer-normalization transformation:

$$s_i = LayerNorm(W\overline{s_i} + b) \tag{3.1}$$

where s_i is the $i-th$ head, and $W \in R^{D*D/H}$, $b \in R^D$ are shared across heads. Fig. 3.1 illustrates

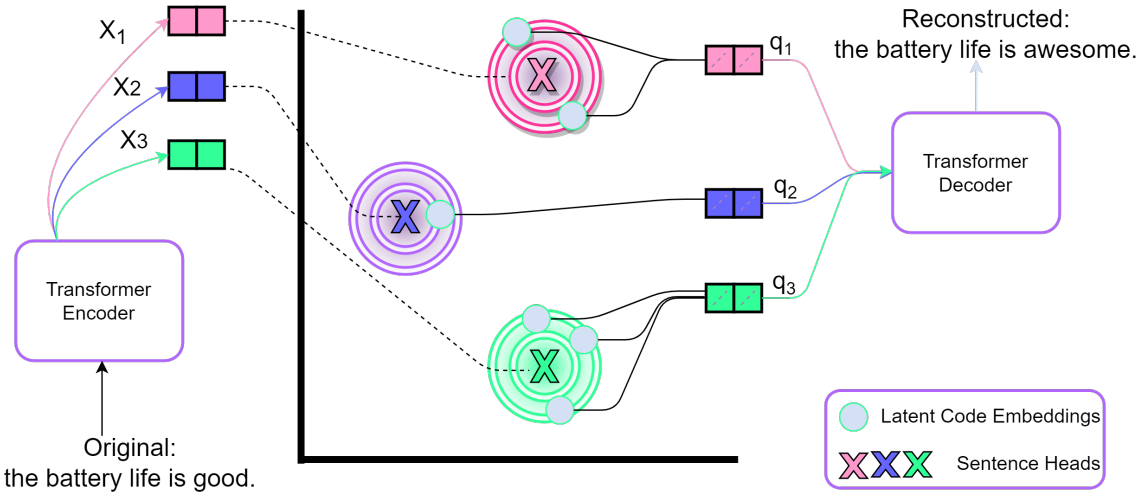

Figure 3.1: Segment reconstruction in latent space. A segment is encoded into 3-head representation and head vectors are used to compute quantized vectors. In turn, these quantized vectors are used to reconstruct the segment [3].

the work of an encoder where a segment is encoded into a 3-head representation. Let K be the total number of latent codes. Each code $e_k \in R^D$ is randomly initialized with a D dimensional vector. Let $z_1, z_2, ..., z_H$ be discrete variables corresponding to H heads. Each discrete variable is assigned one of K possible values, $z_h \in [K]$. Next, every head of each segment is assigned to latent codes. There are multiple ways to perform this assignment. We can assign a head to exactly one latent code. There will be a problem while assigning a head which is equidistant from more than one latent code. To overcome this issue, should we assign a head to m latent codes strictly based on its distance from codes? There are two problems associated with this assignment. First, we will need to find m unique latent codes every time. Second, there may be a possibility that a head is assigned to a code whose other assignments do not share similar semantic information with the code. To overcome these issues, we need a sampling method which samples latent codes on the basis of inverse or negative of the distance between a head and a latent code and can sample the same code more than one time, if required. *Multinomial* sampling method satisfies both of these requirements. Therefore, we use this method to sample, with replacement, m latent codes for each h^{th} head:

$$z_h^1, ..., z_h^m \sim Multinomial(l_1, ..., l_k), \; where \; l_i = -\|x_h - e_i\|_2^2 \tag{3.2}$$

where $Multinomial(l_1, ..., l_K)$ is a K-way multinomial distribution with logits $(l_1, ..., l_K)$. Af-

terward, head representations of each segment is reconstructed. Instead of using the original head vectors, an average of representations of latent codes, to which the h^{th} head is assigned, is considered as follows:

$$q_h = \frac{\sum_{j=1}^{m} e_{z_h^j}}{m} \tag{3.3}$$

As shown in Fig. 3.1, reconstructed head vectors q_1, q_2, and q_3 are computed using the average of embeddings of assigned latent codes. These reconstructed heads are combined and passed to the decoder, which in turn, generates a reconstructed segment. The model is trained to minimize the following reconstruction loss:

$$L = L_r + \sum_{h} \|x_h - sg(q_h)\|_2 \tag{3.4}$$

where L_r is the reconstruction cross-entropy loss, and $sg(\cdot)$ is a stop-gradient operator, which works as an identity function in forward propagation and but emits a zero during backward propagation. Since we use a sampling method to assign latent codes to a head and to extract top segments from each code, it creates a problem in computing gradients of loss function with respect to the inputs using backpropagation. To handle such a stochastic condition, we use the straight-through estimator, proposed by [9], to bypass a sampling equation. [81] shows that an exponentially moving average (EMA) update of the latent space results in more stable training than using the gradient-based methods. Therefore, we use EMA to train latent codes in our work.

Summary Generation

After training latent space, the next task is to extract popular segments from all the segments of a product. To perform this task, we adopt a two-step sampling process, as shown in Fig. 3.3, which simultaneously finds the popular latent codes and promotes commonly-occurring segments closer to popular codes. Specifically, a set of segments, $S_p = s_1, s_2, ..., s_N$, of a product p is input to a trained encoder. The encoder produces $N * H$ head vectors, $x_{11}, ..., x_{ih}, ..., x_{NH}$, where x_{ih} is the h^{th} head of the i^{th} segment. Each head is assigned to m randomly sampled latent codes

as defined in the previous section. Hence, the total number of assignments to a latent code counts as its popularity.

$$z_{ih} = arg \min_{k \in [K]} \|x_{ih} - e_k\|_2 \tag{3.5}$$

$$n_k = \sum_{i,h} 1[z_{i,h} == k] \tag{3.6}$$

Fig. 3.2 shows how segments are encoded, and their heads are assigned to latent codes. Thus, the codes which have been assigned several heads are considered as popular codes, and corresponding heads are considered as popular segments. A good summary should consist of segments from such popular codes.

