

BARTABSA++: Revisiting BARTABSA with Decoder LLMs

Jan Pfister[✉] and Tom Völker[✉] and Anton Vlasjuk and Andreas Hotho

Data Science Chair

Center for Artificial Intelligence and Data Science (CAIDAS)

Julius-Maximilians-Universität Würzburg (JMU), Würzburg, Germany

{pfister, voelker, vlasjuk, hotho}@informatik.uni-wuerzburg.de

Abstract

We revisit the BARTABSA framework for aspect-based sentiment analysis with modern decoder LLMs to assess the importance of explicit structure modeling today. Our updated implementation—BARTABSA++¹—features architectural enhancements that boost performance and training stability. Systematic testing with various encoder-decoder architectures shows that BARTABSA++ with BART-LARGE achieves state-of-the-art results, even surpassing a finetuned GPT-4O model. Our analysis indicates the encoder’s representational quality is vital, while the decoder’s role is minimal, explaining the limited benefits of scaling decoder-only LLMs for this task. These findings underscore the complementary roles of explicit structured modeling and large language models, indicating structured approaches remain competitive for tasks requiring precise relational information extraction.

1 Introduction

In this work, we revisit BARTABSA (Yan et al., 2021a), a pointer network-based method for aspect-based sentiment analysis (ABSA) that achieved state-of-the-art performance with the BART encoder-decoder transformer - considered small by today’s standards. BARTABSA models Aspect Sentiment Triplet Extraction (ASTE) (Peng et al., 2020) as a constrained generation task, using a pointer network (Vinyals et al., 2015) to explicitly copy input tokens into strictly structured outputs.

In the meantime, advances in Large Language Models (LLMs) have transformed natural language processing (NLP) by demonstrating that implicit knowledge acquired during pretraining can often address tasks that previously demanded explicit structured representations (Brown et al., 2020; Wei et al., 2022). This shift prompts the question: Is explicit structured output modeling still

relevant in the LLM era, especially in structured representation-critical tasks like ABSA?

To evaluate this, we employ the methodology of Rothe et al. (2020), which constructs encoder-decoder architectures by reusing pretrained encoder *or* decoder checkpoints. We reimplement BARTABSA using modern libraries and incorporate architectural enhancements like: (1) feature normalization to balance embedding spaces and stabilize training (Zhang and Sennrich, 2019; Xiong et al., 2020); (2) cross-attention mechanisms reusing weights from BART’s decoder layers (Vaswani et al., 2017; See et al., 2017); (3) parametrized gating mechanisms replacing static hyperparameters with learnable weights (See et al., 2017; Chung et al., 2014; Dauphin et al., 2017). These techniques allow stable training with larger models and improve performance, outperforming both the original implementation and a baseline using a finetuned GPT-4O.

With this enhanced framework, we systematically evaluate different architectural configurations to examine potential scaling effects of modernized LLMs within structured language modeling. This research direction is particularly valuable as it combines the implicit knowledge of modern decoder LLMs with the transparency offered by the explicit copying mechanism of pointer networks—a property important for interpretable NLP systems.

Experiments with BART (Lewis et al., 2020), BERT (Devlin et al., 2019), and GPT-2 (Radford et al., 2019) variants reveal performance hinges on the encoder, with minimal decoder impact. Counterintuitively, scaling GPT-2 does not yield performance gains, emphasizing the encoder’s pretraining importance for pointer networks (and other encoder-decoder architectures).

Our study shows large autoregressive models do not universally surpass structured approaches in NLP and contributes to the dialogue on explicit structure modeling, like pointer networks, in the

[✉] These authors contributed equally to this work.

¹<https://github.com/LSX-UniWue/bartabsa-plusplus>

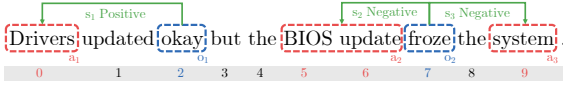


Figure 1: An example sentence for ABSA with **aspect**, **opinion**, **sentiment** and word indices annotated. Adapted from Yan et al. (2021a).

LLM era. It explores scaling structured generation with LLM-sized models to merge their implicit knowledge with structured copying mechanisms.

2 Related Work

2.1 Aspect-Based Sentiment Analysis

Aspect-Based Sentiment Analysis (ABSA) is a fine-grained approach to sentiment analysis focusing on opinions about specific text elements rather than general document or sentence sentiment (Liu, 2012). ABSA involves three main components: aspect terms, opinion terms, and sentiment polarities. In Figure 1, the term “Drivers” is characterized positively by “okay”, whereas “BIOS update” and “system” are negatively characterized by “froze”.

Aspect Sentiment Triplet Extraction (ASTE), the focus of this work, extracts (**aspect**, **opinion**, **sentiment**) triplets simultaneously. This task captures complete opinion structures, making it valuable for applications such as customer feedback analysis (Peng et al., 2020).

The ABSA field has evolved from specialized RNN-based architectures (Wang et al., 2016; Tang et al., 2016) to approaches leveraging pretrained language models (Li et al., 2019). For ASTE specifically, approaches progressed from pipeline methods to joint tagging schemes (Xu et al., 2020; Wu et al., 2020), with influential work reformulating the task as a structured generation problem using sequence-to-sequence models (Yan et al., 2021a; Zhang et al., 2021). This sequence-to-sequence paradigm established a strong foundation that subsequent research has built upon, including recent advances with larger language models and hybrid approaches (Xianlong et al., 2023; Zhang et al., 2024; Sun et al., 2024).

Our experiments use the refined versions of standard ABSA benchmark datasets from SemEval 2014-2016 challenges (Pontiki et al., 2014, 2015, 2016): 14lap, 14res, 15res, and 16res, as processed by Xu et al. (2020) who filtered out low-quality examples and duplicates.

2.2 BARTABSA

Following Yan et al. (2021a), each ABSA task involves transforming an input sentence $X = [x_1, \dots, x_n]$ into a structured target sequence $Y = [y_1, \dots, y_m]$ using pointer networks (Vinyals et al., 2015). These allow the model to refer directly to tokens from the input sequence, facilitating extraction tasks requiring grounding in exact input spans.

For ABSA, BARTABSA uses these pointers to copy start and end token indices from the input sentence into a structured output sequence, paired with sentiment classification tokens. For instance, the input “Drivers updated okay but the BIOS update froze the system.” (Figure 1) is converted into structured outputs pointing to input tokens and sentiment classes.

