Master's Thesis

Text Generation Model Enhanced with Semantic Information in Aspect Category Sentiment Analysis

TRAN, Tu Dinh

Supervisor Kiyoaki Shirai

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Information Science)

September, 2023

Abstract

Sentiment Analysis (SA) is a task of identifying opinions expressed in a piece of text, especially in order to determine whether the writer's attitude toward a particular topic or product is positive, negative, or neutral. Although SA is performed on various types of texts such as a document and a sentence, recently, people have paid more attention to aspect-level sentiment analysis. Aspect Category Sentiment Analysis (ACSA), which is one of the main subtasks of Aspect-based Sentiment Analysis, intends to detect the polarity of the conveying emotions on the aspect within the input text. It is helpful to understand the writer's opinions in detail especially when a review sentence contains multiple aspects with different polarity.

The typical approaches solve ACSA as a text classification task. They concentrate on improving the quality of the representation of contextual information by using better language models and extracting selectively aspect-related information with attention mechanism. However, some previous studies point out that fine-tuning language models for text classification is not effective since the majority of language models are pre-trained on text generation tasks. Therefore, a method called "BART generation" is proposed to solve ACSA as the text generation task, which is based on the outstanding language model called Bidirectional and Auto-Regressive Transformers (BART). This model accepts review sentences as input and generates target sentences that clearly express the polarity toward a given aspect. The target sentence is yielded by filling an aspect category and sentiment word into a predefined template. It outperformed the other classification models for ACSA. However, BART generation faces difficulty in capturing relations between opinion words and aspect words as well as extracting aspect-related information in sentences containing multiple aspects.

To solve these problems, the goal of this study is to propose a method that leverages Abstract Meaning Representation (AMR) for capturing relations between opinion words and aspect words as well as enhancing the aspect-related information extraction within the BART generation method. AMR is the semantic representation that expresses the meaning of a sentence as a rooted, directed, and labeled graph. AMR can provide a better way to model word relations that are difficult to extract within a sentence. Besides, since AMR assigns the same graph for multiple sentences with the same meaning, it can help to alleviate data sparsity in ACSA. To encode the AMR graph, Graph Attention Networks (GAT) is used to obtain the embedding of

nodes, which is updated by applying the attention mechanism. A new Cross-Attention module for AMR is added in each decoder layer to incorporate the semantic information in the AMR graph into our text decoding phase. In this study, the pre-trained AMRBART is used as the AMR parser to obtain the AMR graph for the given review sentence. To calculate the attention to the AMR, the nodes in the AMR graph and the words in the sentence should be aligned. This alignment is determined by the pre-trained AMR aligner LEAMR.

Furthermore, two new regularizers are introduced to improve the allocation of Cross-Attention weights over the AMR graph. The first one is the identical regularizer, which compels the Cross-Attention weights of AMR nodes and the Cross-Attention weights of their aligned words in the review sentence to be equal as much as possible. The second is the entropy regularizer, which enables the model to only pay attention to a few AMR-related nodes. With the new regularizers, we expect to help the model correctly extract aspect-related information from the AMR graph

Since our model follows BART generation, it is based on the Transformer framework with a stack of encoder-decoder layers whose parameters are initialized by the pre-trained language model BART. However, the parameters of the newly introduced GAT module and AMR Cross-Attention layers should be initialized randomly. In practice, we find it challenging to fine-tune our model consisting of the modules initialized by the different ways. Therefore, we propose to pre-train the whole model again using in-domain texts. The pre-training is done by text-denoising task that is used for the pre-training of BART. That is, our model is pre-trained to reconstruct a complete review sentence from a corrupted sentence that is created by applying token masking, text infilling, and text deletion.

We evaluate the performance of our model on three datasets: Rest14, Rest14-hard, and MAMS. Rest14 is a dataset of restaurant reviews, where the most of reviews have only one aspect. Rest14-hard is a modified version of the Rest14 dataset that only includes sentences with multiple aspects in the test set. The training and development sets are the same as the original Rest14 dataset. Since the test set of Rest14-hard is small, we also adopt MAMS where all sentences in the training, development, and test sets contain multiple aspects. For each dataset, the proposed model is trained and evaluated five times with different random initialization of the parameters. In terms of evaluation metric, we utilize mean accuracy together with standard deviation which is denoted as "mean±(std)".

Our model achieves 91.2 ± (0.258), 78.1 ± (0.258), and 84.6 ± (0.453), which outperforms the state-of-the-art method (BART generation) by 0.7, 0.7, 1.5 percentage points on Rest14, Rest14-hard, and MAMS, respectively.

These results prove the effectiveness of incorporating the semantic information from AMR into the text generation model. In addition, the large improvement in MAMS proves that our regularizers can help the model to extract useful aspect-related information more appropriately even when there are multiple aspects in a sentence.

The ablation study is conducted to fully investigate the effect of each module within our model. We remove the identical regularizer, the entropy regularizer, both regularizers or pre-training procedure, then train the models. Compared to the full model, the accuracy scores of all ablated models are degraded. It means that each component plays an important role in the overall performance of our model. Additionally, we visualize the attention scores of AMR Cross-Attention layers in our full model and the model without both regularizers for a few sentences. It shows that our regularizers successfully increase the attention weights over important nodes. Furthermore, we conduct error analysis. The results indicate that our model faces obstacles in extracting aspect-related information for coarse-grained aspects like "miscellaneous".

Despite the promising results, our model still has some problems. The main challenge is to alleviate the gap between the pre-trained language model and the new components that are added to incorporate semantic information from AMR. Although the pre-training using in-domain texts is applied to tackle this problem, it seems insufficient. Further studies should concentrate on overcoming this limitation to achieve stable performance. In addition, the potential of our approach to incorporate semantic information given by AMR into the deep learning model should be further investigated. That is, we will evaluate the performance of our model in other subtasks of sentiment analysis such as Aspect Category Detection or Aspect-based Sentiment Analysis.

Acknowledgements

I would like to express my deepest appreciation to my supervisor, Professor Kiyoaki Shirai, for his guidance, support, and dedication throughout my Master's degree program. I am grateful for his deep knowledge and insights, which have helped me to make significant progress in my research. With his mentorship, patience, and encouragement, I was able to achieve my work goals and broaden my horizons.

I would like to extend my thanks to my second supervisor Professor NGUYEN Minh Le, my supervisor for minor research Professor SAKTI Sakriani Watiasri, and Professor KERTKEIDKACHORN Natthawut for their helps, support, and valuable suggestions for my research.

I would also like to express my sincere appreciation to the Japanese Ministry of Education, Culture, Sports, Science, and Technology (MEXT) for their support in the form of the MEXT scholarship.

Finally, I am deeply thankful to my family for their unconditional love, support, and understanding.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the background and goal of this research. We first introduce the background of our research in Section 1.1. Then, Section 1.2 establishes the motivations and goals of our study. Finally, Section 1.3 provides the general outline of this thesis.

1.1 Background

With the popularity of the Internet, people express their opinions or emotions every day through social media like Facebook, Twitter, and Instagram, or write their reviews about products or services on e-commerce websites such as Amazon, Rakuten, etc. Sentiment analysis (SA), which is the task of extracting opinions and sentiments (positive, negative, neutral) from a text, can assist decision-making on those platforms. SA is applied to different levels of the text which are a document, sentence, and aspect. Document-level and sentence-level sentiment analysis concentrate on detecting the general sentiment over a whole paragraph or a sentence. Since each sentence can contain different aspects with totally opposite sentiments, aspect-based sentiment analysis (ABSA) is necessary to detect the sentiments at the fine-grained level.

ABSA includes many subtasks such as Aspect Category Detection (ACD) and Aspect Category Sentiment Analysis (ACSA). ACD aims to identify the aspect category of a sentence or document while ACSA classifies the sentiment polarity of those categories. In Figure 1.1, with a review sentence "*The staff are rude but food is great*", we predict two aspect categories *service* and *food* in ACD while we classify the polarity of *service* and *food* as negative and positive respectively in ACSA.

In ACSA, modeling the relations between aspect and opinion words is es-

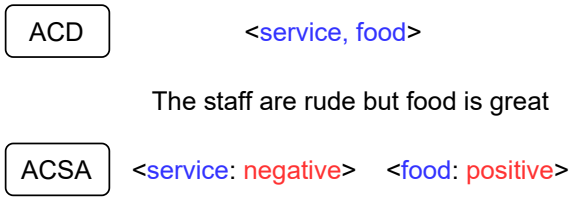


Figure 1.1: An example of ACD and ACSA.

sential to classify the right sentiment for each given aspect. However, models can find it hard to learn those relations from a plain text. Therefore, using Abstract Meaning Representation (AMR) [5], which is semantic representation language by a directed graph, can be a promising approach to take the relations between words into account. For example, Figure 1.2 shows the direct relation of node *rude-01* to node *staff* which corresponds to the relation of opinion word *rude* and aspect word *staff*. This direct relation helps the model easily predict the sentiment polarity *negative* of aspect category *service*.

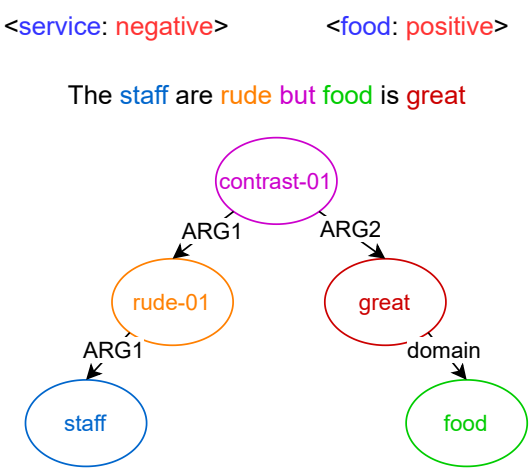


Figure 1.2: An example of ACSA. The word in the review sentence corresponds to the node in the same color in AMR.

1.2 Goal

This research aims to incorporate the relations of target and opinion words via AMR into the text generation model [33] for ACSA. Besides, since AMR attempts to represent sentences with the same meaning by the same struc-

ture, it can help to reduce data sparsity. This study also concentrates on extracting the correct information for each aspect category based on designing two novel regularizers on AMR cross-attention layers. We notice that AMR nodes and words in the review sentence having common semantics should receive the same attention from our model. Therefore, we minimize the difference between the attention weights of two Cross-Attentions layers over aligned AMR nodes and words. Furthermore, for each decoded token, our model should only pay attention to its associated AMR nodes, which is obtained by minimizing the entropy of attention weights of AMR cross-attention layers. The contribution of our study can be summarized as follows:

- We propose a model that integrates an AMR graph encoder into an autoregressive pre-trained language model for capturing relations between aspect and opinion words and applying this semantic information for ACSA.
- We introduce two novel regularizers to improve the cross-attention mechanism over the AMR graph with AMR alignments and information entropy.
- We investigate the effectiveness of our proposed method on three public benchmark datasets and surpass state-of-the-art models.

1.3 Thesis Outline

The rest of this thesis is organized as follows:

- Chapter 2 discusses related work about ACSA, text generation approach for ACSA, and AMR.
- Chapter 3 gives details about our proposed methods such as model structure, regularizers, and pre-training procedure.
- Chapter 4 reports several experiments to evaluate our proposed method. It first explains details about the utilized datasets and hyperparameters settings. Then, it presents the obtained results and ablation study.
- Chapter 5 concludes this research and describes future work.