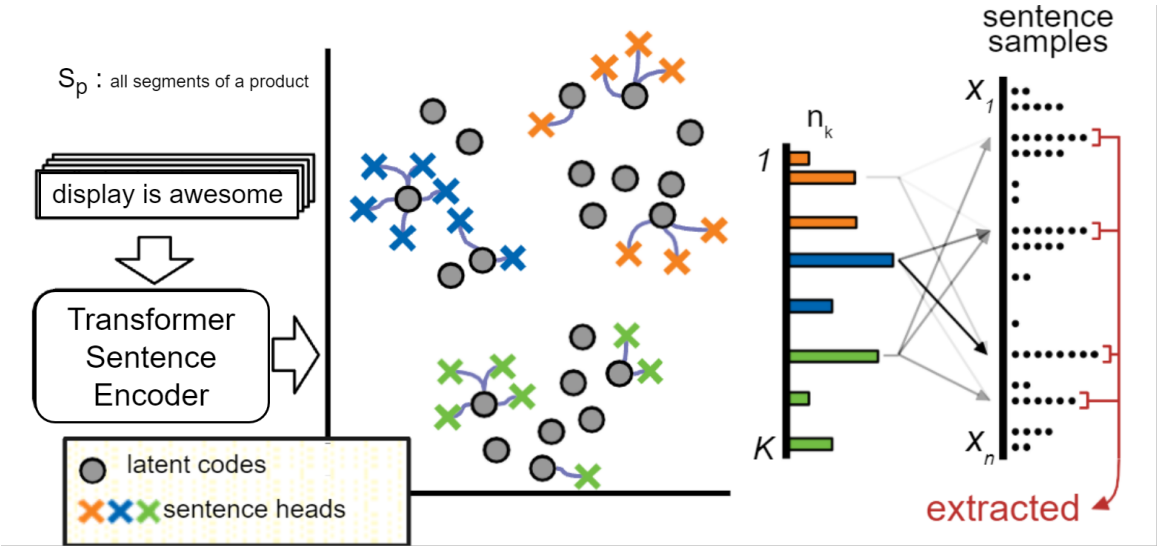

Figure 3.2: Segment heads and their mapping to codes. Figure shows segments encoding and assignment of their heads to latent codes.

Therefore, to generate a summary, we first sample popular codes and then sample top segments from each code. A latent code is sampled with probability proportional to its total number of assignments:

$$P(z = k) = \frac{n_k}{N * H} \tag{3.7}$$

where n_k is the total number of heads assigned to code k according to equation 3.6. For example, if the input contains many segments similar to "display is awesome", these segments are likely

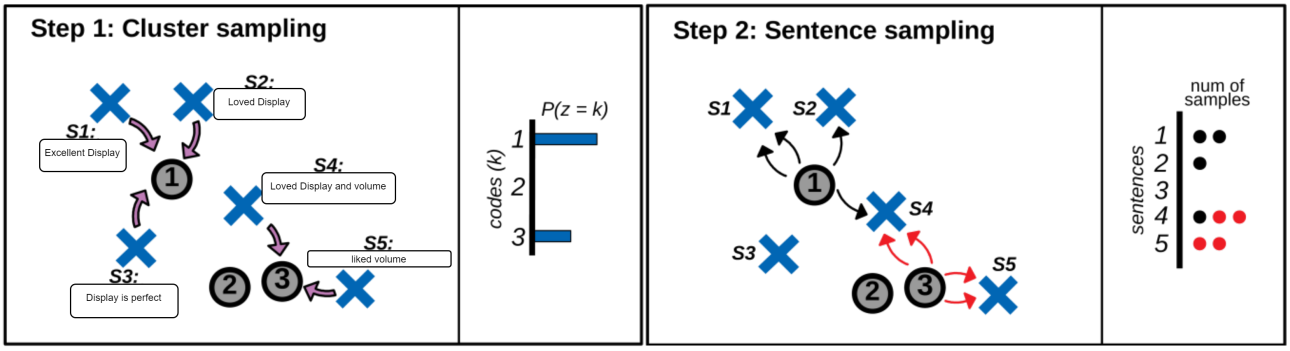

Figure 3.3: Two-step Sampling. In the first step segments are assigned to the clusters and popular clusters are extracted. In the second step, segments are sampled from popular clusters. Segments from Code 1 are shown in Black and segments from Code 3 are shown in Red.

assigned to the same code, which in turn increases the probability of getting that code sampled. The left diagram in Fig. 3.3 represents code sampling and resulting probabilities.

Next, we sample segments from each popular code. The sampled code z has multiple segments in its neighborhood. As mentioned previously, picking only one segment is not right. Therefore, we sample n segments, with replacement, as follows:

$$x_1, ..., x_n \sim Multinomial(\overline{l_i}, ..., \overline{l_N}),\ where\ \overline{l_i} = -\min_h \|x_{ih} - e_z\|_2^2) \tag{3.8}$$

where the multinomial's logits $\overline{l_i}$ represents the distance of the i^{th} segment from the closest latent code. Fig. 3.3 (right) represents an example of sentence sampling. After selecting Cluster 1 during cluster sampling, four segment samples are drawn. Later, Cluster 3 results in four more segment samples. Segment S4 gets the most votes, as it is sampled the most number of times. Two-step sampling is repeated multiple times, and all segments in S_p are ranked according to the number of times these are sampled. A summary is constructed by concatenating the top-ranked segments. Only segments that are sampled more than a tunable threshold t times are used in the output summary. With an overly high t (e.g., $t = 50$), too few segments are selected; but if set too low (e.g., $t = 5$), segment quality is too low and syntactically invalid, semantically incomplete or repetitive segments often get selected. We set the threshold to $t = 18$ based on appropriate empirical tuning using a validation set. For each product, we perform summarization on positive and negative opinions separately. Table 3.3 represents a few examples of extracted summary.

Table 3.3: Extracted Summaries. This table depicts five segments each of positive and negative opinions from a extracted summary of a product.

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

3.2 Training Dataset Generation

After extracting popular positive and negative segments separately, our next task is to generate a dataset into the required format as shown in Table 3.4 under "Response Generation Example". We perform a random analysis of reviews to know how reviewers combine two contrasting opinions while writing a review. After analysis, we finalize the following seven templates to combine a positive and a negative opinion:

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} .

where *POS* means a positive and *NEG* means a negative opinion. For a given product e, let O_p and O_n represent a set of positive and negative opinions, respectively, extracted by summarization as previously mentioned. We combine $o_p \in O_p$, and $o_n \in O_n$ using the templates, as illustrated in Table 3.4, to generate an output response. For a product, we combine each positive segment, o_p, with every negative segment, o_n, present in the respective extracted summaries. We use a phrase *"However , some users have also mentioned that"* to combine contradictory opinions.

Table 3.4: Response Generation Example. This table represents how output response is generated using a positive opinion, and a negative opinion of a product.

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

3.3 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 3.4. We use an Encoder–Decoder based architecture similar to [79], as depicted in Fig. 3.4. For the encoder, we inherit BERT Transformer layer [22] implementations which differs slightly than canonical implementation of a Transformer layer [95]; as BERT uses *GELU* activation [41] instead of standard RELU.

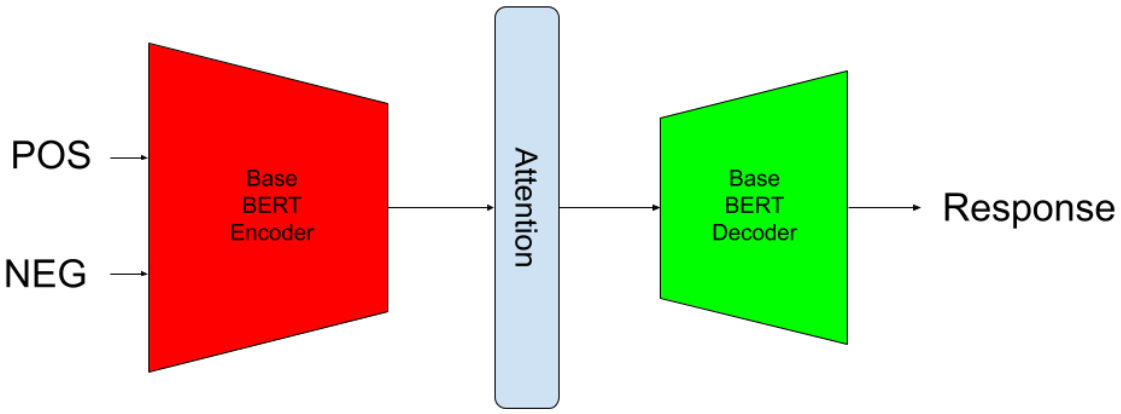

Figure 3.4: Proposed Model Architecture.