Formally, the output follows a fixed grammar where each triplet is represented as a 5-tuple: $(a_i^s, a_i^e, o_i^s, o_i^e, s_i^p)$ where a_i^s, a_i^e are start and end indices of the i th **aspect** term, o_i^s, o_i^e are for the **opinion** term, and s_i^p is the **sentiment polarity** class. For our example, the structured output sequence thus becomes:

$$[(0, 0, 2, 2, \text{POS}), (5, 6, 7, 7, \text{NEG}), (9, 9, 7, 7, \text{NEG})]$$

This syntax explicitly encodes the extracted (**aspect**, **opinion**, **sentiment**) triplets as structured predictions, with each span being clearly represented by its start and end indices in the input sequence. Given this structured output format, BARTABSA models ABSA as a sequence generation task, computing the probability of Y given X as:

$$P(Y|X) = \prod_{t=1}^m P(y_t|X, Y_{<t}) \quad (1)$$

BART (Lewis et al., 2020) is used as a backbone to compute $P(y_t|X, Y_{<t})$, with the encoder processing the input sequence X into contextualized embeddings H^e :

$$H^e = \text{BARTEncoder}([x_1, \dots, x_n]) \quad (2)$$

Decoding involves predicting indices y_t and dereferencing them to tokens \hat{y}_t :

$$\hat{y}_t = \begin{cases} X_{y_t} & \text{if } y_t \text{ is a pointer index} \\ C_{y_t-n} & \text{if } y_t \text{ is a class index} \end{cases} \quad (3)$$

where $C = [c_1, \dots, c_l]$ is a list of class tokens for sentiment classification.

The decoder uses prior dereferenced tokens $\hat{Y}_t = [\hat{y}_1, \dots, \hat{y}_{t-1}]$ to produce the next state:

$$h_t^d = \text{BARTDecoder}(H^e, \hat{Y}_t) \quad (4)$$

On top of the BART model, BARTABSA then uses a pointer network mechanism to generate a distribution over input tokens X and class tokens C : This mechanism combines both the initial token embeddings and the contextualized representations from the transformer layers:

$$E^e = \text{BARTTokenEmbed}(X) \quad (5)$$

$$\hat{H}^e = \text{MLP}(H^e) \quad (6)$$

$$\bar{H}^e = \alpha \hat{H}^e + (1 - \alpha) E^e \quad (7)$$

$$C^d = \text{BARTTokenEmbed}(C) \quad (8)$$

$$P_t = \text{Softmax}([\bar{H}^e; C^d] h_t^d) \quad (9)$$

where P_t represents the common pointer and class token distribution.

Training uses teacher forcing and negative log-likelihood; inference can use strategies like greedy sampling and beam search (Yan et al., 2021a).

3 Methodology

We will combine Yan et al.’s (2021a) BARTABSA framework, with Rothe et al.’s (2020) encoder-decoder model construction methodology in Section 3.3. This requires us to first reimplement BARTABSA in Section 3.1, before stabilizing training with additional features and optimizations in Section 3.2.

3.1 BARTABSA-R: Our Reimplementation

Yan et al.’s (2021a) original BARTABSA implementation has limitations for our use case, especially hindering model extensions: 1) The *fastnlp* package² used is unmaintained, only supporting older 🤖 Transformers versions (Wolf et al., 2020). 2) Data processing and modeling are tightly coupled, with e.g. hard-coded token IDs for mask creation, complicating further experiments and architecture adaptations. To resolve these issues, we reimplement BARTABSA using maintained libraries like PyTorch (Ansel et al., 2024), Lightning (Falcon and The PyTorch Lightning team, 2019), and an updated Transformers version (Wolf et al., 2020). We avoid hard-coded values with tokenizer outputs and shift attention mask creation to the data pipeline. We name our reimplementation BARTABSA-R.

²<https://github.com/fastnlp/fastNLP>

3.2 Extending to BARTABSA++

During the reimplementation of BARTABSA and preliminary experiments, we identified several issues and potential improvements, detailed also in Algorithm 1.

3.2.1 Feature Normalization

Our use of some backbone models, like BART-LARGE, resulted in unstable training, seen through diverging losses. We traced this instability to differing scales of output logits (Figure 2), stemming from concatenating distinct embedding spaces in the pointer network: classification token embeddings (C^d) and the mixed encoder representations (\bar{H}^e). The scale imbalance between the two spaces destabilized attention calculations.

To address this, we applied an L^2 normalization to both special token embeddings and processed encoder outputs (lines 6 & 7), equalizing their contributions, thus stabilizing training³.

An RMSNorm (Zhang and Sennrich, 2019) was also applied to the final decoder output (line 13), further stabilizing gradient flow.

3.2.2 Additional Attention Mechanism

We introduce an additional attention mechanism motivated by an architectural analysis of BARTABSA. While exploring component contributions, we discovered that removing the encoder’s MLP (line 3; Equation (6))—a seemingly auxiliary component—significantly degrades performance (a drop of ≈ 4.3 p.P. across datasets). This insight led us to incorporate an additional processing step for the decoder’s last hidden states—analogueous to the encoder’s MLP.

Based on previous work on pointer networks by See et al. (2017), we opted to add a cross-attention (Vaswani et al., 2017; Bahdanau et al., 2016) on top that resembles See et al.’s (2017) idea around context vectors by computing attention scores between the decoder output $H^d \in \mathbb{R}^{m \times d}$ (as the *query*) and our concatenated encoder representation $X^e \in \mathbb{R}^{n \times d}$ (as *key* and *value*). Internally, this cross-attention follows Vaswani et al.’s (2017) formulation for multi-head-attention, instead of Bahdanau et al.’s (2016) to follow modern transformers such as BART (Lewis et al., 2020) more closely.

For efficiency and to maintain consistent attention mechanisms across model variants, we imple-

³Preliminary experiments show, this also solves the instabilities identified by Pfister et al. (2022) when training an mT5 model (Xue et al., 2021) using the BARTABSA framework.

Algorithm 1 Our BARTABSA++ with improvements (Section 3.2) over the original BARTABSA algorithm in Yan et al. (2021a). The improvements, marked in red, consist of a parametrized gating mechanism (Section 3.2.3), additional normalization (Section 3.2.1), and an attention mechanism on top of the decoder (Section 3.2.2).