Chapter 2

Related Work

This chapter introduces the related work of this study. Section 2.1 discusses different approaches to ACSA. As we will explain later, our model is based on Bidirectional and Auto-Regressive Transformers (BART) model. Therefore, we introduce Transformer that is the basic component of BART in Section 2.2, and review BART model in Section 2.3. Section 2.4 provides details of the text generation model for ACSA. Section 2.5 introduces AMR and its applications to multiple tasks in Natural Language Processing (NLP). Finally, Section 2.6 clarifies the characteristics of our method.

2.1 Aspect Category Sentiment Analysis

The conventional approaches perform ACSA as a classification task. Figure 2.2 shows a general framework of this approach. The classification model encodes an input by two embeddings. The contextual embeddings represent the context of the target aspect or the sentence itself, while the aspect category embeddings represent the information of the aspect. These two embeddings are fed into the attention layer to extract the aspect-related information. Finally, the aspect category embeddings and the new extracted information are passed to the classification layer that determines the polarity of the target aspect.

Ruder et al. represent relations between various sentences within a review using a hierarchical bidirectional LSTM model [40]. Wang et al. combine LSTM structure with the attention mechanism to extract related contextual information for given aspects [52]. Xue and Li explore CNN structure to compute granular features of different words before selectively outputting suitable features for each given aspect [56].

Several studies leverage aspect embedding to select and extract related

information as well as predict correct sentiment polarity for the given aspect [30, 54, 60]. Liang et al. design an aspect gate in Gated Recurrent Unit

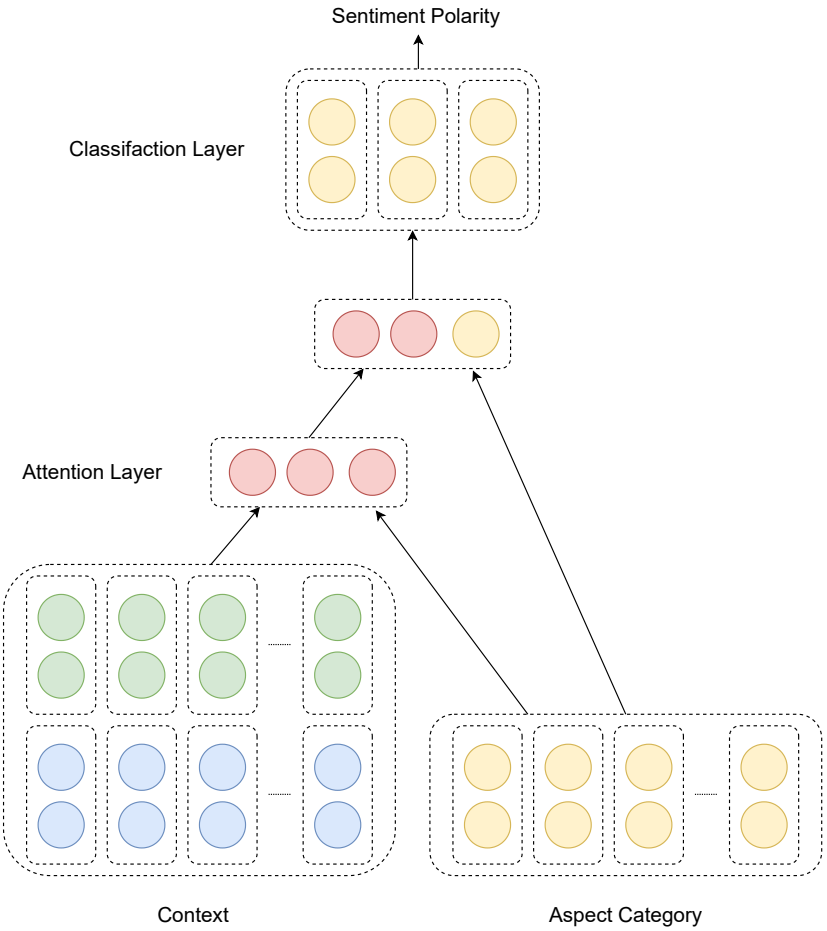


Figure 2.1: General framework of text classification approach for ACSA.

(GRU) for updating token embeddings with aspect-specific representation at each time step [30]. Xing et al. apply the same idea to LSTM-based model by creating three new aspect gates that control how much the aspect vector is fed into the input gate, forget gate, and output gate respectively [54]. Zhu et al. combine the LSTM model with the sentiment memory network which is based on the attention mechanism to extract the contextual information for an aspect category [60].

Li et al. accumulate sentiments of aspect-related words to calculate the polarity of the given aspect category [29]. To obtain better contextual representations, pre-trained language models like Bidirectional Encoder Representations from Transformers (BERT) [12] are utilized [45, 24]. Sun et al.

obtain the deep representations of sentences and aspects with BERT by simply concatenating the review sentence and aspect category as "[CLS] sentence [SEP] aspect [SEP]". Then, the obtained representations are fed to capsule networks for predicting sentiment polarity [45]. Jiang et al. perform ACSA by fine-tuning pre-trained BERT on sentence pair classification between the review sentence and an auxiliary sentence formed by filling the aspect and its category into a predefined question template or simply concatenating the aspect and corresponding category using symbol "-" as the separator. Liu et al. combine data augmentation using prompt-based text generation, syntactic information, and knowledge graph to enhance the performance of ACSA [31]. Shan et al. capture sequential contextual information and syntactic information of specific aspect category for ACSA [42].

To avoid error propagation, several studies explore joint models that perform ACSA and ACD simultaneously. Schmitt et al. propose two models with LSTM and CNN for outputting both aspect category and sentiment polarity together [41]. Wang et al. propose the aspect-level sentiment capsules model (AS-Capsules), which combines the capsule module and RNN structure, to exploit the correlation between aspect and sentiment [53]. Hu et al. apply orthogonal and sparseness constraints on attention weights [22]. Li et al. propose a joint model with a shared sentiment prediction layer to learn similar sentiment expressions of different aspect categories [28].

2.2 Transformer

Transformer [48] is a deep learning architecture that has been applied to various tasks of NLP as well as other research fields such as Computer Vision, and has achieved outstanding results. Figure 2.2 illustrates the architecture of Transformer model. The model is based on the encoder-decoder structure with N stacks of identical encoder layer and N stacks of identical decoder layer. Each decoder contains two main components: the multi-head self-attention module and the positional-wise fully connected feed-forward network. Both components are wrapped by a residual connection [21] followed by layer normalization [2]. The decoder shares the same components with the encoder except for adding another multi-head attention over the output of the encoder stack named Cross-Attention. In addition, the self-attention sub-layer in the decoder stack can only attend to prior positions. Similarly to other sequence transduction models, Transformer contains learned embeddings to transform the input tokens and output tokens to vectors of dimension d_{model}. Then, those vectors are injected with positional information from positional encoding which is availabel in both encoder and decoder

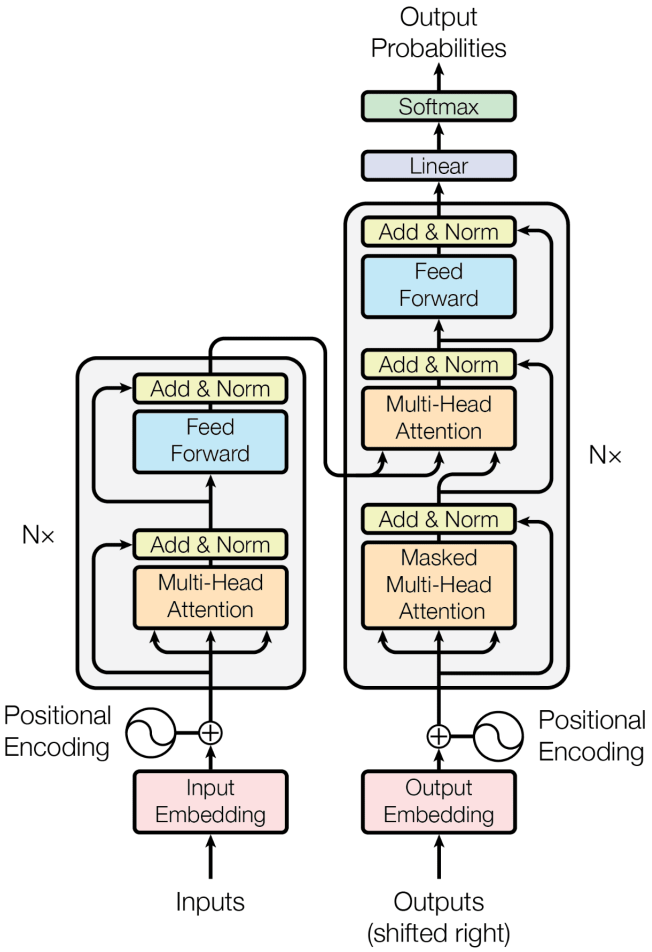


Figure 2.2: Architecture of Transformer [48].

sides.

In this architecture, the attention scores are calculated by the mechanism called "Scaled Dot-product Attention" as shown in left in Figure 2.3. Its inputs are queries and keys represented by vectors whose size is d_k, and values represented by vectors whose size is d_v. The scores are dot products of values and their weights which are computed by applying the softmax function on the output of dot products of the query with all keys scaled on $\sqrt{d_k}$. To speed up computation, sets of queries, keys, and values are packed into matrices Q, K, and V, respectively. The attention function is calculated

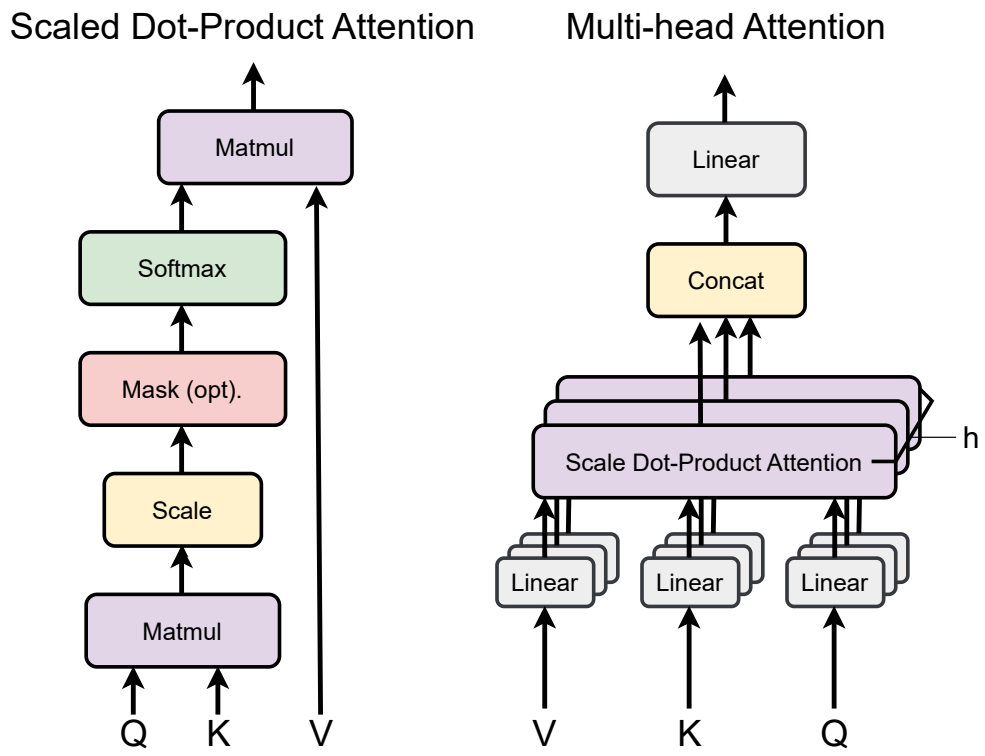


Figure 2.3: Scaled dot-product attention and Multi-head attention mechanism [48].

with those matrices as input. It outputs a single matrix as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

Multi-head Attention is the method to compute attention by using multiple Scale Dot-product Attention modules. This method is believed to let the model obtain the information from different representation subspaces at different positions while a single head cannot. Firstly, three input matrices K, V, and Q are linearly projected on multiple subspaces of d_k, d_k, and d_v dimensions, respectively. On each subspace, we apply the attention function in parallel before concatenating to the final values as shown in right in Figure 2.3. The score of multi-head attention is computed as follows.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, ..., head_h)W^O \tag{2.2}$$

$$head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.3}$$

The matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}, W_i^K \in \mathbb{R}^{d_{model} \times d_k}, W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ are the parameters to be trained.