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. Second, an encoder-decoder attention mechanism is added. 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:

$$L_{gen} = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{L_{R_i}}\sum_{j=1}^{L_{R_i}}\log p(R_{i,j}|s_i, R_{i,1\ldots j-1}) \tag{3.9}$$

where N is the total number of instances in the training dataset, L_{R_i} is the length of i^{th} response R_i, $R_{i,j}$ is the j^{th} token of R_i, and $p(w|s_i, pre)$ is the model's probability for the next token to be w given the input sequence s_i and previously generated tokens pre. We fine-tune our model to generate a response fusing a positive and a negative opinion. In the next chapter, we describe our experimental settings and analyze results in detail.

Chapter 4

Evaluation

We now mention implementation details of our datasets and models, then provide a short description of our evaluation metrics. Later we discuss and analyze results from models trained on different datasets.

Dataset

Generating comparative responses from product reviews is a novel task. To the best of our knowledge, there is no existing dataset or benchmark for it. Therefore we curate our own dataset from Amazon Reviews Dataset (2018) (*cf.* § 3.1) [63] — hereafter known as the small dataset. This dataset contains 17470 training, 971 each test and validation instances.

This dataset is rather limited in size. To increase the size of the dataset, and to include a larger variety of sentence structures, we extract training instances from the balanced data from the Wikipedia category of the DiscoFuse dataset [37]. DiscoFuse is a large-scale dataset for discourse-based sentence fusion. In each data instance, two incoherent sentences are fused into a single coherent sentence. It contains total 60 million instances. The following is an instance taken from the DiscoFuse dataset:

First Sentence:	It may have been used as a temporary camp by the Romans or even by English Civil War forces , as for example , Castle Dore .
Second Sentence:	Evidence from any later period is also absent .
Fused Sentence:	It may have been used as a temporary camp by the Romans or even by English Civil War forces , as for example , Castle Dore . However , evidence from any later period is also absent.

Since our problem statement aims to fuse two contrasting opinions, we include only those instances from DiscoFuse dataset in which two contrasting sentences are connected by contrasting connecting strings; e.g., "however ,", "however", "yet ,", "on the other hand,", "on the other hand", "but", "but,", and "although". However, these instances are not necessarily be review instances. We mix these instances with our curated dataset from Amazon to obtain $432,730$ training instances and keep the same test and validation dataset which we use in our small dataset.

We fine-tune two models on these two datasets of different scales and report results on the same test data.

4.1 Implementation Details

We use the Transformer architecture [95] in segment summarization model and uncased base BERT architecture [22] for our response generation model. We run all experiments on an NVidia Titan RTX GPU and present other details in the following.

Segment Summarization

Our summarization model is similar to [3], therefore we keep the same settings in our experiments similar to theirs. We use a unigram LM SentencePiece vocabulary of size $32K$[1] to encode

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

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 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 [52] 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 the BERT model over Transformers for the text generation tasks [22], 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.

4.2 Metrics

If there are multiple valid reference outputs, then comparing prediction output with only a single ground truth is incorrect. Table 4.1 shows an example of comparative response, containing one ground truth and three references. It can be observed that all of references convey similar and valid meanings. Therefore we should consider all references to evaluate performance of our model. [100] mentions that for sentence fusion tasks, it is important to include input along with references to evaluate performance. Importantly, they introduce a new evaluation metric, **S**ystem output **A**gainst **R**eferences and **I**nput sentence (*SARI*), that better accounts for fidelity against multiple possible refernces. We use SARI to evaluate performance of our model and provides a brief description about it in the following section. We also use BLEU [69] and

Table 4.1: An example showing more than one correct fusion output.

Input	it has awesome battery life . it charges very slow .
Ground Truth	it has awesome battery life . **however** it charges very slow .
References	it has awesome battery life . **however** it charges very slow . it has awesome battery life . **but** it charges . it has awesome battery life . **on the other hand** it charges very slow .

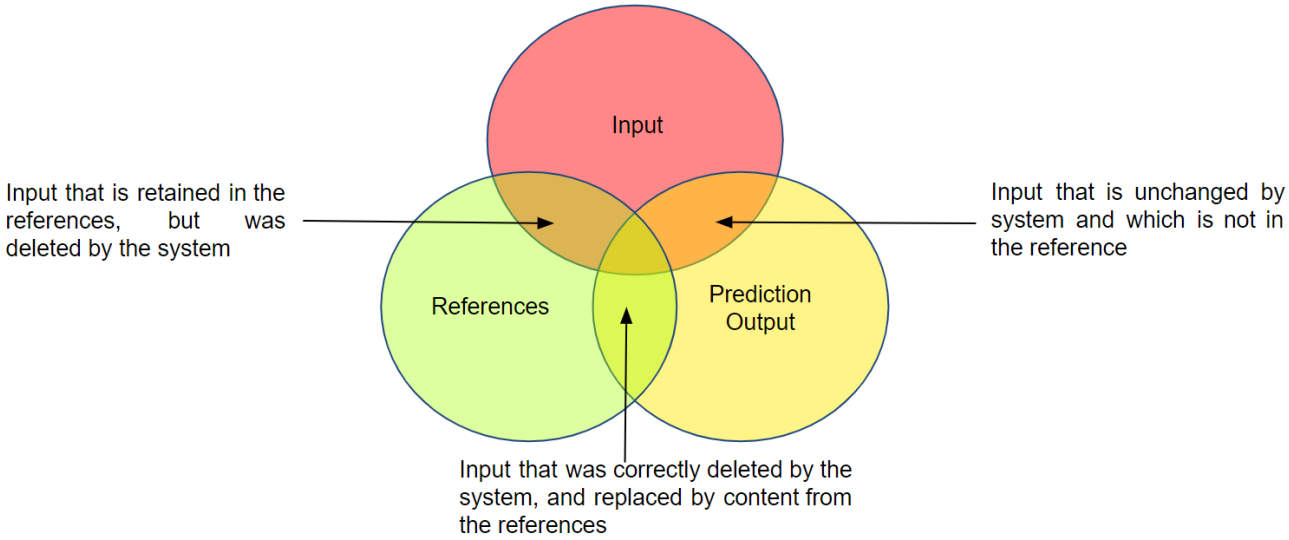

Figure 4.1: Venn Diagram represents different cases in SARI metric.

ROUGE [56] scores, which are standard metrics in NLP to measure quality of generation tasks.