Input: X - Input, C - Special Tokens, \hat{Y}_t - Remapped Token Indices To Tokens

- 1: $E^e \leftarrow \text{BARTTokenEmbed}(X)$
- 2: $H^e \leftarrow \text{BARTEncoder}(X)$
- 3: $\hat{H}^e \leftarrow \text{MLP}(H^e)$
- 4: $\gamma \leftarrow \text{Gating}(\hat{H}^e, E^e)$
- 5: $\bar{H}^e \leftarrow \gamma \hat{H}^e + (1 - \gamma) E^e$
 \triangleright Parametrized gating mechanism avoiding the need for a hyperparameter α (Section 3.2.3).
- 6: $C^d \leftarrow \text{Norm}(\text{BARTTokenEmbed}(C))$
- 7: $\bar{H}^e \leftarrow \text{Norm}(\bar{H}^e)$
- 8: $X^e \leftarrow [C^d; \bar{H}^e]$
 $\triangleright L^2$ normalization to ensure stable training (Section 3.2.1).
- 9: $H^d \leftarrow \text{BARTDecoder}(H^e, \hat{Y}_t)$
- 10: $\hat{H}^d \leftarrow \text{BARTDecoder}_{\text{last_layer}}(H^d, X^e)$
 \triangleright Reusing BART’s built-in cross-attention mechanism (Section 3.2.2).
- 11: $\omega \leftarrow \text{Gating}(\hat{H}^d, H^d)$
- 12: $\bar{H}^d \leftarrow \omega \hat{H}^d + (1 - \omega) H^d$
 \triangleright Parametrized gating mechanism similar to the encoder counterpart (Section 3.2.3).
- 13: $\bar{H}^d \leftarrow \text{LayerNorm}(\bar{H}^d)$
 $\triangleright RMSNorm$ to ensure stable training (Section 3.2.1).
- 14: $P_t \leftarrow \text{Softmax}(X^e \cdot \bar{H}^d)$
- 15: **return** P_t

ment this attention by reusing the weights from the pre-trained cross-attention module in the final decoder layer of the BART model (line 10). This parameter sharing approach eliminates the need for additional parameters (Lan et al., 2020) while ensuring architectural compatibility when extending our approach to different encoder-decoder frameworks in Section 3.3.

3.2.3 Parametrized Gating Mechanism

To enhance model flexibility and bypass manual hyperparameter tuning, we add two learnable gating mechanisms (e.g. thus α in Equation (7) gets “learnable”, after the impact of α remained unexamined by Yan et al. (2021a)). This modification eliminates the need to manually tune weighting parameters during experimentation, while enabling the model to adaptively determine optimal information flow based on the specific input context. Drawing inspiration from gating mechanisms in recurrent architectures (Chung et al., 2014), their successful application in language modeling (Dauphin et al., 2017) and specifically the Pointer Generator Mechanism (See et al., 2017), we implement two

distinct gating modules.

The first gate substitutes α in the encoder path (lines 4 & 5):

$$\gamma = \sigma([\hat{H}^e; E^e] W_{\text{enc}}^T + b_{\text{enc}}) \quad (10)$$

$$\bar{H}^e = \gamma \odot \hat{H}^e + (1 - \gamma) \odot E^e \quad (11)$$

where $\hat{H}^e, E^e \in \mathbb{R}^{n \times d}$ are MLP-processed encoder outputs and token embeddings, with W_{enc} and b_{enc} being learnable. A decoder-side gate combines cross-attention output \hat{H}^d with original hidden states H^d (lines 11 & 12), having its own parameters.

These mechanisms enable interpolation between inputs, dynamically adjusting representation contributions.

3.3 Scaling and Extending BARTABSA++

While BARTABSA was originally based on the pre-trained encoder-decoder transformer BART (Lewis et al., 2020), recent advances in large language models (LLMs) prompt the question of its applicability to other models and especially architectures.

To address this question, we follow the methodology proposed by Rothe et al. (2020): we adapt

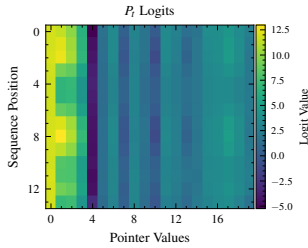


Figure 2: Heatmap of the untrained model’s pointer logit distribution (P_i). The visualization reveals a strong initialization bias towards the lower pointer values corresponding to the special tokens.

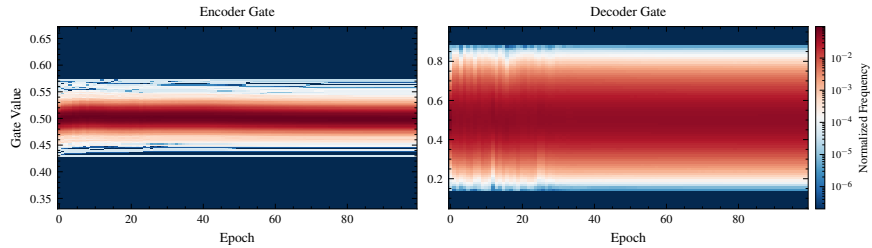


Figure 3: Evolution of gate value distributions over 100 epochs of training. **Left:** Encoder gate values remain tightly clustered around the initialization value of 0.5 with minimal deviation (σ 0.03), suggesting rather limited utilization of the gating mechanism in the encoder. **Right:** Decoder gate values show significantly higher variance (σ 0.15) and clear divergence from initialization, indicating that the model actively leverages the gating mechanism in the decoder to modulate information flow between attention heads and feedforward components.

Table 1: Performance comparison between literature baselines, our GPT-4O finetune, as well as BARTABSA-R and BARTABSA++ (using BART-BASE and BART-LARGE). Tuple level (P)recision, (R)ecall, and F_1 are reported per dataset and averaged across them, with standard deviations in parentheses. The best results per column are **bolded**.

Model	14res			14lap			15res			16res			Avg		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Fine-Tuned GPT-4O	74.65	79.38	76.94	61.67	68.39	64.86	61.91	73.40	67.17	71.38	78.60	74.81	67.40	74.94	70.95
Xianlong et al. (2023)	77.38	72.86	75.05	65.11	62.20	64.53	70.23	65.73	67.90	76.37	76.85	76.61	72.27	69.41	71.02
BARTABSA (Yan et al., 2021a)	65.52	64.99	65.25	61.41	56.19	58.69	59.14	59.38	59.26	66.60	68.68	67.62	63.17	62.31	62.70
Our BARTABSA-R (BART-BASE)	76.43 (σ 1.70)	73.56 (σ 3.41)	74.94 (σ 2.50)	66.24 (σ 2.62)	59.32 (σ 4.18)	62.53 (σ 3.13)	64.46 (σ 1.56)	60.33 (σ 3.26)	62.30 (σ 2.30)	68.52 (σ 2.64)	68.05 (σ 1.83)	68.26 (σ 1.84)	68.91 (σ 2.13)	65.31 (σ 3.17)	67.01 (σ 2.44)
Our BARTABSA++ (BART-BASE)	77.76 (σ 1.04)	75.81 (σ 1.75)	76.74 (σ 1.09)	68.27 (σ 1.29)	61.98 (σ 1.45)	64.96 (σ 1.08)	65.33 (σ 2.08)	63.27 (σ 1.21)	64.27 (σ 1.62)	69.82 (σ 1.27)	69.24 (σ 1.60)	69.51 (σ 1.26)	70.29 (σ 1.42)	67.58 (σ 1.50)	68.87 (σ 1.26)
Our BARTABSA-R (BART-LARGE)	79.13 (σ 1.92)	75.79 (σ 4.42)	77.32 (σ 2.24)	56.33 (σ 31.54)	52.09 (σ 29.18)	54.11 (σ 30.30)	64.63 (σ 7.51)	64.18 (σ 8.56)	64.35 (σ 7.79)	74.99 (σ 3.34)	75.13 (σ 4.46)	75.05 (σ 3.78)	68.77 (σ 11.08)	66.80 (σ 11.65)	67.71 (σ 11.03)
Our BARTABSA++ (BART-LARGE)	80.26 (σ 1.21)	81.43 (σ 1.48)	80.82 (σ 1.02)	70.85 (σ 0.73)	64.94 (σ 1.42)	67.75 (σ 0.64)	67.44 (σ 1.39)	67.63 (σ 1.76)	67.52 (σ 1.14)	76.08 (σ 2.32)	77.10 (σ 1.70)	76.58 (σ 1.94)	73.66 (σ 1.41)	72.77 (σ 1.59)	73.16 (σ 1.18)

