SenPoi at SemEval-2022 Task 10: Point me to your Opinion, SenPoi

Jan Pfister
University of Würzburg

Sebastian Wankerl
University of Würzburg
DHBW Mosbach

Andreas Hotho
University of Würzburg

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

Abstract

Structured Sentiment Analysis is the task of extracting sentiment tuples in a graph structure commonly from review texts. We adapt the Aspect-Based Sentiment Analysis pointer network BARTABSA to model this tuple extraction as a sequence prediction task and extend their output grammar to account for the increased complexity of Structured Sentiment Analysis. To predict structured sentiment tuples in languages other than English we swap BART for a multilingual mT5 and introduce a novel Output Length Regularization to mitigate overfitting to common target sequence lengths, thereby improving the performance of the model by up to 70%. We evaluate our approach on seven datasets in five languages including a zero shot crosslingual setting.

1 Introduction

The goal of sentiment analysis is to understand a writer's opinions expressed in a text. In recent years, this topic became of particular research interest since with the advent of social media platforms, users were encouraged to share their opinions about a wide range of subjects with the world. For example, websites like Yelp collect and share opinions about restaurant visits and various online retailers allow customers to publish their reviews about items in their assortment. Consequently, various text corpora emerged that made it feasible to apply machine learning based approaches to this task (Yadav and Vishwakarma, 2020).

Sentiment analysis can be approached with various degrees of granularity. It is possible to only classify the overall polarity of a sentence as positive or negative, or a more precise intermediate value (Nguyen et al., 2020; Devlin et al., 2019).

Furthermore, the sentiment can be predicted in a more fine-grained manner, like with regard to the various targets addressed in the sentence. This can be of particular interest if the writer is ambiguous in their review and no overall sentiment label can be assigned to a sentence. For example, a customer might describe the service or the interior of a restaurant as unpleasant but the food itself as good. This can be extended to more complex sentence structures, taking into account not only the aspects or the sentiment, but also its holder (i.e. who is expressing the sentiment) or the opinion term itself, e.g. (Yan et al., 2021; Mukherjee et al., 2021).

In this paper, we work on the challenge described by Barnes et al. (2022). Here, the goal is to extract an arbitrary number of opinion tuples from a text. Each tuple can consist of a *holder*, a *target*, a *sentiment expression*, and a *polarity*. However, it is also possible that some of these mentioned *entities* are not present in a certain tuple. Moreover, a sentence may include no sentiment and hence no tuples should be extracted.

The challenge spans data from five different languages, namely English, Spanish, Catalan, Basque, and Norwegian and two different subtasks. First, there is the monolingual setting in which you are allowed to train on all sentiment data, including those in the language that is subsequently used to test the model's capability of extracting sentiment. Moreover, in the crosslingual setting the models are tested on Spanish, Catalan and Basque sentiment data while being trained on any of the other languages. It was also allowed to train on Spanish, Catalan and Basque data as long as it does not contain any annotated sentiment information. This makes the crosslingual structured sentiment analysis task a zero shot setting.

Our approach is based on BARTABSA (Yan et al., 2021), who leverage a *pointer network* to predict sentiment tuples (aspect, opinion, sentiment) by representing them using a custom output grammar. They extend a BART model with a pointer generator network which is thus able to predict token *indices* from the input sequence as output.

Our contributions are as follows: (i) we introduce a new, flexible grammar to model structured sentiment graphs as a pointer sequence, (ii) we explore mT5 as base model allowing us to make cross-lingual predictions, (iii) we introduce a new length regularizer to prevent overfitting to common sequence lengths.

2 Preliminaries

2.1 BART & mT5

BART (Lewis et al., 2020) is a sequence-to-sequence model built using transformer (Vaswani et al., 2017) neural networks. In its default configuration BART consists of 6 encoder and 6 decoder transformer layers. BART is trained as a denoising autoencoder. Hence, the input to its encoder are sentences which are noised using five different methods like masking or permutation of tokens. Consequently, the decoder is trained to restore the original sentence like defined in Equation 1.

mT5 (Xue et al., 2021; Raffel et al., 2020) is another sequence-to-sequence model based on the transformer architecture. It is trained on 101 languages at the same time using the span corruption objective and at time of publication achieved state of the art on many multi-lingual benchmarks. Its training corpus includes all languages used in this challenge (section 1).