Besides the multi-head attention module, each layer of both encoder and decoder consists of an identical feed forward network. This is composed of two linear transformations with a ReLU activation function as follows.

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2 \tag{2.4}$$

To incorporate positional information to the model, the Transformer contains "positional encodings" which are added to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings share the same dimension d_{model} with the input and output embeddings. This module is implemented with sine and cosine functions as follows:

$$\text{PE}(pos, 2i) = sin(pos/10000^{2i/d_{model}}) \tag{2.5}$$

$$\text{PE}(pos, 2i+1) = cos(pos/10000^{2i/d_{model}}) \tag{2.6}$$

In these equations, pos is the position and i is the dimension of the vector of the position encoding.

2.3 BART

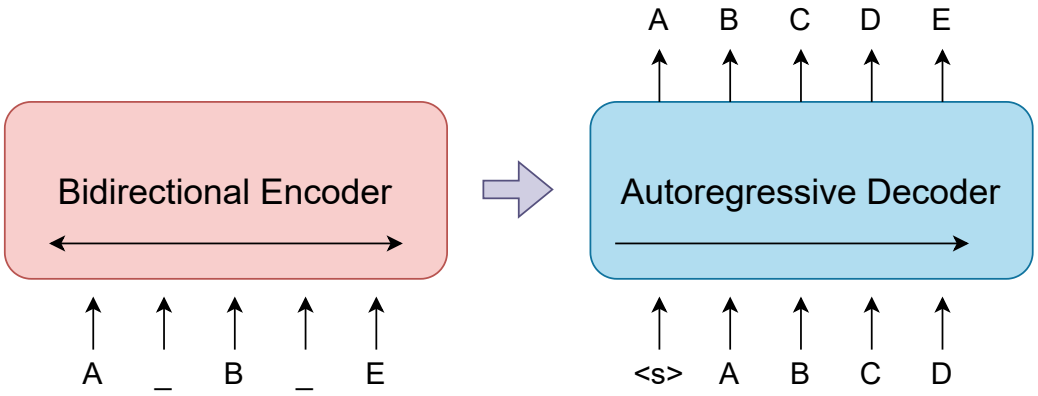


Figure 2.4: Architecture of BART and its pre-training [27].

Bidirectional and Auto-Regressive Transformers (BART) [27] is a a sequence-to-sequence model following the encoder-decoder framework. Both encoder and decoder are based on Transformer. As shown in Figure 2.4, the en- coder is the bidirectional Transformer, while the decoder is the left-to-right autoregressive Transformer. The framework of BART is different from other language models like BERT [12] which consists of the Transformer encoder only and Generative Pre-training (GPT) [39] which consists of the Transformer

decoder only. BART has shown its superiority in many sequence-to-sequence tasks such as machine translation, text summarization, and so on.

Figure 2.4 also shows how BART is pre-trained, that is, it is pre-trained as a text-denoising autoencoder that transforms a corrupted document into the original one. In other words, BART is pre-trained by giving a corrupted document as input and forcing the model to output the original sentence. The parameters of BART are estimated by optimizing the reconstruction loss between the decoder's output and the original document. BART applies five types of document corruption:

- **Token masking:** Random tokens are replaced by [MASK], which is the same as the pre-training of BERT [12]. In the example in Figure 2.5, the token B and D in the original sentences "A B C . D E ." are replaced with the mask "_".

- **Token deletion:** Random tokens are deleted from the input. The model could be trained to be aware of missing tokens' positions. In the example in Figure 2.5, the token B and D are deleted.

- **Text infilling:** In contrast to token masking, this method replaces each token span with a single [MASK] token. The number of text spans is randomly sampled, or more specifically, the span lengths are drawn from the Poisson distribution with $\lambda = 3$. 0-length spans mean the insertion of [MASK] tokens. This method can help the model learn the prediction of multiple tokens for one text span. In the example in Figure 2.5, the token B and C are replaced with a single mask. Another mask token is inserted between D and E by replacing a 0-length span.

- **Sentence permutation:** Sentences are randomly shuffled inside the document. In the example in Figure 2.5, the sentence "A B C ." and "C E ." are swapped.

- **Document rotation:** The document is rotated upon a randomly chosen token. This task enables the model to learn the start of the document. In the example in Figure 2.5, the document is rotated at the position of the token "C".

After the pre-training, BART model is fine-tuned for a downstream task. That is, the parameters of the pre-trained model are updated using a dataset of a certain task. BART can be applicable for many downstream tasks: sequence classification, token classification, sequence generation, and machine translation. For sequence classification, the same input is fed into the encoder and decoder, and the final hidden state of the final decoder token is fed

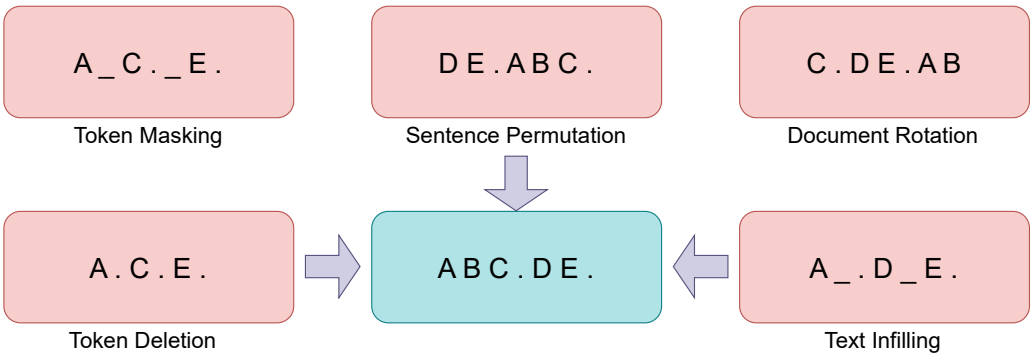


Figure 2.5: Transformations for noising the input [27].

into a new multi-class linear classifier. For token classification, they pass the entire document to the encoder and decoder, and use the final hidden state of the decoder as a representation for each word in the document. For sequence generation task, they can directly fine-tune BART as its decoder is autoregressive. For machine translation from any language to English, they replace the encoder's embedding with the newly initialized embedding. Then, they train the whole model to convert the foreign words into an input that BART can denoise to English. The results of previous studies have suggested that BART achieved competitive results on most of the tasks, especially in the text generation tasks. It might be because the framework of BART and its pre-training method are appropriate for text generation.

2.4 Text Generation for ACSA

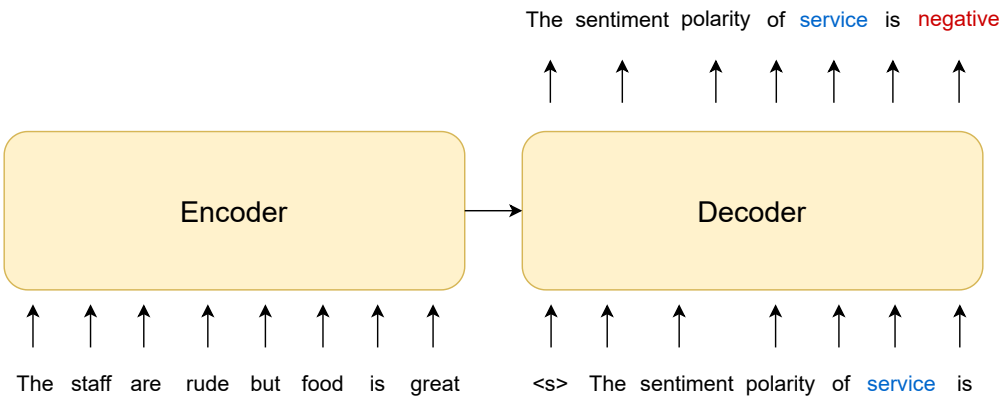


Figure 2.6: BART generation model [33].

Unlike the studies introduced in Section 2.1 that define ACSA as a classification task, Liu et al. propose a method to solve ACSA as the text generation task [33]. Their ACSA model is based on the prompt-based method and pre-trained autoregressive seq2seq language model BART [27]. This method is called "BART generation".

This section gives an overview of the BART generation method. The BART generation model aims to cast sentiment classification to text generation task. In Figure 2.6, the model is illustrated with the seq2seq framework whose input and output are the review sentence and the target sentence indicating the polarity of the aspect, respectively. It utilizes the parameters of the pre-trained BART model for initialization. The review sentence and the target sentence are denoted by $\mathbf{X} = \{x_1, x_2, ..., x_{|X|}\} = x_{1:|X|}$ (x_i is the i-th token) and $\mathbf{Y} = \{y_1, y_2, ..., y_{|Y|}\} = y_{1:|Y|}$ (y_i is also the i-th token), respectively.

A target sentence is generated by completing the blank spaces of a predefined template with an aspect category and a sentiment word. We denote the set of aspect categories by $\mathbf{A} = \{a_1, a_2, ..., a_{|A|}\}$ and the set of polarity types by $\mathbf{S} = \{s_1, s_2, ..., s_{|S|}\}$. The template is defined manually like "*The sentiment polarity of [ASPECT_CATEGORY] is [POLARITY_TYPE]*". For each review sentence $\mathbf{X}$ whose corresponding aspect category is a_p and sentiment polarity is s_t, we fill the slots in the template and get the target sentence "*The sentiment polarity of* $\langle a_p \rangle$ *is* $\langle s_t \rangle$" (E.g., "*The sentiment polarity of food is positive*").

For the training, given a pair of sentences $(\mathbf{X}, \mathbf{Y})$, the model is fed with an input sentence $\mathbf{X}$ through the encoder to get vector presentation h^{enc} of $\mathbf{X}$ as in Equation (2.7). In the decoder, the hidden vector at a time step j is calculated using h^{enc} and the hidden vectors of the previous time steps, as in Equation (2.8).

$$h^{enc} = \text{Encoder}(x_{1:|X|}) \tag{2.7}$$

$$h_j^{dec} = \text{Decoder}(h^{enc}, h_{1:j-1}^{dec}) \tag{2.8}$$

The conditional probability of the output token y_j is:

$$P(y_j | y_{1:j-1}, x_{1:|X|}) = \text{softmax}(\mathbf{W} h_j^{dec} + \mathbf{b}), \tag{2.9}$$

where $\mathbf{W} \in \mathbb{R}^{d_h \times |\mathcal{V}|}$ and $\mathbf{b} \in \mathbb{R}^{|\mathcal{V}|}$, $|\mathcal{V}|$ represents the vocabulary size. The loss function of this model is the following Cross Entropy:

$$\mathcal{L}_{ce} = -\sum_{j=1}^{|Y|} \log P(y_j | y_{1:j-1}, x_{1:|X|}). \tag{2.10}$$

For inference, we calculate the probabilities of all possible target sentences with different sentiment polarity classes using the trained model and choose the one with the highest probability. For an input sentence $\mathbf{X}$, aspect category a_p and sentiment polarity s_t, the probability of a target sentence $\mathbf{Y}_{a_p,s_t} = \{y_1, y_2, ..., y_m\}$ is calculated as follows:

$$f(\mathbf{Y}_{a_p,s_t}) = \sum_{j=1}^{m} \log P(y_j | y_{1:j-1}, \mathbf{X}) \tag{2.11}$$

For example, Figure 2.7 shows the three target sentences where the aspect "service" and one of the sentiment words (positive, negative, or neutral) are filled. Since the score of the second sentence is the highest, the polarity of the input review is guessed as negative.