pretrained encoder-only and decoder-only models into encoder-decoder formats, modifying attention masks and incorporating new cross-attention blocks akin to Vaswani et al.’s (2017). The Transformers library (Wolf et al., 2020) offers model combinations like BERT (Devlin et al., 2019), ROBERTA (Liu et al., 2019), and GPT-2 (Radford et al., 2019), sufficing for our experiments.

Our reimplementations and improvements upon BARTABSA (Sections 3.1 and 3.2) provide the necessary flexibility to reuse pretrained checkpoints as encoder-decoder models.

4 Experiments

We evaluate our approaches on four popular SemEval ABSA benchmark datasets: 14lap, 14res, 15res, and 16res (Pontiki et al., 2014, 2015, 2016), using the refined version from Xu et al. (2020) and commonly used tuple-level metrics (Yan et al., 2021a). Our backbone models are

BART-BASE and BART-LARGE, with full implementation details in Appendix A.

We compare our results against the original BARTABSA (Yan et al., 2021a) and the state-of-the-art by Xianlong et al. (2023), who achieve their results using a mixture of sequence tagging and sequence generation.

To assess how the advancing capabilities of LLMs might impact the ASTE task, we additionally establish an LLM-based baseline by fine-tuning OpenAI’s GPT-4O model⁴ on each ABSA dataset. Using the prompt template shown in Appendix B, we train for 3 epochs with consistent hyperparameters across all datasets. This (methodologically) very simple baseline achieves an average F_1 score of 70.95, roughly on par with the current SOTA results. In particular, the LLM-based approach exhibits a distinctive pattern of trading

⁴<https://platform.openai.com/docs/models/gpt-4o>, snapshot gpt-4o-2024-08-06

precision for recall, with significantly higher recall (74.94) compared to other methods.

4.1 Our Modern Reimplementation (3.1)

We find our reimplementation (BARTABSA-R) of the BARTABSA framework to perform much better than the original implementation by Yan et al. (2021a) (Table 1). This comes rather unexpected, as to the best of our knowledge, both codebases implement the exact same algorithm. To rule out evaluation issues, we employ the same tuple-level evaluation script as published by Yan et al., where a prediction is considered correct only when all components match the ground truth exactly. Interestingly, we find that the performance discrepancies vary between the datasets (nearly 10 p.P. for 14res, and only about 2 p.P. for 16res).

4.1.1 Potential Differences

Whilst we were unable to pinpoint the exact source for this discrepancy, we validated these findings through multiple steps: 1) We independently reproduced the original results by rerunning the authors’ published code. 2) We thoroughly debugged the data loaders to confirm that both implementations receive identical inputs, thereby eliminating data processing discrepancies as a potential explanation. 3) We maintained consistent hyperparameter settings across both implementations. 4) We used their evaluation script to rule out issues during evaluation.


Given these controls, the performance differences most likely either stem from undetected issues in their implementation of the algorithm or simply our use of a modernized tech stack using a different framework, newer transformers, torch and CUDA versions as well as a different attention implementation within the BART models (Ansel et al., 2024; Dao et al., 2022).

4.1.2 Scaling to BART-LARGE

As a first step toward scaling this approach, we swap the BART-BASE for a BART-LARGE backbone. This gives us mixed results (BART-BASE vs. BART-LARGE in Table 1): while switching for the larger backbone mostly improves the scores (+0.70 F_1 on average), it also instabilizes training, which can be seen from the larger standard deviations (σ 11.03) and even the complete performance breakdown on “14lap”. As already mentioned in Section 3.2.1, these instabilities can be mitigated by using our improvements to the architecture.

Table 2: Model performance and ablation study using BART-LARGE. (P)recision, (R)ecall, and F_1 averaged across the four datasets, with standard deviations in parentheses. Red indicates performance decrease compared to our full architecture (described in Section 3.2).


Model	Avg		
	P	R	F_1
Our BARTABSA-R	68.77 (σ 11.08) (-4.89)	66.80 (σ 11.65) (-5.97)	67.71 (σ 11.03) (-5.45)
No Encoder Normalization	67.03 (σ 10.34) (-6.63)	65.39 (σ 10.68) (-7.38)	66.16 (σ 10.41) (-7.00)
No Final RMS-Norm	73.28 (σ 2.07) (-0.38)	72.64 (σ 2.27) (-0.13)	72.90 (σ 2.01) (-0.26)
No additional Attention	71.36 (σ 4.61) (-2.30)	70.78 (σ 4.34) (-1.99)	71.02 (σ 4.36) (-2.14)
No Decoder Gating	62.20 (σ 17.34) (-11.46)	60.60 (σ 15.98) (-12.17)	61.33 (σ 16.40) (-11.83)
No Encoder Gating	72.08 (σ 2.75) (-1.58)	71.94 (σ 2.44) (-0.83)	71.98 (σ 2.30) (-1.18)
Our BARTABSA++	73.66 (σ 1.41)	72.77 (σ 1.59)	73.16 (σ 1.18)

 **Surprising:** Our reimplementation significantly outperforms the literature, but it suffers instabilities when scaling to larger models.

4.2 Our Improvements (3.2)

Scaling pointer networks to larger models presents significant challenges, particularly regarding training stability. During our preliminary experiments with BART-LARGE, we observed unstable optimization dynamics stemming from substantial magnitude differences between input token pointer logits and classification token logits (Figure 2). This is naturally mitigated by our introduced normalization (Section 3.2.1).