SARI

In such tasks, words are added, deleted, or kept the same while generating a response. SARI is used to measure how good it is to perform these operations on n-grams. SARI consists of three sub–scores, namely Keep score, Add score, and Del score. As the name suggests, a Keep score measures how good a model is to keep n-grams in the prediction output which are also present in the input and references. Similarly, an Add score measures how good a model is to add new n-grams which are not present in the input but exist in the references. A Del score represents how good a model is to delete n-grams which are present in the input but not present in the references. Fig. 4.1 represents different cases in SARI evaluation.

4.3 Results and Analysis

Table 4.2: SARI, Corpus BLEU, and ROUGE scores on the small (Amazon) and large (Amazon + DiscoFuse) datasets.

	Amazon	Amazon + DiscoFuse
Avg SARI	0.7128	**0.7248**
Avg Keep	0.9706	**0.9766**
Avg Add	**0.2856**	0.2820
Avg Del	0.8822	**0.9157**
Rouge-4	0.7539	0.7439
Rouge-3	0.8058	0.7922
Rouge-2	0.8369	0.8251
Rouge-1	0.8902	0.8938
Rouge-l	0.9077	0.9109
Corpus Bleu	0.8773	0.8731

Overall Performance

Table 4.2 shows average SARI, Keep, Add, Del, Corpus BLEU, and ROUGE scores for models fine-tuned on a small and large dataset. As we can see, the SARI score is better for the model fine-tuned on the large dataset compared to the model fine-tuned on the small dataset. Our analysis shows that the model fine-tuned on the large dataset, containing 24 times more instances than the small one, learns to insert a single connecting string at the right place correctly. On the other hand, the model fine-tuned on the small dataset, sometimes inserts an incomplete connecting string, e.g., instead of "on the other hand", it inserts "the other hand", and sometimes it merges words from two connecting strings; e.g., it merges "but," with "some users have also mentioned that". Both of these cases are penalized by SARI. However, both models have almost same Corpus BLEU and ROUGE scores which suggests that most of the generated words are correct and are in order. Table 4.3 shows generated output corresponding to best SARI and Add scores. Our models fuse opinions in these examples perfectly with proper

Table 4.3: Correctly fused cases from models. These have been generated correctly by both models, fine-tuned on small and large datasets, separately.

1	Input	it rocks. it is extremely slow.
	Output	it rocks. **on the other hand, a few users have complained that** it is extremely slow.
	References	1. it rocks. however, there are people who have complained that it is extremely slow. 2. it rocks. but, it is extremely slow. 3. it rocks. on the other hand, it is extremely slow. 4. it rocks. however it is extremely slow. 5. it rocks. yet, some users have also mentioned that it is extremely slow. 6. although it rocks, according to a few users it is extremely slow. 7. it rocks. on the other hand, a few users have complained that it is extremely slow.
2	Input	battery is amazing. the battery life does not last a day.
	Output	battery is amazing. **however, there are people who have complained that** the battery life does not last a day.
	References	1. battery is amazing. however, there are people who have complained that the battery life does not last a day. 2. battery is amazing. but, the battery life does not last a day. 3. battery is amazing. on the other hand, the battery life does not last a day. 4. battery is amazing. however the battery life does not last a day. 5. battery is amazing. yet, some users have also mentioned that the battery life does not last a day. 6. although battery is amazing, according to a few users the battery life does not last a day. 7. battery is amazing. on the other hand, a few users have complained that the battery life does not last a day.
3	Input	this is a really great unit. battery life stinks.
	Output	this is a really great unit. **however, some users have also mentioned that** battery life stinks.
	References	1. this is a really great unit. however, there are people who have complained that battery life stinks. 2. this is a really great unit. but, battery life stinks. 3. this is a really great unit. on the other hand, battery life stinks. 4. this is a really great unit. however battery life stinks. 5. this is a really great unit. yet, some users have also mentioned that battery life stinks. 6. although this is a really great unit, according to a few users battery life stinks. 7. this is a really great unit. on the other hand, a few users have complained that battery life stinks.

Table 4.4: Failure cases from both generation models.

1	Input	it is a delight. you can not download apps on it.
	Output	it is a delight. **on** you can not download apps on it.
	References	1. it is a delight. however, there are people who have complained that you can not download apps on it. 2. it is a delight. but, you can not download apps on it. 3. it is a delight. on the other hand, you can not download apps on it. 4. it is a delight. however you can not download apps on it. 5. it is a delight. yet, some users have also mentioned that you can not download apps on it. 6. although it is a delight, according to a few users you can not download apps on it. 7. it is a delight. on the other hand, a few users have complained that you can not download apps on it.
2	Input	it runs fast. this is a really sad device.
	Output	it runs fast. **however,** is a really sad device.
	References	1. it runs fast. however, there are people who have complained that this is a really sad device. 2. it runs fast. but, this is a really sad device. 3. it runs fast. on the other hand, this is a really sad device. 4. it runs fast. however this is a really sad device. 5. it runs fast. yet, some users have also mentioned that this is a really sad device. 6. although it runs fast, according to a few users this is a really sad device. 7. it runs fast. on the other hand, a few users have complained that this is a really sad device.
3	Input	it looks perfect. it is ridiculous.
	Output	it looks perfect. **on,** is ridiculous.
	References	1. it looks perfect. however, there are people who have complained that it is ridiculous. 2. it looks perfect. but, it is ridiculous. 3. it looks perfect. on the other hand, it is ridiculous. 4. it looks perfect. however it is ridiculous. 5. it looks perfect. yet, some users have also mentioned that it is ridiculous. 6. although it looks perfect, according to a few users it is ridiculous. 7. it looks perfect. on the other hand, a few users have complained that it is ridiculous.

Table 4.5: SARI and other scores corresponding to best and worst generations. These scores correspond to the best and worst examples in Tables 4.3 and 4.4, respectively.

		SARI	Keep	Add	Del
Correctly Fused	1	0.8304	0.9909	0.5002	1
	2	0.8124	0.9929	0.4443	1
	3	0.7906	0.9909	0.3811	1
Failure Cases	1	0.6751	0.9929	0.0323	1
	2	0.4993	0.9155	0.1032	0.4791
	3	0.4543	0.8137	0.0701	0.4791

Table 4.6: Effect of references on Add score. This table shows that there is always more n-grams in references than in prediction output. For illustration purpose, we consider only two references, and only unigrams and bigrams.

Prediction Output	it is good. however it is costly.
References	it is good. however it is costly. it is good. but it is costly.
Additional Unigrams and Bigrams	**but, . but, but it**

connecting string. Example 2 in this table is an interesting case. In this example, opinions are connected with phrase *"however, some users have also mentioned that"*, which is not present in any of the references in the training, validation and test sets. Instead, it is a combination of two separate connecting strings, demonstrating that the model not only learned to fuse opposite opinions perfectly, but also to fuse connecting strings properly.