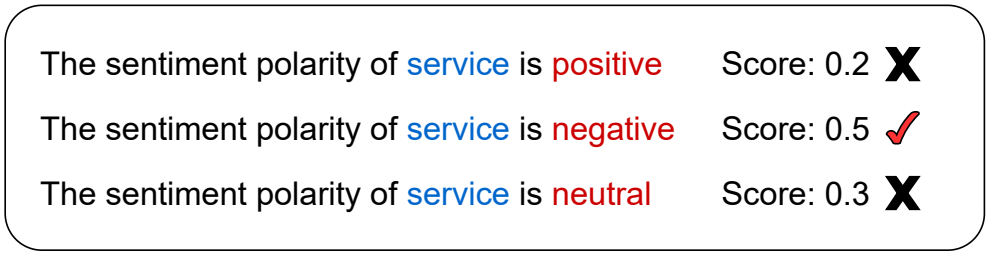
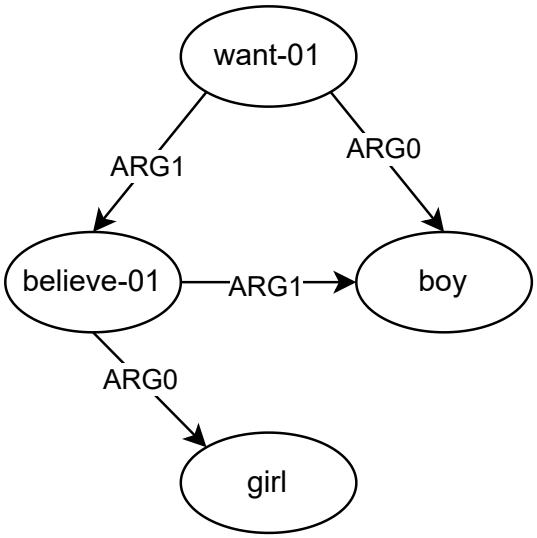


Figure 2.7: Inference step of BART generation model.

2.5 Abstract Meaning Representation

AMR [5] is the semantic formalism that represents the meaning of a sentence as a rooted, directed, and labeled graph. AMR aims at making abstract representation that is different from syntactic representation so that semantically similar sentences can be assigned to the same graph. For example, the following sentences could be represented by the same graph in Figure 2.8a.

- The boy desires the girl to believe him.
- The boy desires to be believed by the girl.
- The boy has a desire to be believed by the girl.
- The boy's desire is for the girl to believe him.
- The boy is desirous of the girl believing him.



(a) AMR as graph format.

$$w/want-01$$
$$:ARG0\ (b/boy)$$
$$:ARG1\ (b2/believe-01$$
$$:ARG0\ (g/girl)$$
$$:ARG1\ b))$$

(b) AMR as PENMAN format.

$$\exists w, b, b2, g :$$
$$instance(w, want-01) \wedge instance(b, boy) \wedge$$
$$instance(b2, believe-01) \wedge instance(g, girl)$$
$$ARG0(w, b) \wedge ARG(w, b2) \wedge$$
$$ARG(b2, g) \wedge ARG1(b2, b)$$

(c) AMR as logical format.

Figure 2.8: Different representation formats for AMR [5].

Besides the conventional graph format, AMR can also be converted to PENMAN format [7] as in Figure 2.8b or logical triples as in Figure 2.8c.

AMR graph consists of nodes that are labeled with concepts in PropBank framesets [26, 36] or special keywords, and edges that are labeled with relations. AMR covers roughly 100 relations divided into five general types: frame arguments, general semantic relations, relations for quantities, rela-

tions for date-entities, and relations for lists. There have been three public AMR corpora: LDC2014T12[1], LDC2017T10[2], and LDC2020T02[3]. The third one is the latest, which contains 59,255 pairs of sentence and corresponding AMR. To automatically evaluate the performance of AMR parsers, Cai and Knight introduce a metric called *smatch* [11]. This metric measures the overlap of parsed AMR and ground truth by considering each AMR as a conjunction of logical triples.

AMR parsing has been paid attention recently. Several methods have already been studied. They can be divided into three types: the graph-based method which identifies graph concepts first and then predicts relations among them [58, 10], the transition-based method which constructs the graph following the left-to-right direction of the input sentence and AMR alignments [14, 59], and seq2seq-based method which linearizes an AMR graph to fit with the seq2seq framework like Transformer [51, 8, 57]. Besides, following BART [27], another method is proposed to combine text-to-text and graph-to-graph conversion for pre-training an AMR-based model on text-denoising tasks [4].

AMR alignment is also an essential topic for fully utilizing AMR. This task concentrates on mapping nodes in AMR to corresponding words that are semantically similar. The conventional approaches focus on rule-based strategies [16, 15, 34, 46], and statistical strategies which utilize Expectation Maximization (EM)[38]. In addition, pre-trained word embeddings for tokens and nodes are utilized in low-resource settings [1]. Wang and Xue [50] leverage graph distance as a locality constraint on predicted alignments to align tokens and nodes. LEAMR [9] is a method that combines both predefined rules and EM to cover four types of aligned structures: subgraphs, relations, reentrancies, and duplicate subgraphs.

With the development of AMR parsers, AMR-to-text generation models, and large parallel datasets of the sentences and AMR graphs, AMR has been applied successfully to many downstream text generation tasks. For example, it has been integrated into a machine translation model as additional information for the source side [44, 35, 55]. In text summarization, several researchers transform AMR representations of sentences into an AMR graph of a summary and generate a text summary from the extracted subgraph [32, 13, 20, 23].

[1] `https://catalog.ldc.upenn.edu/LDC2014T12`
[2] `https://catalog.ldc.upenn.edu/LDC2017T10`
[3] `https://catalog.ldc.upenn.edu/LDC2020T02`

2.6 Characteristics of this study

This study follows the idea of the BART generation model [33] which casts ACSA from classification task to text generation task. BART generation, which is an efficient approach as the latest language models are pre-trained on text generation tasks, achieves state-of-the-art performance in ACSA. However, it undergoes difficulty in capturing relations of opinion words and target words within sentences that include multiple aspects. Therefore, we propose using AMR for modeling those relations. Furthermore, we also explore how semantic information from AMR contributes to improve the performance of ACSA. In addition, we introduce two regularizers for selectively extracting aspect-related information.

Chapter 3

Proposed Model

This chapter provides details about our proposed model. Section 3.1 formalizes the task definition. Section 3.2 describes the structure of our proposed model. Section 3.3 introduces the proposed regularizers. Section 3.4 explains the loss function of the whole model. Finally, Section 3.5 explains the pre-training procedure in our approach.

3.1 Task Definition

ACSA aims to predict the polarity of a given aspect category within an input sentence. Since our proposed method follows BART generation [33], we set up ACSA as a text generation task, which is explained in Section 2.4. More specifically, the input and output of our model are the review sentence and the target sentence that explicitly expresses the sentiment toward a given aspect category, respectively. In addition, we utilize a template shown in Figure 3.1, which is different from one used in the BART generation [33]. The *[SENTIMENT_WORD]* is filled by one of $\{excellent, awful, fine\}$ which corresponds to $\{positive, negative, neutral\}$, respectively. We believe that this template offers a more natural target sentence than the original one "*The sentiment polarity of [ASPECT_CATEGORY] is [POLARITY_TYPE]*" to the language model.

Quality of [ASPECT_CATEGORY] is [SENTIMENT_WORD].

Figure 3.1: Template to generate target sentence for ACSA.

3.2 Model Structure

Figure 3.2 shows the overall structure of our model which follows the text generation method [33]. The model is based on the encoder-decoder framework of Transformer [48], which is explained in Section 2.2. The parameters of the model are initialized by pre-trained language model BART [27], which is explained in 2.3. The model contains N encoder layers to encode the source sentence and N decoder layers to decode the target sentence, respectively. To encode semantic information from an AMR graph, we utilize an AMR Encoder module. In addition, we add a new AMR Cross-Attention layer and a subsequent layer normalization [2] into each decoder layer to incorporate the semantic information from AMR for generating the target sentence. In the following subsections, the details of the modules are described.

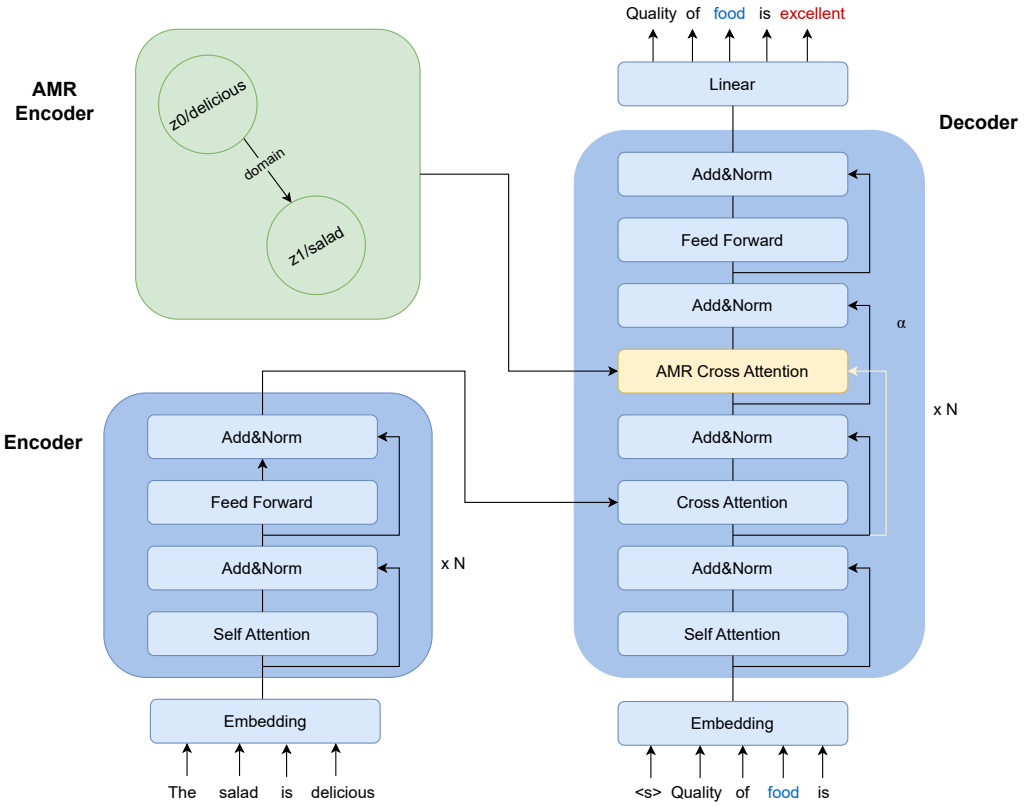


Figure 3.2: Our proposed model.

3.2.1 AMR Encoder

For a given input sequence $\mathbf{X} = \{x_1, x_2, ...x_{|X|}\}$, we construct a corresponding AMR graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ by the pre-trained AMR parser [4], where $\mathbf{V} = \{v_1, v_2, ..., v_{|V|}\}$ is the set of nodes and $\mathbf{E} \in \mathbb{R}^{|V|\times|V|}$ is the adjacency matrix presenting the relations between the nodes. Although AMR is a directed graph, we treat the AMR graph as an undirected graph, which means $e_{ij} = e_{ji} = 1$ if the two nodes v_i and v_j are connected, otherwise 0. This modification aims at transferring the information in both directions between nodes of an opinion word and an aspect word, when the embedding of the nodes are trained.