Our introduced architectural enhancements coined BARTABSA++ address these challenges systematically, as demonstrated by the performance improvements in Table 1 with an average 1.86 points F_1 increase for BART-BASE and +5.45 F_1 points for BART-LARGE, highlighting the greater benefits of our improvements for larger models. In fact, we find our BARTABSA++ not only outperforms our finetuned GPT-4O, but also represents the new SOTA for the ABSA triplet extraction task (averaged across the four datasets). To understand the individual contribution of each component, we conduct a comprehensive ablation study using the BART-LARGE model (Table 2), revealing several key insights. We show our improvements also generalize, when applying this methodology to other structured extraction tasks in Appendix D.

 **Surprising:** Our structured language modeling BARTABSA++ significantly outperforms a much more parameter heavy GPT-4O finetune and even sets the new state of the art.

4.2.1 Component Analysis

Normalization (3.2.1) The feature normalization proves crucial for model stability, as removing the encoder normalization results in a notable performance decline (-7.00 F₁ points) coupled with substantially increased variance across runs (σ 10.41 compared to σ 1.18 for BARTABSA++). The effect of this normalization on the final model output is also apparent in the comparisons of the value heatmaps (Figures 2 and 4), which shows a clear bias towards the prepended special tokens without the normalization step. This confirms our hypothesis that normalizing representation spaces is essential for stable optimization. While the final layer normalization contributes more modestly (-0.26 F₁ points when removed), it also helps in stabilizing training results, almost halving the standard deviation.

Attention Mechanism and Decoder Gating

(3.2.2, 3.2.3) The additional cross-attention mechanism combined with its corresponding decoder gating mechanism also provides a significant performance improvement. Removing the attention mechanism alone reduces performance by 2.14 F₁ points, while removing only the decoder gating (but keeping the attention) causes a drastic drop of 11.83 F₁ points with extreme instability (σ 16.40).

This indicates that while the additional processing from the attention mechanism is beneficial, it requires controlled integration through the gating mechanism to be effective. An analysis of the decoder gate values in Figure 3 show considerable variance during training, confirming that the model actively uses this mechanism to regulate information flow between the original decoder output and the attention-processed representation.

Encoder Gating (3.2.3) The encoder gating mechanism has a modest impact (-1.18 F₁ points when removed), which aligns with the observation in Figure 3 that the encoder gates values remain closer to their initial value of 0.5 throughout training.

4.3 Synthetic Encoder-Decoder Models (3.3)

As a first step towards transferring the pointer methodology to LLMs, we sanity check the com-

Table 3: Performance comparison across different synthetic encoder-decoder models. Models with ++ indicate the BARTABSA++ architecture.

Model	Avg		
	P	R	F ₁
Reimpl. BART++	70.29 (σ 1.42)	67.58 (σ 1.50)	68.87 (σ 1.26)
Roberta2Roberta	57.70 (σ 23.84)	54.04 (σ 23.87)	55.66 (σ 24.00)
Roberta2Roberta++	68.92 (σ 6.17)	64.53 (σ 7.75)	66.54 (σ 6.83)
Roberta2GPT2	67.64 (σ 1.96)	61.78 (σ 2.36)	64.50 (σ 2.05)
Roberta2GPT2++	69.55 (σ 1.98)	65.54 (σ 1.95)	67.44 (σ 1.74)
Bert2Bert++	66.19 (σ 1.62)	61.27 (σ 1.95)	63.58 (σ 1.63)
Bert2GPT2++	65.58 (σ 1.82)	57.81 (σ 1.35)	61.38 (σ 1.35)
RobertaLarge2-RobertaLarge++	03.41 (σ 7.62)	00.95 (σ 2.13)	01.49 (σ 3.33)
RobertaLarge2-GPT2Medium ++	69.62 (σ 1.18)	65.27 (σ 2.27)	67.33 (σ 1.57)

binations of BARTABSA++ and the methodology introduced in Rothe et al. (2020).

4.3.1 Overall

For this, we first take the base models explored in the original paper (BERT, ROBERTA, and GPT-2) by Rothe et al. (2020) and evaluate how well the two methodologies interact (Table 3).

The approach works successfully, with some combinations like ROBERTA2ROBERTA even outperforming the original BARTABSA results. However, none of these synthetic combinations surpasses our enhanced BARTABSA++ even with BART-BASE as the backbone.

Our architectural enhancements consistently improve performance across all synthetic models, with particularly dramatic stabilization effects for ROBERTA2ROBERTA (reducing variance from σ 24.00 to σ 6.83 while improving F₁ by +10.88 points). Generally, ROBERTA outperforms BERT as an encoder, and pairing a ROBERTA encoder with a GPT-2 decoder yields better results than using ROBERTA for both components. The only exception is the highly unstable ROBERTALarge2ROBERTALarge configuration, which fails to train effectively despite our enhancements.

4.3.2 Scaling to Modern Sizes

In order to expand our analysis to decoder LLMs, and size ranges which cross the threshold into what is commonly acknowledged as the LLM regime (Zhao et al., 2025; Kaplan et al., 2020), we run a scaling benchmark using only GPT-2 in Table 4.

For this, we use GPT-2 models ranging from base (137M) to XL (1.6B) for both: the encoder

Table 4: Performance of different GPT-2 model sizes, with the pretrained weights being used in the encoder and decoder part of the synthetic model.

Model	Params	Avg		
		P	R	F ₁
Our BARTABSA-R	139M	68.91 (σ 2.13)	65.31 (σ 3.17)	67.01 (σ 2.44)
GPT2GPT Base	277M	58.92 (σ 2.35)	52.11 (σ 1.81)	55.26 (σ 1.73)
GPT2GPT Medium	810M	60.06 (σ 2.01)	53.99 (σ 2.68)	56.80 (σ 2.19)
GPT2GPT Large	1.78B	59.78 (σ 2.28)	53.84 (σ 1.96)	56.60 (σ 1.76)
GPT2GPT XL	3.61B	59.72 (σ 0.97)	53.38 (σ 3.71)	56.34 (σ 2.45)


Table 5: Impact of random weight initialization on model performance.

Model	Avg		
	P	R	F ₁
Full Pretrained	68.91 (σ 2.13)	65.31 (σ 3.17)	67.01 (σ 2.44)
Random Encoder	18.35 (σ 12.57)	13.42 (σ 9.04)	15.45 (σ 10.44)
Random Decoder	65.57 (σ 1.51)	61.57 (σ 2.28)	63.46 (σ 1.88)
Random Both	36.53 (σ 2.57)	27.57 (σ 1.62)	31.38 (σ 1.83)

and the decoder. As the parameters get duplicated, this creates models with up to 3.6B parameters overall—including the added cross attention.