However, for these outputs, the Add score is not as good as other scores, as shown in Table 4.5. So, naturally, a question arises: **why is the Add score not higher, given perfect generation?** It happens due to how Add score is computed in the SARI metric. To compute Recall for the Add operation, n-grams from all connecting strings in references are considered along with n-grams corresponding to the connecting string in prediction output. Therefore, Add score does not reach a higher value. Table 4.6 shows an example where there are more number of n-grams in references than in prediction output. Here, for illustration purposes, we show only two responses in the references and consider only unigrams and bigrams. Table 4.4 shows instances corresponding to low average SARI, Keep, Add, and Del scores. In some of the instances, the model is unable to insert a complete connecting string. For the first instance shown in Table 4.4, the model inserts *"on"* instead of the whole connecting string *"on the other hand"*. In some cases, not only does the model fails to insert a complete connecting string, but it also incorrectly deletes words from the second opinion, as shown in the second example in Table 4.4. In other cases, although the model inserts the correct and complete connecting string, it also deletes a few words from the second opinion. Our analysis yields a plausible cause. Since our decoder is autoregressive and generates words on the basis of left context and previously generated words, we find that in the training dataset, for the same left context (i.e., initial substring), different connecting strings appear in different examples. We believe

Table 4.7: Exact match with one of the references. This table shows exact match of generated output on test dataset with one of the references.

	Exact Match	
	Amazon (20.08%)	Amazon+DiscoFuse (15.13%)
however	74	87
on the other hand, a few users have complained that	41	19
however, there are people who have complained that	27	17
but,	22	1
on the other hand,	17	23
yet, some users have also mentioned that	14	0

Table 4.8: This table compares quality of the generated responses by a rule-based system and the models trained on the small and large datasets. We can see that the rule-based generations outperform the model-based generations.

	Model Based (Small Dataset)	Model Based (Large Dataset)	Rule Based
Avg SARI	0.7128	0.7248	**0.7473**
Avg Keep	0.9706	0.9766	**0.9805**
Avg Add	0.2856	0.2820	**0.3264**
Avg Del	0.8822	0.9157	**0.935**

that such instances confuse the model and lead to such failure cases. We also compare the quality of responses generated by a trained model and a rule–based system. In the rule–based system, we follow the same rules to generate responses which we use to generate the training dataset. Table 4.8 shows SARI and its sub-scores corresponding to the responses generated by rules and by the models trained on the small and large datasets. We see that the rule-based system outperforms the model-based systems. The rule-based system uses templates to produce responses and generates perfect outputs. On the other hand, the model-based responses suffer from incorrect generations and mixing of discourse connectives. Hence, the corresponding SARI and sub-scores are lower compared against the rule-based system.

Study on Connective Prediction

Table 4.7 shows the percentage of instances in the test dataset for which prediction output matches exactly with one of the references. For the model fine-tuned on the small dataset, around 20% instances in the test dataset have an exact match and for the model fine-tuned on

Table 4.9: Percentage of Instances with new connecting strings. Connecting strings present in this table are not present in the training dataset.

New Connecting String	Amazon (9.88%)
however,	90
yet,	5
yet	1

Table 4.10: Examples of correct compositions of two strings. This table represents that model learns to connect words from two connecting strings. These mixed connecting strings are not present in the training dataset.

Correctly composed Connecting String	Amazon
but, some users have also mentioned that	107 11.01%
but, there are people who have complained that	
however, some users have also mentioned that	
proper merging with input text	

the large dataset, around 15% instances in the test dataset have an exact match. We perform a comprehensive manual analysis for all the test instances for which prediction output does not match with any of the corresponding references. Table 4.9 shows new connecting strings generated by the model trained on the small dataset. These connecting strings are not present in the training instances. The model learns to either append a punctuation with a discourse connective or insert a proper substring of a multi–word discourse connective. For example, "yet" is taken from the connecting string "yet, some users have also mentioned that", and "," is inserted along with "however" to generate a new connecting string "however,".

On the other hand, Table 4.10 shows instances of connecting strings which are generated by the model, trained on the small dataset, by merging two connecting strings present in the training dataset. For example, "but, some users have also mentioned that" is not present in the training instances. Our model generates "but," from "but," and "some users have also mentioned that" from the "yet, some users have also mentioned that" connecting string, successfully merging both. Around 11.07% instances in the test dataset produce output containing correctly composed connecting strings.

Table 4.11 shows some of the output instances which are generated by the model trained on the small dataset, with correctly composed connecting strings. Unfortunately, our model trained over the large dataset does not produce such interesting responses. Since the large dataset

Table 4.11: Correctly composed example. This table presents some example of test instances which have been connected by the model correctly. "I" represents an input, and "O" an output.

I	it is fine. it does not have the drive.
O	it is fine. **but,** does not have the drive.
I	this pc is incredible. it is not good.
O	this pc is incredible. **but, some users have also mentioned that** it is not good.
I	it's pretty great. it's not perfect.
O	it's pretty great. **but, there are people who have complained that** it's not perfect.
I	as an entry - level laptop, this one is very good. the speakers are kind of bad.
O	as an entry - level laptop, this one is very good. **however,** speakers are kind of bad.
I	the keyboard is great and fast. it has some limitations.
O	the keyboard is great and fast. **however, some users have also mentioned that** it has some limitations.

Table 4.12: This table represents distribution of main error types on test dataset from model trained on Amazon dataset. In most of the failure cases either a word 'on' is used to fuse sentences or words of discourse connectives are repeated in the prediction output. In the other failure cases either mixing is done incorrectly or input opinions are modified.

Error Type	Amazon
Insertion of only word 'on'	16.37%
Repetition of words	10.71%
Incorrect mixing	3.08%
Opinion text modification	2.78%

contains more than $400,000$ instances, we believe that model is overly fine-tuned on such a large dataset to produce outputs with connecting strings present in the training dataset. Table 4.12 shows the main error types present in the test instances. In most failure cases, a single word 'on' is inserted to fuse the two sentences. In the second most of the failure cases, discourse-connective words are repeated. The third one relates to incorrect composition cases, which leads to 3.08% failure cases for the small dataset. In such failure cases, although the model is able to insert a connecting string at the right place, it deletes or inserts word[s] incorrectly. As shown in the first two examples in Table 4.13, in the first, the model inserts connecting string "but" at the correct position but incorrectly merges it with "on the other hand". This merging does not include the word "on", hence an example of correct insertion but incorrect composition. In the second example, the model inserts connecting string at the right place, but it inserts "it" incorrectly in between when merging the two connecting strings. The next two examples belong to incorrect modification of input opinion text. As shown in Table 4.12, this constitutes around

Table 4.13: Two examples of each error type. The first two represent incorrect composition where either an extra word is present or a word is missing. The last two show incorrect modification of words from input opinion text. All incorrect words are bolded.

Correct Insertion + Wrong Mixing	I	it has great sound. it is not as good as the tab 2.
	O	it has great sound. **but the** other hand, it is not as good as the tab 2.
	I	Its very responsive. It's infuriating.
	O	Its very responsive. on the other hand, **it** few users have complained that its infuriating.
Input Opinion modification	I	The speed and graphics are great. It's loud and clunky.
	O	The speed and graphics are great. on the other hand, **a's** loud and clunky.
	I	It's great. This is a waste of $300.
	O	**It** it's great, according to a few users this is a waste of $300.

2.78% failure cases for model trained on small dataset. The last two examples in Table 4.13 depict failure cases of this type. As we can see in the second last example, "it's" is replaced with "a's" incorrectly and in the last example, and an additional "it" is added incorrectly before the positive opinion.