The AMR encoder adopts Graph Attention Networks (GAT) [49]. Given a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ and node $v_i \in \mathbf{V}$, we can obtain h'_i, the hidden state of node v_i, as follows:

$$h'_i = \sigma\big(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\mathbf{W}h_j\big) \tag{3.1}$$

$$\alpha_{ij} = \frac{\exp\big(\sigma\big(\mathbf{a}^T[\mathbf{W}h_i \| \mathbf{W}h_j]\big)\big)}{\sum_{k\in\mathcal{N}_i}\exp\big(\sigma\big(\mathbf{a}^T[\mathbf{W}h_i \| \mathbf{W}h_k]\big)\big)}, \tag{3.2}$$

where $\mathbf{a}^T$ and $\mathbf{W}$ are trainable parameters, σ is the LeakyRELU function, $\|$ denotes the concatenation of two vectors, $\mathcal{N}_i$ is the set of neighbor nodes of v_i in $\mathbf{G}$, and h_i is the initial representation of v_i. Note that a node (word) consists of several subwords in general. Using the embedding of the AMR parser [4], h_i is defined as the average of the subword vectors.

In addition, applying the multi-head attention mechanism from the Transformer architecture [48], the formula to update the representation of node v_i is changed from Equation (3.1) to:

$$h'_i = \Big\Vert_{k=1}^{K} \sigma\big(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k h_j\big), \tag{3.3}$$

where K is the number of attention heads, α_{ij}^k are the attention coefficients of the k-th head, $\mathbf{W}^k$ is the weight matrix at the k-th head, and $\|$ stands for the concatenation of multiple vectors.

3.2.2 Decoder

After obtaining the graph information, we feed it into each decoder layer by adding a new Cross-Attention module for AMR referred to as "AMR Cross-Attention" in Figure 3.2. We write h' for the representations of the AMR

nodes obtained from GAT, x is the vector representation of the input sentence and y^l is the output of l-th decoder layer. The output of the $(l+1)$-th decoder layer, y^{l+1}, is obtained as follows:

$$\dot{y}^l = \text{LN}(y^l + \text{SelfAttn}(y^l)) \tag{3.4}$$

$$\ddot{y}^l = \text{LN}(\dot{y}^l + \text{CrossAttn}(\dot{y}^l, x)) \tag{3.5}$$

$$\dddot{y}^l = \text{LN}(\ddot{y}^l + \text{CrossAttn}(\ddot{y}^l, h')) \tag{3.6}$$

$$y^{l+1} = \text{LN}(\dddot{y}^l + \text{FFN}(\dddot{y}^l)), \tag{3.7}$$

where LN is the layer normalization function, SelfAttn is the self-attention module, CrossAttn is the cross-attention module, and FFN is the feed-forward neural network. Note that the first cross-attention module in Equation (3.5) encodes the attention to the encoder, while the second one in Equation (3.6) encodes the attention to the AMR graph.

Training a deep model like Transformer is really hard and even harder with one more cross-attention module. To overcome this difficulty, we employ ReZero [3] as the AMR Cross-Attention module instead of the normal residual module. This method is implemented as follows:

$$\tilde{y}^l = \ddot{y}^l + \alpha\mathcal{F}(\ddot{y}^l), \tag{3.8}$$

where $\mathcal{F}$ denotes non-trivial functions and α is a trainable parameter that helps moderate the updating of the AMR Cross-Attention.

3.3 AMR Cross-Attention Regularizer

To incorporate the semantic information from the AMR graph more effectively, we propose two regularizers over the attention scores of the AMR Cross-Attention module.

3.3.1 Identical Regularizer

Intuitively, a word in a sentence and its aligned node in the AMR graph should receive the same attention as they are supposed to represent similar semantic information. This motivates us to introduce an extra regularizer (or loss function) that evaluates how a word and its corresponding node have equal attention. Figure 3.3 illustrates an example of ideal Cross-Attention and AMR Cross-Attention for the generation of the word "excellent", whose attention weights for pairs of aligned nodes and word are almost equal.

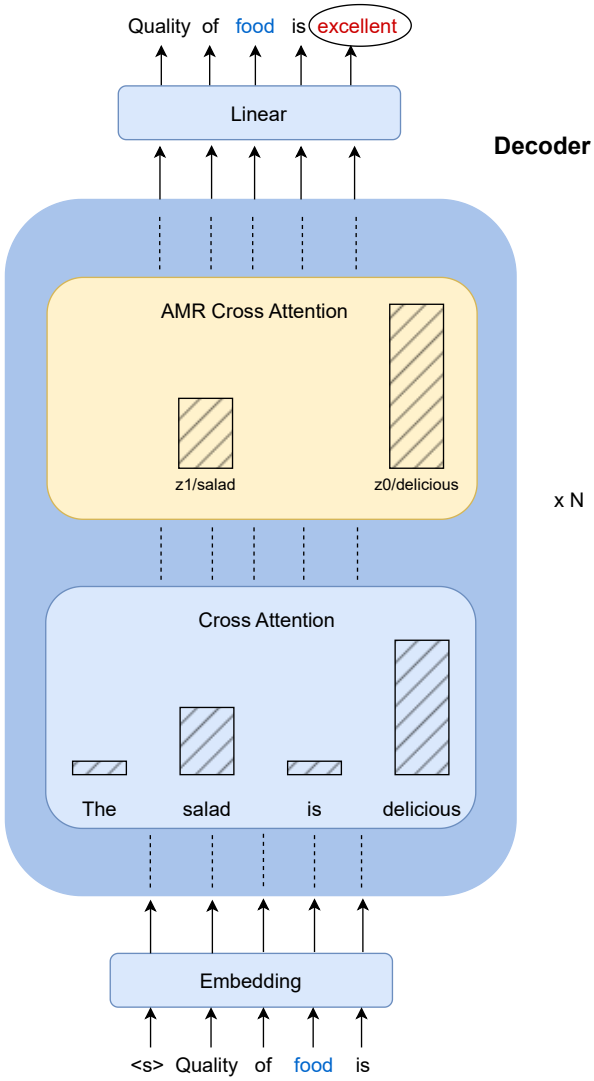


Figure 3.3: An example of ideal Cross-Attention and AMR Cross-Attention derived by identical regularizer.

Two transformation matrices for the Cross-Attention matrix over each of the source (input) sentences and the AMR graphs are defined by $\text{align}_{src} \in R^{|X|\times|P|}$ and $\text{align}_{amr} \in R^{|V|\times|P|}$, respectively, where $|P|$ is the number of pairs of aligned words and nodes. A cell $align_{src}[i,k]$ shows whether the i-th subword belongs to a word within the k-th pair of aligned word and node. A cell $align_{amr}[j,k]$ shows whether the j-th AMR node belongs to the k-th pair of aligned word and node. The precise definition of the value for each

cell is presented as follows:

$$align_{src}[i,k] = \begin{cases} \frac{1}{|\mathcal{T}_i|} & \text{if subword } x_i \text{ belongs to} \\ & \text{the word in } k\text{-th aligned pair} \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

$$align_{amr}[j,k] = \begin{cases} 1 & \text{if node } v_j \text{ and node in } k\text{-th} \\ & \text{aligned pair are the same} \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

Here, $\mathcal{T}_i$ denotes a set of subwords in the aligned word. With these matrices and two given Cross-Attention matrices $A_{src} \in \mathbb{R}^{|Y|\times|X|}$, $A_{i\_amr} \in \mathbb{R}^{|Y|\times|V|}$ over the review sentence and the AMR graph, respectively, the identical regularizer is formulated as follows:

$$\mathcal{L}_{ir} = \sum_{i=1}^{L} \frac{1}{L} \| A^i_{src} \cdot align_{src} - A^i_{i\_amr} \cdot align_{amr} \|_F, \tag{3.11}$$

where $\|\|_F$ denotes the Frobenius norm and L is the number of the decoder layers. The matrix A_{src} is obtained from an oracle fine-tuned text generation model. Also, the matrix A_{i_amr} is obtained by fetching the same input with the regular Cross-Attention layer over the source sentence, which is indicated by the yellow line in Figure 3.2.

3.3.2 Entropy Regularizer

We expect that our model concentrates on a few important nodes. This means that the Cross-Attention distribution of the tokens over the AMR nodes is supposed to be skewed. Therefore, we try to minimize the information entropy [43] of the attention scores of the tokens over the AMR nodes. Figure 3.4 shows the ideal effect of the entropy regularizer. The cross bars and the orange bars show the attention distribution before and after being applied entropy regularizer. The attention weight over the node "*z0/delicious*" of AMR Cross-Attention is increased since it may be the most important node to generate the word "*excellent*", indicating that the polarity of the aspect *food* is positive.

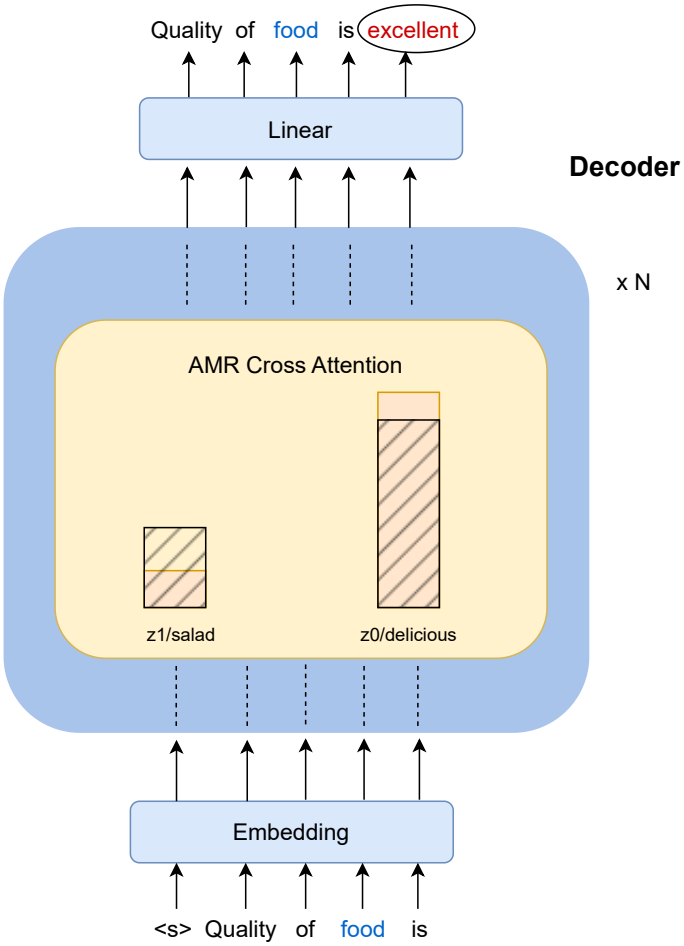


Figure 3.4: An example of ideal effect of the entropy regularizer on the AMR Cross-Attention distribution.

We first calculate the mean of the Cross-Attention score of the token i at the node j over $\mathcal{H}$ attention heads as follows:

$$\tilde{a}_{ij} = \frac{1}{\mathcal{H}} \sum_{h=1}^{\mathcal{H}} a_{ijh} \tag{3.12}$$

Then, the entropy of the l-th decoder layer is calculated over $|V|$ nodes and $|Y|$ output tokens:

$$H^l = -\frac{1}{|Y|} \sum_{i}^{|Y|} \sum_{j}^{|V|} \tilde{a}_{ij} \log \tilde{a}_{ij} \tag{3.13}$$

The entropy regularizer is defined as the mean entropy of the L decoder

layers:

$$\mathcal{L}_{er} = \frac{1}{L}\sum_{l=1}^{L} H^l \tag{3.14}$$

3.4 Loss Function

For training the proposed model, the loss function is the sum of the normal cross entropy loss and the aforementioned two regularizers:

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda_1 \mathcal{L}_{ir} + \lambda_2 \mathcal{L}_{er}, \tag{3.15}$$

where λ_1 is the scaling factors of the identical regularizer and λ_2 is that of the entropy regularizer.