We find basically no “scaling effects” at all, meaning the performance mostly does not increase with model size increases. Furthermore, the model itself performs substantially worse than our basic reimplementation, or the original BARTABSA results. This draws the applicability of this combination of methodologies to combine pointer networks with (large) decoder LLMs into question.

To ensure that the lack of pretraining for the newly initialized weights isn’t responsible for this unexpected non-scaling behavior, we conduct additional experiments pretraining these models on the CNN/DailyMail summarization dataset (See et al., 2017; Hermann et al., 2015) before fine-tuning on ABSA (Appendix E). These experiments confirm our findings, showing similar patterns of non-scaling and instability even after extensive pre-training.


 **Negative Result:** (Larger) decoder LMs show no benefit in this structured prediction framework, and even exhibit slightly negative scaling effects.

4.3.3 Ablation using Random Initialized Models

In order to better understand this unexpected finding of non-scaling, we analyze which parts of the newly crafted encoder-decoder influence the performance the most. We hypothesize that there is

an inherent difference between the encoder and decoder architecture, especially when either is converted into the other one.

To be able to analyze the impact of the encoder and decoder separately, we initialize the encoder, the decoder or both with random weights (except for the token embeddings) and then train as before (Table 5). Interestingly, we find that entirely randomizing the decoder and training it from scratch has a surprisingly small impact on overall model performance. In contrast, random initialization of the encoder severely degrades performance. This is in line with our previous findings: the encoder’s “token-level representational quality” significantly outweighs the decoder’s contribution to overall performance (Kasai et al., 2021). Consequently, models pretrained explicitly as encoders consistently outperform those in which a decoder is retrained as an encoder (Pfister and Hotho, 2024; Reimers, 2022).

 **Confirmation:** As identified by Kasai et al. (2021), encoder-decoder model performance strongly correlates with encoder representational strength.

5 Conclusion

In this work, we revisited the BARTABSA framework in the context of modern decoder LLMs. Our enhanced implementation—BARTABSA++—demonstrates that explicit structured models remain highly competitive for tasks requiring precise extraction of relational information, even outperforming a finetuned GPT-4O model.

Our systematic experiments reveal a fundamental insight: while our architectural enhancements enable effective scaling with pretrained encoder-decoder models like BART, this scaling behavior does not transfer encoder-decoder models based on decoder-only LLMs. The critical factor appears to be the encoder’s representational quality at the token level, which decoder LLMs struggle to match when repurposed as encoders.

These findings highlight the complementary nature of structured approaches and large language models in NLP. While LLMs excel at tasks leveraging their implicit knowledge, structured pointer networks seem to be able to provide superior performance and interpretability for precise relational extraction tasks.

Limitations

Optimization and Architectural Constraints

Although we employed normalization and gating mechanisms to mitigate training instability, certain model configurations, notably ROBERTA2ROBERTA, still exhibited high variance across runs.⁵ Moreover, our decoder LLM experiments were restricted to GPT-2 variants due to limitations in the existing implementation in the 🤗 Transformers library. While GPT-2 showed non-scaling behavior, preliminary experiments by us suggest that newer decoder-only architectures also face similar issues, although differences in their architectures might yield varying outcomes.

Representational Limitations of Decoder-only Models

Our methodology for adapting decoder-only models into encoders, based on existing work (Rothe et al., 2020), likely does not fully resolve fundamental representational constraints for token-level tasks. Recent adaptations like LLM2Vec (BehnamGhader et al., 2024) suggest promising techniques that might overcome these limitations, though exploring such adaptations was beyond the scope of our experiments.

Scope of Pointer Networks and Structured Prediction

Our analysis focuses specifically on encoder-decoder architectures and their components, reflecting the typical formulation used in Pointer Networks. However, our findings indicate that extending the pointer paradigm to decoder-only architectures could potentially better leverage pretrained LLMs for structured prediction tasks, presenting a valuable direction for future research.

Baseline Limitations Our GPT-4O-based baseline experiments employed a straightforward prompt template primarily to provide a comparative reference point close to the previous SOTA. While adequate for this purpose, the adopted approach does not explore advanced prompting methods, chain-of-thought reasoning, or specialized instruction-tuning strategies that could further boost the capabilities of modern LLMs.

Generalizability Although we demonstrate that structured methods such as BARTABSA++ remain competitive for ABSA tasks, our results may not generalize across all structured prediction tasks in

⁵We observed similar instability when training this configuration on the summarization task (Appendix E), suggesting a potential limitation of the approach by Rothe et al. (2020).

NLP, particularly those involving different structural characteristics or task-specific constraints.

Acknowledgements

This publication was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under the project LitBERT, project no. 529659926. The authors gratefully acknowledge the HPC resources provided by the JuliaV2 cluster at the Universität Würzburg (JMU), which was funded as DFG project as “Forschungsgroßgerät nach Art 91b GG” under INST 93/1145-1 FUGG. The data science chair is part of the CAIDAS, the Center for Artificial Intelligence and Data Science, and is supported by the Bavarian High-Tech Agenda, which made this research possible.

References

- Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhersch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. 2024. [PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation](#). In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. [Neural machine translation by jointly learning to align and translate](#).
- Jeremy Barnes, Laura Oberlaender, Enrica Troiano, Andrey Kutuzov, Jan Buchmann, Rodrigo Agerri, Lilja Øvrelid, and Erik Velldal. 2022. [SemEval 2022 task 10: Structured sentiment analysis](#). In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 1280–1295, Seattle, United States. Association for Computational Linguistics.
- Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. [LLM2vec: Large language models are secretly powerful text encoders](#). In *First Conference on Language Modeling*.

- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Matusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20*, Red Hook, NY, USA. Curran Associates Inc.
- Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. [Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling](#).
- Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17*, page 933–941. JMLR.org.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- William Falcon and The PyTorch Lightning team. 2019. [PyTorch Lightning](#).
- Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. [Teaching machines to read and comprehend](#). In *NIPS*, pages 1693–1701.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. [Scaling laws for neural language models](#).
- Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah Smith. 2021. [Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation](#). In *International Conference on Learning Representations*.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. [Albert: A lite bert for self-supervised learning of language representations](#). In *International Conference on Learning Representations*.
- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. [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. Association for Computational Linguistics.
- Xin Li, Lidong Bing, Wenxuan Zhang, and Wai Lam. 2019. [Exploiting BERT for end-to-end aspect-based sentiment analysis](#). In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 34–41, Hong Kong, China. Association for Computational Linguistics.
- Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Springer International Publishing, Cham.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized bert pretraining approach](#).
- Haiyun Peng, Lu Xu, Lidong Bing, Fei Huang, Wei Lu, and Luo Si. 2020. [Knowing What, How and Why: A Near Complete Solution for Aspect-Based Sentiment Analysis](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8600–8607.
- Jan Pfister and Andreas Hotho. 2024. [SuperGLEBer: German language understanding evaluation benchmark](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7904–7923, Mexico City, Mexico. Association for Computational Linguistics.
- Jan Pfister, Sebastian Wankerl, and Andreas Hotho. 2022. [SenPoi at SemEval-2022 task 10: Point me to your opinion, SenPoi](#). In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 1313–1323, Seattle, United States. Association for Computational Linguistics.
- Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphee De Clercq, Veronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Nuria Bel, Salud Marıa Jimenez-Zafra, and Gulsen Eryigit. 2016. [SemEval-2016 task 5: Aspect based sentiment analysis](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 19–30, San Diego, California. Association for Computational Linguistics.
- Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015.