Chapter 5

Conclusion and Future Work

The analysis and summarization of opinions, expressed by online user reviews, is an important language processing task which can benefit both customers and product sellers alike. The rapid increase of online review content has increased the demand for systems to aggregate opinions automatically. Such large-scale data also enables the collection of sufficient data to train deep learning algorithms. Currently, template-based comparison functionality only compares seller–provided product information, i.e., characteristics, price, etc. This comparison mechanism does not include experienced customers' opinions. QA systems on reviews provide one side of the story, i.e., positive or negative information. Such systems do not provide a combined response containing both sides of opinions regarding a product. To handle this problem, our thesis attempts to extract positive and negative opinions, separately, from unannotated Amazon reviews and train a neural language model to generate comparative responses expressing both sides of opinions. Our work is motivated by two main research questions:

1. Can we extract fine-grained opinion sentences (i.e., sentences containing either positive or negative opinions, but not both) from unannotated Amazon reviews?

A significant portion of our thesis is devoted to gathering fine-grained positive and negative opinions from reviews. Only extracting sentences from reviews is not sufficient. A review sentence may contain both positive and negative opinions regarding different aspects of the same product. Therefore, we extract Elementary Discourse Units (EDUs) from reviews to extract

fine-grained opinion sentences. Then we learn sentence representation and latent code spaces by training an unsupervised sentence reconstruction task. We then extract positive and negative opinions separately by mapping fine-grained opinions to latent codes and extracting opinions corresponding to popular codes. We manually remove opinion sentences from summaries which do not contain at least one noun or pronoun and at least one main or auxiliary verb, and which do not express valid meaning.

2. Can we combine a positive and a negative opinion to generate a comparative response about a product?

In the latter half of our thesis, we focus on generating a comparative response. We initialize our encoder and decoder with publicly available BERT pre-trained weights, then apply the encoder–decoder mechanism to train our model. We train our model on generated dataset. Evaluation on our test dataset shows promising results. We also discussed in detail some failure scenarios which we plan to handle in our future work.

Possible future work directions are many. Here we mention some of the limitations of our work and discuss how these may lead to possible new research directions.

Joint Modeling of Sentiment and Opinion Summarization. In our thesis, we fine-tune BERT [22] to classify opinions into positive and negative categories. Then we pass these positive and negative opinions separately into extractive opinion summarization model [3]. Previous works show the benefit of jointly modeling sentiment and aspect in review-related tasks [92, 13, 105, 53, 24]. The success of this work can inspire jointly modeling sentiment and opinion summarization. Our summarization model follows unsupervised training; however, our sentiment classification model is supervised. Previous work on unsupervised sentence sentiment analysis shows promising results [66, 32, 48, 14]. Combining an unsupervised sentiment analysis task with an unsupervised opinion summarization task may lead to new exciting research. Such joint learning not only would reduce human annotation time but also may result in a new state-of-the-art in respective tasks.

Combining Abstractive Opinions for Response Generation. Neural language models show promising results on text generation tasks, including machine translation [6, 82], document summarization [17, 62], data-to-text generation [72] and sentence simplification [104]. Our thesis generates extractive opinion summaries from reviews. Previous works use graph-based approach [35, 36] and fully supervised approach [98] to generate abstractive summaries. Sometimes an extractive summary generation model generates incoherent summaries lacking fluency. On the other hand, an abstractive model may generate well-formed summaries using extracted opinions. Either a template-based approach, similar to [36], or a generation-based approach can be used. A neural model-based text generation model can encode a context of an opinion (e.g., the whole review) and then try to generate the opinion conditioned on the same context and specific aspect and polarity.

Connecting String as Context. Providing context along with input is beneficial in multiple Natural Language Processing tasks [55, 96]. In our thesis, for generating a comparative response, we only pass a positive and a negative opinion. At training time, we do not show which connecting string to be used in the input. [8] shows the benefit of jointly learning connecting string classifier and fused sentence generation in sentence fusion task. In our opinion, passing the connecting string along with input helps to increase the accuracy of response generation. A few failure cases in our work are associated with incomplete discourse connective in output response. We believe that such cases may be handled by providing a discourse connecting string as context.

Bibliography

[1] Stefanos Angelidis and Mirella Lapata. "Multiple Instance Learning Networks for Fine-Grained Sentiment Analysis". In: *Transactions of the Association for Computational Linguistics* 6 (2018), pp. 17–31. DOI: 10.1162/tacl_a_00002. URL: https://aclanthology.org/Q18-1002.

[2] Stefanos Angelidis and Mirella Lapata. "Summarizing Opinions: Aspect Extraction Meets Sentiment Prediction and They Are Both Weakly Supervised". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 3675–3686. DOI: 10.18653/v1/D18-1403. URL: https://aclanthology.org/D18-1403.

[3] Stefanos Angelidis et al. *Extractive Opinion Summarization in Quantized Transformer Spaces*. 2020. arXiv: 2012.04443 [cs.CL].

[4] Stefanos Angelidis et al. "Extractive Opinion Summarization in Quantized Transformer Spaces". In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 277–293.

[5] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. "SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining." In: vol. 10. Jan. 2010.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR* abs/1409.0473 (2015).

[7] Regina Barzilay and Kathleen R. McKeown. "Sentence Fusion for Multidocument News Summarization". In: *Computational Linguistics* 31.3 (2005), pp. 297–328. DOI: 10.1162/089120105774321091. URL: https://aclanthology.org/J05-3002.

[8] Eyal Ben-David et al. "Semantically Driven Sentence Fusion: Modeling and Evaluation". In: *FINDINGS*. 2020.

[9] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation". In: *ArXiv* abs/1308.3432 (2013).

[10] Lidong Bing et al. "Abstractive Multi-Document Summarization via Phrase Selection and Merging". In: *ACL*. 2015.

[11] Chloé Braud and Pascal Denis. "Learning Connective-based Word Representations for Implicit Discourse Relation Identification". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 203–213. DOI: 10.18653/v1/D16-1020. URL: https://aclanthology.org/D16-1020.

[12] Arthur Bražinskas, Mirella Lapata, and Ivan Titov. "Unsupervised Opinion Summarization as Copycat-Review Generation". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 5151–5169. DOI: 10.18653/v1/2020.acl-main.461. URL: https://aclanthology.org/2020.acl-main.461.

[13] Samuel Brody and Noémie Elhadad. "An Unsupervised Aspect-Sentiment Model for Online Reviews". In: *NAACL*. 2010.

[14] Samuel Brody and Noemie Elhadad. "An unsupervised aspect-sentiment model for online reviews". In: *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics*. 2010, pp. 804–812.

[15] Yllias Chali, Moin Mahmud Tanvee, and Mir Tafseer Nayeem. "Towards Abstractive Multi-Document Summarization Using Submodular Function-Based Framework, Sentence Compression and Merging". In: *IJCNLP*. 2017.

[16] Danqi Chen and Christopher D. Manning. “A Fast and Accurate Dependency Parser using Neural Networks”. In: *EMNLP*. 2014.

[17] Jianpeng Cheng, Li Dong, and Mirella Lapata. “Long Short-Term Memory-Networks for Machine Reading”. In: *EMNLP*. 2016.