3.5 Pre-training Procedure

It is hard to fine-tune our model, which consists of randomly initialized modules like the AMR graph encoder and the AMR Cross-Attention layers together with the pre-trained BART. Bataa and We showed the positive effect of additional pre-training of a language model with in-domain data before fine-tuning for downstream tasks [6], while Gheini et al. proved the effectiveness of fine-tuning Cross-Attention layers [18]. Following their ideas, after initializing the whole model with pre-trained BART model, we train it again with text-denoising tasks using in-domain texts, i.e., review sentences. Similar to the pre-training of BART, we artificially make a noisy sentence and train the model so that it generates the original sentence when the noisy sentence is given as the input. In this study, we add noise into the input sentences using the following three methods:

- **Token Masking**: Random tokens are sampled and replaced by $[MASK]$ token.
- **Text Infilling**: Random text spans are replaced by $[MASK]$ using a Poisson distribution.
- **Text Deletion**: Random text spans are deleted. Unlike Word Deletion used for pre-training of BART where a randomly chosen single token is deleted, the whole of text spans is deleted in this method.

Algorithm 1 shows the pseudocode of the text corruption algorithm that adds noise by the above three methods.

Algorithm 1 Text corruption algorithm

Input: review sentence $\mathbf{X} = \{x_1, x_2, ...x_n\}$
Output: corrupted review sentence $\mathbf{X'} = \{x'_1, x'_2, ...x'_m\}$
$p_{token} \leftarrow 0.15$ - probability of replacing one token by [MASK]
$p_{word} \leftarrow 0.3$ - probability of replacing text spans by [MASK]
$\lambda_{Possion} \leftarrow 3$ - value for λ parameter in Possion distribution

$p \leftarrow gen\_random[0, 1]$
if $p < \frac{1}{3}$ **then**
    $\mathbf{X'} \leftarrow mask\_tokens(\mathbf{X}, p_{token})$
else if $p < \frac{2}{3}$ **then**
    $\mathbf{X'} \leftarrow mask\_text\_spans(\mathbf{X}, p_{word}, \lambda_{Possion})$ - Sampling text spans with span lengths drawn from a Poisson distribution and masking them.
else
    $\mathbf{X'} \leftarrow mask\_text\_spans(X, p_{word}, \lambda_{Possion})$
    $\mathbf{X'} \leftarrow delete\_mask\_tokens(\mathbf{X'})$
end if

The entropy regularizer is also taken into account in this pre-training step. That is, the loss function of the pre-training is defined as:

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda_3 \mathcal{L}_{er}, \tag{3.16}$$

where λ_3 is the scaling factor for the entropy regularizer.

Chapter 4

Evaluation

This chapter gives details about our experiments to evaluate our proposed method. Section 4.1 describes the datasets used in the experiments. Section 4.2 provides information about our settings of the models. Section 4.3 presents a list of baselines which are compared with our model. Section 4.4 demonstrates the main results of the experiments. Section 4.5 investigates the performance of our model with different settings. Finally, Section 4.6 provides a qualitative analysis of our obtained results.

4.1 Datasets

The following three datasets are used for performing ACSA in our experiments.

- **Rest14**: This dataset consists of reviews in the restaurant domain, which is included in the Semeval-2014 task [37]. It consists of review sentences about restaurants. For each sentence, aspect categories and their polarity labels ("positive", "neutral", "negative", or "conflict") are annotated. In this experiment, samples labeled with "conflict" are removed, so the remaining samples have the labels "positive", "negative" or "neutral". In addition, we follow the splitting of the development set suggested by Tay et al. [47] for the sake of fair comparison.

- **Rest14-hard**: Xue and Li construct this dataset for better evaluating a model on sentences with multiple aspects [56]. Since its size is small, it is only used as the test data. The training set and development set are the same as those of Rest14.

- **MAMS-ACSA**: For the same purpose as the Rest14-hard, Jiang et al. propose a larger dataset for ACSA in which each sentence contains at

least two different aspects [24]. This dataset also focuses on restaurant reviews which are excerpted from Citysearch New York dataset [17].

The number of samples for each polarity class in those datasets is shown in Table 4.1.

Table 4.1: Statistics of datasets.

Dataset		**#Pos**	**#Neg**	**#Neu**
Rest14	Train	1855	733	430
	Dev	324	106	70
	Test	657	222	94
Rest14-hard	Test	21	20	12
MAMS-ACSA	Train	1929	2084	3077
	Dev	241	259	388
	Test	245	363	393

Table 4.2 shows examples of review sentences in Rest14, Rest14-hard, and MAMS, which includes each review sentence and its given aspect categories together with corresponding polarity. P, N, and O represent positive, negative, and neutral respectively.

As described in Section 3.5, our text generation model is pre-trained using in-domain data. In this experiment, we only use review sentences in the training set of each dataset as training data for the pre-training.

Table 4.2: Examples of review sentences in three datasets.

Dataset	Sentence	Aspect Category	Label
Rest14	The bread is top notch as well.	{food}	(P)
	The design is very intimate and romantic.	{ambience}	(P)
	The staff is arrogant, the prices are way high for Brooklyn.	{service, price}	(N, N)
	We'd walked by it dozens of times and finally settled on a Monday night.	{miscellaneous}	(O)
	Anyway, the owner was fake.	{service}	(N)
Rest14-hard	Although the restaurant itself is nice, I prefer not to go for the food.	{ambience, food}	(P, N)
	A mix of students and area residents crowd into this narrow, barely there space for its quick, tasty treats at dirt-cheap prices.	{ambience, food, price}	(N, P, P)
	I was on jury duty, rode my bike up Centre Street on my lunch break and came across this great little place with awesome chicken tacos and Hibiscus lemonade.	{food, ambiance, miscellaneous}	(P, P, O)
	Give it a try, menu is typical French but varied.	{food, miscellaneous}	(O, P)
	How can they survive serving mediocre food at exorbitant prices?!	{food, price}	{O, N}
MAMS	Thanks to waiter I learned so much about wine too.	{staff, food}	(P, O)
	Nothing on the menu jumped out at me, but when we tasted the chicken and the pork tenderloin.	{menu, food}	(N, O)
	Small dishes, a bit pricier than you'd pay in Miami or LA, but the atmosphere is on the sexy side (however pared down) and its cozy.	{food, miscellaneous}	(N, O)
	Service was a tad spotty, but the food was VERY good and the noise never detracted from the dining experience.	{service, food, miscellaneous}	(N, P, N)
	We let the very kind hostess know we were there and had some drinks at the bar.	{staff, food}	(P, O)

4.2 Implementation Details

4.2.1 AMR Processing

For AMR parsing and embeddings extraction, we use the pre-trained model of AMRBART[1][4] which is inspired by text denoising task of BART [27] to build a graph pre-trained model. This model follows Transformer [48] with encoder-decoder framework. Two pre-trained AMRBART models with different settings, *base* and *large*, are available. We utilize the checkpoints of pre-trained AMRBART with both settings as follows.

- **checkpoint of AMRBART-base[2]:** is used to extract the weights of the embedding layer for initial representations of nodes, i.e., h_i in Equation (3.1) and (3.2) in Subsection 3.2.1 since our model is based on BART-base model.
- **checkpoint of AMRBART-large[3]:** is used for AMR parsing since it achieves better performance than AMRBART-base on LDC2017T10 which is the training dataset for both models.

Figure 4.1 illustrates an example of the input and output of AMRBART. Our input in JSON format consists of two parts: one for the source sentence and one for the target sentence. Since we only use the pre-trained model for AMR parsing, we only fill an input review sentence into the source side with our review sentence. On the other hand, the AMR output is obtained in two formats: (1) text sequence where the nested structure is represented by parentheses or (2) PENMAN format. For simplicity, we only use the PENMAN format as our main format. Then, we utilize the package Penman[4][19], which is designed for reading and writing AMR graphs in PENMAN format, to calculate the adjacency matrix of the graph. The information of adjacency matrix and subtoken id for each node based on AMRTBART tokenizer in the graph is saved as a Numpy[5] file.

[1] `https://github.com/goodbai-nlp/AMRBART`

[2] `https://huggingface.co/xfbai/AMRBART-base-finetuned-AMR2.0-AMRParsing`

[3] `https://huggingface.co/xfbai/AMRBART-large-finetuned-AMR2.0-AMRParsing-v2`

[4] `https://github.com/goodmami/penman`

[5] `https://numpy.org`

{"src": "The atmosphere was wonderful, however the service and food were not.", "tgt": ""}

(a) Input format in JSON.

<s> (<pointer:0> contrast-01 :ARG1 (<pointer:1> wonderful-03 :ARG1 (<pointer:2> atmosphere)) :ARG2 (<pointer:3> wonderful-03 :polarity - :ARG1 (<pointer:4> and :op1 (<pointer:5> service) :op2 (<pointer:6> food)))) </AMR>

(b) AMR output in text sequence.

```
# ::id 22
# ::annotator bart-amr
# ::date 2022-05-15 15:12:40.638438
# ::snt The atmosphere was wonderful, however the service and food
were not.
(z0 / contrast-01
   :ARG1 (z1 / wonderful-03
         :ARG1 (z2 / atmosphere))
   :ARG2 (z3 / wonderful-03
         :polarity -
         :ARG1 (z4 / and
               :op1 (z5 / service)
               :op2 (z6 / food))))
```

(c) AMR output in PENMAN format.

Figure 4.1: Example of input and output of AMRBART [4].

In addition, LEAMR[6][9] is adapted to align the words in the input sentence and the nodes in the AMR graph. The LEAMR model accepts the AMR in PENMAN format as an input and provides the alignments in JSON format as an output. Figure 4.2 shows an example of the output of LEAMR which contains AMR graph information of nodes and edges at the top and alignments between nodes and words at the bottom. For example, the words "*wonderful*" and "*however*" are corresponded to the nodes "*wonderful-03*"

[6] `https://github.com/ablodge/leamr`

and "*contrast-01*", respectively.

```
# ::id 22
# ::tok The atmosphere was wonderful, however the service and food were not.
# ::annotator bart-amr
# ::date 2022-05-15 15:12:40.638438
# ::snt The atmosphere was wonderful, however the service and food were not.
# ::node   1   contrast-01
# ::node   1.1   wonderful-03
# ::node   1.1.1   atmosphere
# ::node   1.2   wonderful-03
# ::node   1.2.2   and
# ::node   1.2.2.1   service
# ::node   1.2.2.2   food
# ::node   1.2.1   -
# ::root   1   contrast-01
# ::edge   wonderful-03   polarity   -   1.2   1.2.1
# ::edge   contrast-01   ARG1   wonderful-03   1   1.1
# ::edge   wonderful-03   ARG1   atmosphere   1.1   1.1.1
# ::edge   contrast-01   ARG2   wonderful-03   1   1.2
# ::edge   wonderful-03   ARG1   and   1.2   1.2.2
# ::edge   and   op1   service   1.2.2   1.2.2.1
# ::edge   and   op2   food   1.2.2   1.2.2.2

<AMR_Alignment: subgraph>: tokens [1] nodes ['1.1.1'] edges [] (subgraph : atmosphere => atmosphere)
<AMR_Alignment: subgraph>: tokens [2] nodes [] edges [] (subgraph : was => )
<AMR_Alignment: subgraph>: tokens [3] nodes ['1.1'] edges [] (subgraph : wonderful, => wonderful-03)
<AMR_Alignment: subgraph>: tokens [4] nodes ['1'] edges [] (subgraph : however => contrast-01)
<AMR_Alignment: subgraph>: tokens [5] nodes [] edges [] (subgraph : the => )
<AMR_Alignment: subgraph>: tokens [6] nodes ['1.2.2.1'] edges [] (subgraph : service => service)
<AMR_Alignment: subgraph>: tokens [7] nodes ['1.2.2'] edges [] (subgraph : and => and)
<AMR_Alignment: subgraph>: tokens [8] nodes ['1.2.2.2'] edges [] (subgraph : food => food)
<AMR_Alignment: subgraph>: tokens [9] nodes [] edges [] (subgraph : were => )
<AMR_Alignment: subgraph>: tokens [10] nodes ['1.2', '1.2.1'] edges [('1.2', ':polarity', '1.2.1')] (subgraph : not.
=> wonderful-03, -, ('wonderful-03', ':polarity', '-'))
```

Figure 4.2: Example of output of LEAMR [9] output.