- SemEval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495, Denver, Colorado. Association for Computational Linguistics.
- Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. [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. Association for Computational Linguistics.
- Behrang QasemiZadeh and Anne-Kathrin Schumann. 2016. [The ACL RD-TEC 2.0: A language resource for evaluating term extraction and entity recognition methods](#). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*, pages 1862–1868, Portorož, Slovenia. European Language Resources Association (ELRA).
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- Nils Reimers. 2022. [Openai gpt-3 text embeddings - really a new state-of-the-art in dense text embeddings?](#) Accessed: 2025-03-14.
- Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. [Leveraging pre-trained checkpoints for sequence generation tasks](#). *Transactions of the Association for Computational Linguistics*, 8:264–280.
- Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. [Get to the point: Summarization with pointer-generator networks](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.
- Sasha Spala, Nicholas Miller, Franck Dernoncourt, and Carl Dockhorn. 2020. [SemEval-2020 task 6: Definition extraction from free text with the DEFT corpus](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 336–345, Barcelona (online). International Committee for Computational Linguistics.
- Qiao Sun, Liujia Yang, Minghao Ma, Nanyang Ye, and Qinying Gu. 2024. [MiniConGTS: A near ultimate minimalist contrastive grid tagging scheme for aspect sentiment triplet extraction](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2817–2834, Miami, Florida, USA. Association for Computational Linguistics.
- Duyu Tang, Bing Qin, and Ting Liu. 2016. [Aspect level sentiment classification with deep memory network](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 214–224, Austin, Texas. Association for Computational Linguistics.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. [Pointer networks](#). In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.
- Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. [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. Association for Computational Linguistics.
- Sebastian Winkerl, Jan Pfister, Andrzej Dulny, Gerhard Götz, and Andreas Hotho. 2025. Identifying axiomatic mathematical transformation steps using tree-structured pointer networks. *Transactions on Machine Learning Research*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Zhen Wu, Chengcan Ying, Fei Zhao, Zhifang Fan, Xinyu Dai, and Rui Xia. 2020. [Grid tagging scheme for aspect-oriented fine-grained opinion extraction](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2576–2585, Online. Association for Computational Linguistics.
- Julia Wunderle, Julian Schubert, Antonella Cacciatore, Albin Zehe, Jan Pfister, and Andreas Hotho. 2024. OtterlyObsessedWithSemantics at SemEval-2024 task 4: Developing a hierarchical multi-label classification head for large language models. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, page 602–612, Mexico City, Mexico. Association for Computational Linguistics.

- Luo Xianlong, Meng Yang, and Yihao Wang. 2023. [Tagging-assisted generation model with encoder and decoder supervision for aspect sentiment triplet extraction](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2078–2093, Singapore. Association for Computational Linguistics.
- Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tiejian Liu. 2020. On layer normalization in the transformer architecture. In *International conference on machine learning*, pages 10524–10533. PMLR.
- Lu Xu, Hao Li, Wei Lu, and Lidong Bing. 2020. [Position-aware tagging for aspect sentiment triplet extraction](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2339–2349, Online. Association for Computational Linguistics.
- Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. [mT5: A massively multilingual pre-trained text-to-text transformer](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.
- Hang Yan, Junqi Dai, Tuo Ji, Xipeng Qiu, and Zheng Zhang. 2021a. [A unified generative framework for aspect-based sentiment analysis](#). 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 2416–2429, Online. Association for Computational Linguistics.
- Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021b. [A unified generative framework for various NER subtasks](#). 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 5808–5822, Online. Association for Computational Linguistics.
- Biao Zhang and Rico Sennrich. 2019. *Root mean square layer normalization*. Curran Associates Inc., Red Hook, NY, USA.
- Wenxuan Zhang, Yue Deng, Bing Liu, Sinno Pan, and Lidong Bing. 2024. [Sentiment analysis in the era of large language models: A reality check](#). In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3881–3906, Mexico City, Mexico. Association for Computational Linguistics.
- Wenxuan Zhang, Xin Li, Yang Deng, Lidong Bing, and Wai Lam. 2021. [Towards generative aspect-based sentiment analysis](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 504–510, Online. Association for Computational Linguistics.
- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2025. [A survey of large language models](#).

A Implementation Details

We train our models using NVIDIA L40 GPUs and the following key hyperparameters:

Parameter	Value
Learning rate	5e-5 with polynomial decay
Batch size	16
Optimizer	AdamW (weight decay 0.01)
Gradient clipping	5.0 (norm)
Training epochs	50
Sampling strategy	Greedy

For statistical robustness, we run each experiment with five different random seeds (1, 42, 123, 420, 1337) and report means and standard deviations. Model selection for testing is based on validation set F_1 -Score. We use the same tuple-level evaluation script as published by Yan et al. (2021a), where a prediction is considered correct only when all components match the ground truth exactly.

B GPT-4o Fine-tuning Prompt Template

For fine-tuning GPT-4o on the ASTE task, we employ a structured prompt template transforming tuple-based annotations into natural language. Given the sample:

*“The **drinks** are always **well made** and **wine selection** is **fairly priced**”*

The tuple annotation

(2, 2, 5, 6, POS, 8, 9, 11, 12, POS)

is transformed into:

```
{
  "messages": [
    {
      "role": "system",
      "content": "You are an AI trained to perform aspect-based sentiment analysis. Identify aspects in the given text and determine their associated sentiments and opinions."
    },
    {
      "role": "user",
      "content": "Analyze the following text for aspect-based sentiment: The drinks are always well made and wine selection is fairly priced Identify the aspects and their associated sentiments."
    },
    {
      "role": "assistant",
      "content": "Here's the aspect-based sentiment analysis:
      Aspect: drinks
      Sentiment: POS
      Opinion: well made

      Aspect: wine selection
      Sentiment: POS
      Opinion: fairly priced"
    }
  ]
}
```

This structure follows OpenAI’s fine-tuning specifications, providing context through the system message, input text through the user message, and expected output through the assistant message. All examples were automatically processed into this format. During inference, responses are parsed back into tuple format for consistent evaluation with the other models.