[18] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”. In: *EMNLP*. 2014.

[19] Eric Chu and Peter Liu. “MeanSum: A Neural Model for Unsupervised Multi-Document Abstractive Summarization”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 1223–1232. URL: https://proceedings.mlr.press/v97/chu19b.html.

[20] Junyoung Chung et al. “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. In: *ArXiv* abs/1412.3555 (2014).

[21] Andrew M. Dai and Quoc V. Le. “Semi-supervised Sequence Learning”. In: *NIPS*. 2015.

[22] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[23] Giuseppe Di Fabbrizio, Amanda Stent, and Robert Gaizauskas. “A Hybrid Approach to Multi-document Summarization of Opinions in Reviews”. In: *Proceedings of the 8th International Natural Language Generation Conference (INLG)*. Philadelphia, Pennsylvania, U.S.A.: Association for Computational Linguistics, June 2014, pp. 54–63. DOI: 10.3115/v1/W14-4408. URL: https://aclanthology.org/W14-4408.

[24] Qiming Diao et al. “Jointly Modeling Aspects, Ratings and Sentiments for Movie Recommendation (JMARS)”. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’14. New York, New York, USA: Association for Computing Machinery, 2014, pp. 193–202. ISBN: 9781450329569. DOI: 10.1145/2623330.2623758. URL: https://doi.org/10.1145/2623330.2623758.

[25] Qiming Diao et al. “Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS)”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014).

[26] Wenjing Duan, Bin Gu, and Andrew Whinston. “The Dynamics of Online Word-of-Mouth and Product Sales: An Empirical Investigation of the Movie Industry”. In: *eBusiness & eCommerce* (2008).

[27] Chris Dyer et al. “Transition-Based Dependency Parsing with Stack Long Short-Term Memory”. In: *ACL*. 2015.

[28] Michael Elhadad and Kathleen R. McKeown. “Generating Connectives”. In: *COLING 1990 Volume 3: Papers presented to the 13th International Conference on Computational Linguistics*. 1990. URL: https://aclanthology.org/C90-3018.

[29] Micha Elsner and Deepak Santhanam. “Learning to Fuse Disparate Sentences”. In: *Proceedings of the Workshop on Monolingual Text-To-Text Generation*. Portland, Oregon: Association for Computational Linguistics, June 2011, pp. 54–63. URL: https://aclanthology.org/W11-1607.

[30] Vanessa Wei Feng and Graeme Hirst. “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)*. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 511–521. DOI: 10.3115/v1/P14-1048. URL: https://aclanthology.org/P14-1048.

[31] Vanessa Wei Feng and Graeme Hirst. “Two-pass Discourse Segmentation with Pairing and Global Features”. In: *ArXiv* abs/1407.8215 (2014).

[32] Milagros Fernández-Gavilanes et al. “Unsupervised method for sentiment analysis in online texts”. In: *Expert Systems with Applications* 58 (2016), pp. 57–75.

[33] Katja Filippova. “Multi-Sentence Compression: Finding Shortest Paths in Word Graphs”. In: *COLING*. 2010.

[34] Seeger Fisher and Brian Roark. “The utility of parse-derived features for automatic discourse segmentation”. In: *ACL*. 2007.

[35] Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. “Opinosis: A Graph Based Approach to Abstractive Summarization of Highly Redundant Opinions”. In: *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Beijing, China: Coling 2010 Organizing Committee, Aug. 2010, pp. 340–348. URL: https://aclanthology.org/C10-1039.

[36] Shima Gerani et al. “Abstractive Summarization of Product Reviews Using Discourse Structure”. In: *EMNLP*. 2014.

[37] Mor Geva et al. “DiscoFuse: A Large-Scale Dataset for Discourse-Based Sentence Fusion”. In: *NAACL*. 2019.

[38] “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation”. In: *ArXiv* abs/1609.08144 (2016).

[39] Brigitte Grote and Manfred Stede. “Discourse Marker Choice in Sentence Planning”. In: *INLG*. 1998.

[40] Mansi Gupta et al. “AmazonQA: A Review-Based Question Answering Task”. In: *IJCAI*. 2019.

[41] Dan Hendrycks and Kevin Gimpel. “Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units”. In: *ArXiv* abs/1606.08415 (2016).

[42] Hugo Hernault et al. “HILDA: A Discourse Parser Using Support Vector Machine Classification”. In: *Dialogue Discourse* 1 (2010), pp. 1–33.

[43] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9 (1997), pp. 1735–1780.

[44] Ari Holtzman et al. *The Curious Case of Neural Text Degeneration*. 2020. arXiv: 1904.09751 [cs.CL].

[45] Jeremy Howard and Sebastian Ruder. “Fine-tuned Language Models for Text Classification”. In: *ArXiv* abs/1801.06146 (2018).

[46] Minqing Hu and Bing Liu. "Mining and summarizing customer reviews". In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (2004).

[47] Yacine Jernite, Samuel R. Bowman, and David A. Sontag. "Discourse-Based Objectives for Fast Unsupervised Sentence Representation Learning". In: *ArXiv* abs/1705.00557 (2017).

[48] Salud M Jiménez-Zafra et al. "Combining resources to improve unsupervised sentiment analysis at aspect-level". In: *Journal of Information Science* 42.2 (2016), pp. 213–229.

[49] Shafiq R. Joty, Giuseppe Carenini, and Raymond T. Ng. "A Novel Discriminative Framework for Sentence-Level Discourse Analysis". In: *EMNLP*. 2012.

[50] Nal Kalchbrenner and Phil Blunsom. "Recurrent Convolutional Neural Networks for Discourse Compositionality". In: *CVSM@ACL*. 2013.

[51] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. DOI: 10.3115/v1/D14-1181. URL: https://aclanthology.org/D14-1181.

[52] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2015).

[53] Angeliki Lazaridou, Ivan Titov, and Caroline Sporleder. "A Bayesian Model for Joint Unsupervised Induction of Sentiment, Aspect and Discourse Representations". In: *ACL*. 2013.

[54] Quoc V. Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents". In: *ArXiv* abs/1405.4053 (2014).

[55] Huayu Li et al. "A context-aware attention network for interactive question answering". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017, pp. 927–935.

[56] Chin-Yew Lin. “ROUGE: A Package for Automatic Evaluation of Summaries”. In: *ACL 2004*. 2004.

[57] Zhouhan Lin et al. “A Structured Self-attentive Sentence Embedding”. In: *ArXiv* abs/1703.03130 (2017).

[58] Yang Liu et al. “Implicit Discourse Relation Classification via Multi-Task Neural Networks”. In: *ArXiv* abs/1603.02776 (2016).

[59] Eric Malmi et al. “Automatic Prediction of Discourse Connectives”. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), May 2018. URL: https://aclanthology.org/L18-1260.

[60] Eric Malmi et al. “Encode, Tag, Realize: High-Precision Text Editing”. 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 5054–5065. DOI: 10.18653/v1/D19-1510. URL: https://aclanthology.org/D19-1510.

[61] William C. Mann and Sandra A. Thompson. “Rhetorical Structure Theory: Toward a functional theory of text organization”. In: *Text & Talk* 8 (1988), pp. 243–281.