4.2.2 Model Settings

In the pre-training step, we initialize the parameters of BART using the checkpoint of BART base[7]. Unlike the parameters of BART, those of the AMR graph encoder and the AMR Cross-Attention modules are newly initialized with a uniform distribution. After pre-training, the last checkpoint is used for fine-tuning the ACSA model. The Adam optimizer [25] is used for optimizing the model. The original parameters of BART's encoder and decoder are trained with a learning rate of 2e-5 while the learning rate is set to 3e-5 for the parameters in the AMR graph encoder and the AMR Cross-Attention modules. We set the number of the attention heads of the AMR

[7] `https://huggingface.co/facebook/bart-base`

encoder to 6, the number of AMR Cross-Attention heads to 6, the batch size to 16, and the dropout value to 0.1. The initial value for the ReZero weight α is 1. The regularization coefficients λ_1 and λ_2 are set to (0.075, 0.1), (0.075, 0.1), and (0.025, 0.0075) for Rest14, Rest14-hard, and MAMS datasets respectively, while λ_3 is always set to 5e-3. All hyperparameters are tuned based on the accuracy of the development set. As for the optimization of coefficients λ_i , they are changed as $\lambda_1 \in \{0.1, 0.075, 0.05, 0.025\}$, $\lambda_2 \in \{0.25, 0.1, 0.075, 0.05\}$, and $\lambda_3 \in \{0.01, 0.0075, 0.005, 0.0025\}$.

4.3 Baselines

We compare our method with multiple baselines:

- **GCAE** [56]: employs CNN model with the gating mechanism to selectively output the sentiment polarity related to a given aspect.
- **AS-Capsules** [53]: exploits the correlation between aspects and corresponding sentiments through a capsule-based model.
- **CapsNet** [24]: is the capsule network-based model to learn the relations between the aspects and the contexts.
- **CapsNet-BERT** [24]: is the CapsNet model based on the pre-trained BERT.
- **BERT-pair-QA-B** [45]: performs ACSA as the sentence pair classification task by fine-tuning of the pre-trained BERT.
- **AC-MIMLLN** [29]: predicts the polarity of a given aspect by combining the sentiments of the words indicating the aspect.
- **AC-MIMLLN-BERT** [29]: is the AC-MIMLLN model based on the pre-trained BERT.
- **BART generation** [33]: performs ACSA by a text generation model with the pre-trained BART. It is almost equivalent to our model without AMR.
- **BART generation with pre-training**: is the BART generation model combined with our pre-training method except for applying entropy regularization.

Table 4.3: Accuracy (%) of ACSA models. † refers to citation from [24].

Model	Rest14	Rest14-hard	MAMS
GCAE [56]	81.3(±0.883)	54.7(±4.92)	72.1†
AS-Capsules [53]	82.2(±0.414)	60.8(±2.77)	75.1(±0.473)
CapsNet [24]	81.2(±0.631)	54.0(±0.924)	74.0†
AC-MIMLLN [29]	81.6(±0.715)	65.3(±2.26)	76.4(±0.704)
BERT-pair-QA-B [45]	87.5(±1.18)	69.4(±4.37)	79.1(±0.973)
CapsNet-BERT [24]	86.6(±0.943)	51.3(±1.41)	79.5†
AC-MIMLLN-BERT [29]	89.3(±0.720)	74.7(±3.29)	81.2(±0.606)
BART generation [33]	90.5(±0.315)	77.4(±2.16)	83.1(±0.478)
BART generation with pre-training	90.6(±0.517)	75.5(±3.77)	83.6(±0.847)
Our model	**91.2(±0.258)**	**78.1(±2.53)**	**84.6(±0.453)**

4.4 Experimental Results

The results of the experiments are presented in Table 4.3. The models were trained and evaluated five times with different initializations of the parameters. The table shows the average and standard deviation of the accuracy of five trials using the format "mean (±std)". First, our model outperforms all baselines on the three datasets, which indicates the necessity of incorporating semantic information into the text generation model for ACSA. Second, compared with the models that learn relations between the aspect and the context like CapsNet, AC-MIMLLN, BERT-pair-QA-B and BART generation, the dominance of our model proves that exploiting the AMR graph to learn relations between words is a better way to capture contextual information. The fact that our model also outperforms BART generation with the pre-training further supports that the improvement on our model is not only from the in-domain data but also from the AMR. Third, the competitive results over the Rest14-hard and MAMS datasets show the effectiveness of the identical and entropy regularizers in enabling the model to concentrate on the correct aspect-related nodes, which is essential for the identification of the polarity over multiple aspects.

4.5 Ablation Study

To further investigate the effects of the different modules in our model, we conducted ablation studies. The results are presented in Table 4.4. First, it is found that the removal of the identical regularizer downgrades the performance, which indicates the importance of precisely capturing the semantic information. Second, we also notice that the models without the entropy regularizer perform poorly with a reduction of 0.8, 1.1, and 0.4 percentage

Table 4.4: Ablation study.

Model	Rest14	Rest14-hard	MAMS
Our model	**91.2(±0.258)**	**78.1(±2.53)**	**84.6(±0.453)**
w/o identical regularizer	91.0(±0.424)	77.4(±1.89)	84.0(±0.320)
w/o entropy regularizer	90.4(±0.162)	77.0(±1.68)	84.2(±1.10)
w/o entropy and identical regularizer	90.3(±0.426)	74.3(±1.69)	83.8(±0.638)
w/o pre-training	89.8(±0.217)	70.6 (±1.03)	83.1(±0.618)

points in the accuracy on Rest14, Rest14-hard, and MAMS, respectively. This shows that the entropy regularizer is essential to prevent models from attending to unnecessary AMR nodes. In addition, removing both regularizers degrades the performance more than removing each of the regularizers, which confirms the essential roles of these regularizers in performing ACSA. Third, removing the pre-training procedure hurts the performance badly, which leads to decreases by 1.4, 7.5, and 1.5 percentage points on the three datasets respectively. This indicates the big gap between the newly initialized modules and the pre-trained model and the necessity of the pre-training step for overcoming this problem. The ablation studies show that each component positively affects the entire model. The contribution of the pre-training step is the greatest, while those of the identical and entropy regularizers are comparable to each other.

4.6 Analysis

4.6.1 Case Study

To further examine how the semantic information of AMR and two regularizers work well in ACSA, a few examples are shown as a case study. Table 4.5 compares our model with the state-of-the-art method "BART generation". The symbols P, N, and O represent the positive, negative, and neutral classes respectively. The first example, "*I never had an orange donut before so I gave it a shot*", has no explicit sentiment expression. With the help of semantic information and two regularizers, our model can correctly predict the true label while BART generation cannot. The second and third examples contain multiple aspects, which can affect each other's predictions. In the second example, the BART generation model may capture the positive sentiment toward the aspect word "*atmosphere*" for anticipating the sentiment of the different aspect "*service*', which leads to outputting the wrong label. Another incorrect prediction by this baseline is shown in the third example, where the polarities of "*food*" and "*staff*" are mistakenly swapped.

In contrast, our model pays attention to only the aspect-related AMR nodes, resulting in the correct predictions in both examples. However, our model also faces difficulty in some cases. In the last example, it wrongly predicts the sentiment polarity for "*miscellaneous*" because it is really hard to capture aspect-related AMR nodes for a coarse-grained aspect class like "*miscellaneous*".

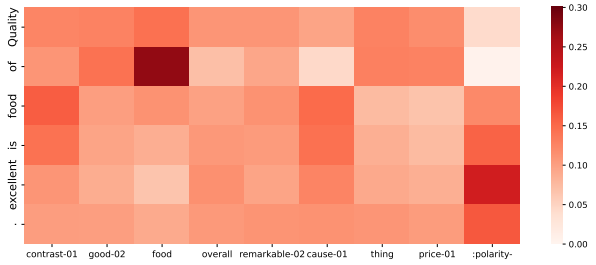
Table 4.5: Case studies of our model compared with state-of-the-art method.

Sentence	Aspect Category	BART generation	Our model	Label
I never had an orange donut before so I gave it a shot	{food}	(P)	(O)	(O)
The atmosphere was wonderful, however the service and food were not.	{ambiance, service, food}	(P, P, N)	(P, N, N)	(P, N, N)
There are several specials that change daily, which the servers recite from memory.	{food, staff}	(O, P)	(P, O)	(P, O)
The place was busy and had a bohemian feel.	{place, miscellaneous}	(P, P)	(N, P)	(N, O)

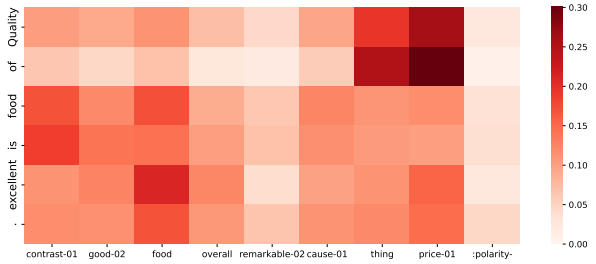
4.6.2 Attention Visualization

To study the effectiveness of the two regularizers in guiding the AMR Cross-Attention collocation, we illustrate the Cross-Attention matrix produced by our full model and the model without two regularizers in Figure 4.3. The review sentence is "*The food was good overall, but unremarkable given the price.*" For the aspect category "*food*", both models correctly predict the sentiment polarity "*positive*". However, for the aspect category "*price*", the model without regularizers incorrectly predicts the label "*neutral*" while our full model successfully predicts the correct label "*negative*". From Figure 4.3, we can see that the model without two regularizers has dense attention matrices that introduce noise to the prediction of the polarity. In contrast, the attention matrices of our full model are sparse. For example, as for the food category, the words "*food*" and "*excellent*" in the target sentence pay much attention or more attention than the model without the regularizers to the nodes "*food*" and "*good-02*" in the AMR graph. Similarly, as for the price category, "*price*" in the target sentence pays a great deal of attention to the node "*price-01*" in the AMR graph, while "*awful*" pays less attention to "*remarkable-02*" than the model without the regularizers. Those cases

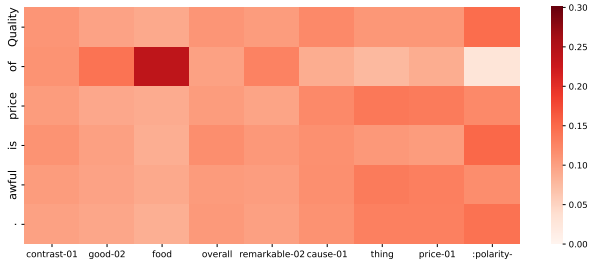
indicate that our attention regularizers help attention layers work well even when a review sentence contains multiple aspects.