C Heatmap after Normalization

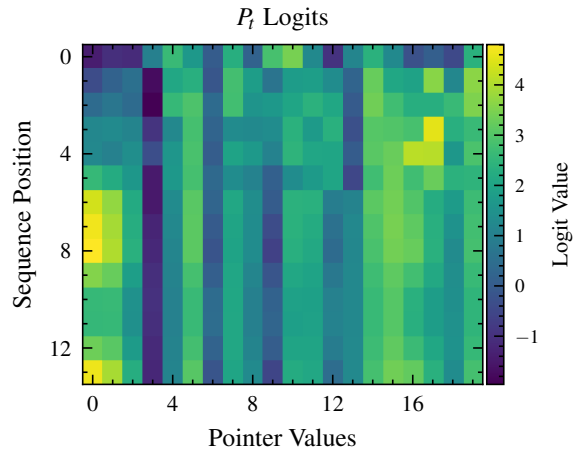


Figure 4: Heatmap of the untrained model’s pointer logit distribution (P_i) with the encoder normalization enabled (Lines 6 and 7 in Algorithm 1). The visualization shows no noticeable inherent initialization bias towards the lower pointer values corresponding to the special tokens, as was the case in Figure 2

D Generalization to other Tasks

Table 6: Performance comparison between BART-LARGE baseline and our enhanced BARTABSA++ on three additional structured prediction tasks.


Task	Our BARTABSA-R			BARTABSA++		
	P	R	F1	P	R	F1
SSA	57.51 (σ 2.72)	41.76 (σ 12.42)	47.38 (σ 10.07)	61.02 (σ 1.73)	51.51 (σ 3.45)	55.80 (σ 2.44)
SRE	20.25 (σ 11.41)	29.03 (σ 16.39)	23.85 (σ 13.45)	25.75 (σ 1.46)	35.99 (σ 1.55)	30.01 (σ 1.52)
DEFT	51.26 (σ 0.28)	37.86 (σ 5.12)	43.36 (σ 3.46)	54.20 (σ 3.94)	41.30 (σ 2.27)	46.73 (σ 1.96)

To demonstrate the generalization of our architectural improvements beyond ABSA tasks, we evaluate our enhanced model on three additional structured prediction tasks and report those results in Table 6. To model these tasks appropriately for pointer networks to solve, the output grammars for these tasks are natural extensions or modifications of the previously introduced ABSA output grammar (Section 2.2), adapted to the specific requirements of each task. This approach has been proven effective for related tasks across different languages (Yan et al., 2021b; Wunderle et al., 2024), and even in entirely different domains such as mathematical transformation identification (Wankerl et al., 2025).

Structured Sentiment Analysis (SSA) extends ABSA by extracting more comprehensive opinion structures, including opinion holders, targets, expressions, and sentiment polarities, with support for discontinuous spans. We use the combined English datasets from SemEval 2022 Task 10 (Barnes et al., 2022): OpeNER_{EN} (hotel reviews) and DS_{Unis} (university reviews).

Semantic Relation Extraction (SRE) identifies relations between entities in scientific abstracts, while also classifying them into six predefined relation types. We use the manually annotated ACL RD-TEC 2.0 dataset (QasemiZadeh and Schumann, 2016) from SemEval 2018 Task 7.

Definition Extraction from Free Text (DEFT) extracts definition terms and their corresponding definitions from naturally occurring text, requiring classification of both terms and definitions, as well as their relationship. We use the dataset from SemEval 2020 Task 6 (Spala et al., 2020), containing samples from open-source textbooks.

 **We find:** As shown in Table 6, our enhanced model consistently outperforms the baseline across all tasks, with particularly notable improvements in F₁ scores for SSA (+8.42) and SRE (+6.16), demonstrating generalization of our architectural changes to a diverse set of structured prediction tasks.

E Pre-Training Synthetic Encoder-Decoder Models

To verify that our findings about non-scaling decoder LLMs aren’t simply due to insufficient training of the newly initialized weights, we conduct additional pretraining experiments using the CNN/DailyMail summarization dataset (See et al., 2017; Hermann et al., 2015). This is the same dataset used by Rothe et al. (2020) in their original work on synthetic encoder-decoder models, making it particularly appropriate for our investigation. As the dataset contains over 300,000 news articles paired with human-written summaries, it provides substantial training data for adapting the models to the encoder-decoder paradigm.

We pretrain each GPT-2 model for 3 epochs on the summarization task, measuring Rouge2 scores throughout training. As shown in Table 7, the results reveal both training instability and a complete lack of scaling benefits. Most notably, GPT-

Table 7: Rouge2 scores for models pretrained on CNN-DailyMail summarization dataset for 3 epochs. The table shows both the best score achieved during training and the final score after 3 epochs.


Model	Best Rouge2	Final Rouge2
BART-BASE	19.65	19.65
ROBERTA2ROBERTA	19.35	19.35
GPT2GPT Base	14.60	2.32
GPT2GPT Medium	4.13	4.13
GPT2GPT Large	14.86	14.77
GPT2GPT XL	11.86	11.76

Table 8: Performance of different pretrained GPT-2 model sizes.

Model	Params	Avg		
		P	R	F ₁
Our BARTABSA-R	139M	68.91 (σ 2.13)	65.31 (σ 3.17)	67.01 (σ 2.44)
GPT2GPT Base	277M	60.64 (σ 2.38)	53.82 (σ 2.03)	56.98 (σ 1.95)
GPT2GPT Medium	810M	60.61 (σ 2.10)	54.39 (σ 1.94)	57.28 (σ 1.80)
GPT2GPT Large	1.78B	60.73 (σ 2.12)	54.04 (σ 1.85)	57.12 (σ 1.64)
GPT2GPT XL	3.61B	60.45 (σ 2.05)	53.90 (σ 2.10)	56.98 (σ 1.85)

2 Base exhibits dramatic performance degradation (from a best Rouge2 score of 14.60 to a final score of 2.32), while GPT-2 Medium performs consistently poorly. Even the best-performing GPT-2 Large only achieves a Rouge2 of 14.86, substantially below both BART-BASE (19.65) and RoBERTa2RoBERTa (19.35).

When initializing our pointer networks with these already pretrained checkpoints (Table 8), we observe very similar patterns to our non-pretrained experiments. The lack of scaling benefits persists, with performance plateauing or even declining as model size increased.

 **Confirmation:** These results further support our conclusion that encoder quality is the primary determinant of performance in encoder-decoder models, and that decoder-only models face fundamental limitations when adapted to serve as encoders, regardless of this additional pretraining.