[62] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. “Don’t Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization”. In: *EMNLP*. 2018.

[63] Jianmo Ni, Jiacheng Li, and Julian McAuley. “Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects”. In: *EMNLP*. 2019.

[64] Allen Nie, Erin Bennett, and Noah Goodman. “DisSent: Learning Sentence Representations from Explicit Discourse Relations”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 4497–4510. DOI: 10.18653/v1/P19-1442. URL: https://aclanthology.org/P19-1442.

[65] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. "Neural Discrete Representation Learning". In: *NIPS*. 2017.

[66] Georgios Paltoglou and Mike Thelwall. "Twitter, MySpace, Digg: Unsupervised sentiment analysis in social media". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 3.4 (2012), pp. 1–19.

[67] Bo Pang and Lillian Lee. "Opinion Mining and Sentiment Analysis". In: *Found. Trends Inf. Retr.* 2 (2007), pp. 1–135.

[68] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up? Sentiment Classification using Machine Learning Techniques". In: *EMNLP*. 2002.

[69] Kishore Papineni et al. "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *ACL*. 2002.

[70] Ankur P. Parikh et al. "A Decomposable Attention Model for Natural Language Inference". In: *EMNLP*. 2016.

[71] Romain Paulus, Caiming Xiong, and Richard Socher. "A Deep Reinforced Model for Abstractive Summarization". In: *ArXiv* abs/1705.04304 (2018).

[72] Laura Perez-Beltrachini and Mirella Lapata. "Bootstrapping Generators from Noisy Data". In: *NAACL*. 2018.

[73] Matthew E. Peters et al. "Deep Contextualized Word Representations". In: *NAACL*. 2018.

[74] Emily Pitler et al. "Easily Identifiable Discourse Relations". In: *Coling 2008: Companion volume: Posters*. Manchester, UK: Coling 2008 Organizing Committee, Aug. 2008, pp. 87–90. URL: https://aclanthology.org/C08-2022.

[75] Ana-Maria Popescu and Oren Etzioni. "Extracting Product Features and Opinions from Reviews". In: *HLT*. 2005.

[76] Lianhui Qin et al. "Adversarial Connective-exploiting Networks for Implicit Discourse Relation Classification". In: *ACL*. 2017.

[77] Alec Radford and Karthik Narasimhan. "Improving Language Understanding by Generative Pre-Training". In: 2018.

[78] Anna Rohrbach et al. "Object Hallucination in Image Captioning". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 4035–4045. DOI: 10.18653/v1/D18-1437. URL: https://aclanthology.org/D18-1437.

[79] Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. "Leveraging Pre-trained Checkpoints for Sequence Generation Tasks". In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 264–280.

[80] Aurko Roy and David Grangier. "Unsupervised Paraphrasing without Translation". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 6033–6039. DOI: 10.18653/v1/P19-1605. URL: https://aclanthology.org/P19-1605.

[81] Aurko Roy et al. "Theory and Experiments on Vector Quantized Autoencoders". In: *ArXiv* abs/1805.11063 (2018).

[82] Rico Sennrich, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units". In: *ArXiv* abs/1508.07909 (2016).

[83] Tian Shi et al. "Neural Abstractive Text Summarization with Sequence-to-Sequence Models". In: *ACM/IMS Trans. Data Sci.* 2.1 (Jan. 2021). ISSN: 2691-1922. DOI: 10.1145/3419106. URL: https://doi.org/10.1145/3419106.

[84] Richard Socher et al. "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. URL: https://aclanthology.org/D13-1170.

[85] Richard Socher et al. "Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions". In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, July 2011, pp. 151–161. URL: https://aclanthology.org/D11-1014.

[86] Radu Soricut and Daniel Marcu. "Sentence Level Discourse Parsing using Syntactic and Lexical Information". In: *NAACL*. 2003.

[87] Rajen Subba and Barbara Maria Di Eugenio. "Automatic Discourse Segmentation using Neural Networks". In: 2007.

[88] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to Sequence Learning with Neural Networks". In: *NIPS*. 2014.

[89] Maite Taboada et al. "Lexicon-Based Methods for Sentiment Analysis". In: *Computational Linguistics* 37 (June 2011), pp. 267–307. DOI: 10.1162/COLI_a_00049.

[90] Duyu Tang, Bing Qin, and Ting Liu. "Document Modeling with Gated Recurrent Neural Network for Sentiment Classification". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1422–1432. DOI: 10.18653/v1/D15-1167. URL: https://aclanthology.org/D15-1167.

[91] Kapil Thadani and Kathleen McKeown. "Supervised Sentence Fusion with Single-Stage Inference". In: *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. Nagoya, Japan: Asian Federation of Natural Language Processing, Oct. 2013, pp. 1410–1418. URL: https://aclanthology.org/I13-1198.

[92] Ivan Titov and Ryan T. McDonald. "Modeling online reviews with multi-grain topic models". In: *ArXiv* abs/0801.1063 (2008).

[93] Jenine Turner and Eugene Charniak. "Supervised and Unsupervised Learning for Sentence Compression". In: *ACL*. 2005.

[94] Peter D. Turney. "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews". In: *ACL*. 2002.

[95] Ashish Vaswani et al. "Attention is All you Need". In: *ArXiv* abs/1706.03762 (2017).

[96] Elena Voita et al. "Context-aware neural machine translation learns anaphora resolution". In: *arXiv preprint arXiv:1805.10163* (2018).

[97] Hanna M Wallach. "Conditional random fields: An introduction". In: *Technical Reports (CIS)* (2004), p. 22.

[98] Lu Wang and Wang Ling. "Neural Network-Based Abstract Generation for Opinions and Arguments". In: *NAACL*. 2016.

[99] Janyce Wiebe, Theresa Wilson, and Claire Cardie. "Annotating Expressions of Opinions and Emotions in Language". In: *Language Resources and Evaluation (formerly Computers and the Humanities)* 39 (May 2005), pp. 164–210. DOI: 10.1007/s10579-005-7880-9.

[100] Wei Xu et al. "Optimizing Statistical Machine Translation for Text Simplification". In: *Transactions of the Association for Computational Linguistics* 4 (2016), pp. 401–415. DOI: 10.1162/tacl_a_00107. URL: https://aclanthology.org/Q16-1029.

[101] Zichao Yang et al. "Hierarchical Attention Networks for Document Classification". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 1480–1489. DOI: 10.18653/v1/N16-1174. URL: https://aclanthology.org/N16-1174.

[102] Qiang Ye, Rob Law, and Bin Gu. "The impact of online user reviews on hotel room sales". In: *International Journal of Hospitality Management* 28 (2009), pp. 180–182.

[103] Biao Zhang et al. "Shallow Convolutional Neural Network for Implicit Discourse Relation Recognition". In: *EMNLP*. 2015.

[104] Xingxing Zhang and Mirella Lapata. "Sentence Simplification with Deep Reinforcement Learning". In: *EMNLP*. 2017.

[105] Zhi-Min Zhou et al. "The Effects of Discourse Connectives Prediction on Implicit Discourse Relation Recognition". In: *SIGDIAL Conference*. 2010.