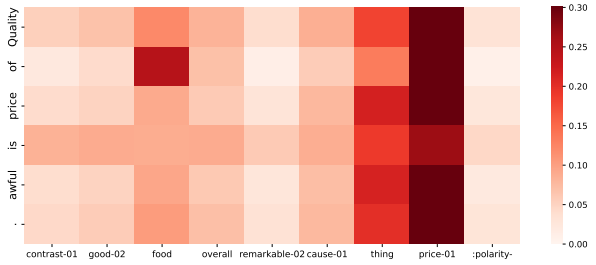
(a) Food category, model without regularizers.



(b) Food category, full model.



(c) Price category, model without regularizers.



(d) Price category, full model.

Figure 4.3: Attention scores of target sentences over AMR graph in models with and without regularizes.

Chapter 5

Conclusions

5.1 Summary

The primary goal of this thesis was to model the relations between aspect and opinion words and to incorporate the semantic information into the text generation model for Aspect Category Sentiment Analysis (ACSA). To complete this goal, we proposed an efficient method that leveraged Abstract Meaning Representation (AMR) to capture the aforementioned relations and provide high-level semantic information to the text-generation-based ACSA model. The proposed model encoded AMR through the Graph Attention Network (GAT) before integrating it into the pre-trained BART model by the new AMR Cross-Attention layers. In addition, we also introduced two new regularizers that enabled the model to extract aspect-related information from AMR within sentences containing multiple aspects. The identical regularizer enhanced the AMR Cross-Attention by minimizing the gap between the Cross-Attention weights of the AMR nodes in the AMR Cross-Attention layers and those of their corresponding words in the source sentence Cross-Attention layers. The entropy regularizer enabled the model to focus only on aspect-related AMR nodes by minimizing the information entropy of AMR Cross-Attention scores. The experimental results on three datasets Rest14, Rest14-hard, and MAMS showed that our model outperformed other state-of-the-art methods indicating the effectiveness of our method.

5.2 Future Work

Although our model achieved competitive results in ACSA, there have been some remaining problems that we have not solved yet. Firstly, we still do not consider the edge information in AMR, which can also represent the mean-

ing of sentences. Secondly, training the model consisting of new modules and the pre-trained language model is unstable. This problem is expected to be solved by building a pre-trained language model with semantic information from AMR. To adopt the seq2seq framework of the language model, we can linearize the AMR graph and concatenate it with a plain text. Conventionally, we can apply token masking on opinion words and target words as well as aligned AMR nodes in order to learn contextual information together with aspect-related information. Finally, since AMR is a promising framework for obtaining the semantic representation of a sentence, the use of AMR could be carefully exploited for other subtasks of sentiment analysis such as Aspect Category Detection or Aspect-based Sentiment Analysis.

Publication

Tu Tran, Kiyoaki Shirai, and Natthawut Kertkeidkachorn. "Text Generation Model Enhanced with Semantic Information in Aspect Category Sentiment Analysis". In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5256–5268, Toronto, Canada. Association for Computational Linguistics.

Bibliography

[1] Rafael Anchiêta and Thiago Pardo. Semantically inspired AMR alignment for the Portuguese language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1595–1600, Online, November 2020. Association for Computational Linguistics.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[3] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: fast convergence at large depth. In Cassio de Campos and Marloes H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 1352–1361. PMLR, 27–30 Jul 2021.

[4] Xuefeng Bai, Yulong Chen, and Yue Zhang. Graph pre-training for AMR parsing and generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6001–6015, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[5] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[6] Enkhbold Bataa and Joshua Wu. An investigation of transfer learning-based sentiment analysis in Japanese. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages

4652–4657, Florence, Italy, July 2019. Association for Computational Linguistics.

[7] John A. Bateman. Upper modeling: organizing knowledge for natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Generation*, Linden Hall Conference Center, Dawson, Pennsylvania, June 1990. Association for Computational Linguistics.

[8] Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. One spring to rule them both: Symmetric amr semantic parsing and generation without a complex pipeline. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12564–12573, May 2021.

[9] Austin Blodgett and Nathan Schneider. Probabilistic, structure-aware algorithms for improved variety, accuracy, and coverage of AMR alignments. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3310–3321, Online, August 2021. Association for Computational Linguistics.

[10] Deng Cai and Wai Lam. AMR parsing via graph-sequence iterative inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301, Online, July 2020. Association for Computational Linguistics.

[11] Shu Cai and Kevin Knight. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[13] Shibhansh Dohare and Harish Karnick. Text summarization using abstract meaning representation. *CoRR*, abs/1706.01678, 2017.

[14] Ramón Fernandez Astudillo, Miguel Ballesteros, Tahira Naseem, Austin Blodgett, and Radu Florian. Transition-based parsing with stack-transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1001–1007, Online, November 2020. Association for Computational Linguistics.

[15] Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206, San Diego, California, June 2016. Association for Computational Linguistics.

[16] Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. A discriminative graph-based parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[17] Gayatree Ganu, Noémie Elhadad, and Amélie Marian. Beyond the stars: Improving rating predictions using review text content. In *International Workshop on the Web and Databases*, 2009.

[18] Mozhdeh Gheini, Xiang Ren, and Jonathan May. Cross-attention is all you need: Adapting pretrained Transformers for machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1754–1765, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[19] Michael Wayne Goodman. Penman: An open-source library and tool for AMR graphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 312–319, Online, July 2020. Association for Computational Linguistics.

[20] Hardy Hardy and Andreas Vlachos. Guided neural language generation for abstractive summarization using Abstract Meaning Representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 768–773, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern*

Recognition (CVPR), pages 770–778, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society.

[22] Mengting Hu, Shiwan Zhao, Li Zhang, Keke Cai, Zhong Su, Renhong Cheng, and Xiaowei Shen. CAN: Constrained attention networks for multi-aspect sentiment analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4601–4610, Hong Kong, China, November 2019. Association for Computational Linguistics.

[23] Marcio Inácio and Thiago Pardo. Semantic-based opinion summarization. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 619–628, Held Online, September 2021. INCOMA Ltd.

[24] Qingnan Jiang, Lei Chen, Ruifeng Xu, Xiang Ao, and Min Yang. A challenge dataset and effective models for aspect-based sentiment analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6280–6285, Hong Kong, China, November 2019. Association for Computational Linguistics.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*, 2015.

[26] Paul Kingsbury and Martha Palmer. From TreeBank to PropBank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Canary Islands - Spain, May 2002. European Language Resources Association (ELRA).

[27] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.

[28] Yuncong Li, Zhe Yang, Cunxiang Yin, Xu Pan, Lunan Cui, Qiang Huang, and Ting Wei. A joint model for aspect-category sentiment analysis with shared sentiment prediction layer. In *Proceedings of the 19th Chinese National Conference on Computational Linguistics*, pages 1112–1121, Haikou, China, October 2020. Chinese Information Processing Society of China.

[29] Yuncong Li, Cunxiang Yin, Sheng-hua Zhong, and Xu Pan. Multi-instance multi-label learning networks for aspect-category sentiment analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3550–3560, Online, November 2020. Association for Computational Linguistics.

[30] Yunlong Liang, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen, and Jie Zhou. A novel aspect-guided deep transition model for aspect based sentiment analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5569–5580, Hong Kong, China, November 2019. Association for Computational Linguistics.

[31] Bin Liu, Tao Lin, and Ming Li. Enhancing aspect-category sentiment analysis via syntactic data augmentation and knowledge enhancement. *Knowledge-Based Systems*, 264:110339, 2023.

[32] Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[33] Jian Liu, Zhiyang Teng, Leyang Cui, Hanmeng Liu, and Yue Zhang. Solving aspect category sentiment analysis as a text generation task. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4406–4416, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[34] Yijia Liu, Wanxiang Che, Bo Zheng, Bing Qin, and Ting Liu. An AMR aligner tuned by transition-based parser. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages

2422–2430, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[35] Long H. B. Nguyen, Viet Pham, and Dien Dinh. Improving neural machine translation with amr semantic graphs. *Mathematical Problems in Engineering*, 2021.

[36] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005.

[37] Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. SemEval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland, August 2014. Association for Computational Linguistics.

[38] Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. Aligning English strings with Abstract Meaning Representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 425–429, Doha, Qatar, October 2014. Association for Computational Linguistics.

[39] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[40] Sebastian Ruder, Parsa Ghaffari, and John G. Breslin. A hierarchical model of reviews for aspect-based sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 999–1005, Austin, Texas, November 2016. Association for Computational Linguistics.

[41] Martin Schmitt, Simon Steinheber, Konrad Schreiber, and Benjamin Roth. Joint aspect and polarity classification for aspect-based sentiment analysis with end-to-end neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1109–1114, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[42] Yongxue Shan, Chao Che, Xiaopeng Wei, Xiaodong Wang, Yongjun Zhu, and Bo Jin. Bi-graph attention network for aspect category sentiment classification. *Knowledge-Based Systems*, 258:109972, 2022.

[43] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(4):623–656, 1948.

[44] Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. Semantic Neural Machine Translation Using AMR. *Transactions of the Association for Computational Linguistics*, 7:19–31, 03 2019.

[45] Chi Sun, Luyao Huang, and Xipeng Qiu. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 380–385, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[46] Ida Szubert, Adam Lopez, and Nathan Schneider. A structured syntax-semantics interface for English-AMR alignment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1169–1180, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[47] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Learning to attend via word-aspect associative fusion for aspect-based sentiment analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018.

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[49] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[50] Chuan Wang and Nianwen Xue. Getting the most out of AMR parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1257–1268, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[51] Peiyi Wang, Liang Chen, Tianyu Liu, Damai Dai, Yunbo Cao, Baobao Chang, and Zhifang Sui. Hierarchical curriculum learning for AMR parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 333–339, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[52] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, Austin, Texas, November 2016. Association for Computational Linguistics.

[53] Yequan Wang, Aixin Sun, Minlie Huang, and Xiaoyan Zhu. Aspect-level sentiment analysis using as-capsules. In *The World Wide Web Conference*, WWW '19, page 2033–2044. Association for Computing Machinery, 2019.

[54] Bowen Xing, Lejian Liao, Dandan Song, Jingang Wang, Fuzheng Zhang, Zhongyuan Wang, and Heyan Huang. Earlier attention? aspect-aware lstm for aspect-based sentiment analysis. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5313–5319. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[55] Dongqin Xu, Junhui Li, Muhua Zhu, Min Zhang, and Guodong Zhou. XLPT-AMR: Cross-lingual pre-training via multi-task learning for zero-shot AMR parsing and text generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 896–907, Online, August 2021. Association for Computational Linguistics.

[56] Wei Xue and Tao Li. Aspect based sentiment analysis with gated convolutional networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2514–2523, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[57] Chen Yu and Daniel Gildea. Sequence-to-sequence AMR parsing with ancestor information. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 571–577, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[58] Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. AMR parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94, Florence, Italy, July 2019. Association for Computational Linguistics.

[59] Jiawei Zhou, Tahira Naseem, Ramón Fernandez Astudillo, and Radu Florian. AMR parsing with action-pointer transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5585–5598, Online, June 2021. Association for Computational Linguistics.

[60] Peisong Zhu, Zhuang Chen, Haojie Zheng, and Tieyun Qian. Aspect aware learning for aspect category sentiment analysis. *ACM Trans. Knowl. Discov. Data*, 13(6), oct 2019.