2.2 BARTABSA

We model the Structured-Sentiment Analysis task as a seq2seq-task by adopting the framework introduced in BARTABSA. It encodes sentiment tuples (section 1) as a sequence of token indices (pointers) and special tokens.

2.2.1 Output Grammar

In BARTABSA, the target sequence consists of tuples with a fixed size of five:

$$\ldots, a_i^s, a_i^e, o_i^s, o_i^e, s_i^p, \ldots$$

which allows them to express the i^{th} aspect term (a) and its opinion term (o) in an input sentence by their respective start^s and end^e integer *token-index* in the input token sequence, as well as the sentiment polarity class s^p associated with this combination of aspect and opinion terms. Sequences structured like this can be unambiguously converted to aspect-based sentiment tuples as they occur in fixed sequence lengths of five: four token indices and one sentiment class.

2.2.2 Model

Sequences of this *output grammar* can be predicted by the model by *pointing* to the tokens in the input sequence in this fixed order and finally completing the triplet of aspect, opinion and sentiment by predicting the associated sentiment class.

Thus, the task is modelled as an auto-regressive decoding task where the probability of the next output token $P(Y|X)$ depends on both the input X as well as the previously decoded tokens $Y_{<t}$. For a sequence of length m the decoding process is hence given as

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

Usually, the decoding is stopped as soon as the End-Of-Sequence token `EOS` is predicted.

An architecture to generate such output was introduced as BARTABSA (Yan et al., 2021). BARTABSA is based on BART (Lewis et al., 2020) and uses a pretrained BART encoder to encode the input sentence. Hence, given an input sequence of n tokens $s = [x_1, \ldots, x_n]$, BARTABSA first applies the encoder to obtain the input representation

$$H_e = \text{BARTEncoder}(s) \tag{2}$$

with $H_e \in \mathbf{R}^{n \times d}$ (d denotes the embedding size).

Since the goal of BARTABSA is to predict indices to tokens in the input sequence, its decoder is augmented with a pointing mechanism (Vinyals et al., 2015) to generate these indices (Yan et al., 2021) instead of tokens from a vocabulary (see Equation 5 – 8). To be able to predict l additional sentiment class labels C, they are concatenated to the input and treated as tokens of the input sentence. However, to train and inference the neural network in an auto-regressive manner, i.e. feeding the previous output back into the decoder, the yet generated pointers must be mapped back to their indexed tokens first using the following mapping:

$$\hat{y}_t = \begin{cases} X_{y_t} & \text{if } y_t \in X \\ C_{y_t - n} & \text{if } y_t \text{ is an index of a class label} \end{cases} \tag{3}$$

The BART decoder is then applied to the mapped pointers:

$$h_t^d = \text{BARTDecoder}(H^e; \hat{Y}_{<t}) \tag{4}$$

Using h_t^d, the distribution for pointing to the input sequence is then calculated as:

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

$$C^d = \text{BARTTokenEmbed}(\text{C}) \tag{6}$$

BARTTokenEmbed is a shared embedding layer which is used to embed both the tokens from the input sequence X as well as the class labels C.

$$\bar{H}^e = \alpha \ \text{MLP}(H^e) + (1 - \alpha)E^e \tag{7}$$

Finally, the embedded input sequence is concatenated with the embedded class labels. The pointing distribution is then given as the softmax over the concantenated sequence multiplied with the hidden representation h_t^d obtained from the BART decoder:

$$\begin{aligned} P_t &= \text{softmax}([\bar{H}^e; C^d]h_t^d) \\ &= P(y_t|X, Y_{<t}) \end{aligned} \tag{8}$$

where $E^e, H^e, \bar{H}^e \in \mathbb{R}^{n \times d}$, $C^d \in \mathbb{R}^{l \times d}$, $P_t \in \mathbb{R}^{n+l}$. Teacher forcing (Williams and Zipser, 1989) is used during training together with negative log-likelihood as optimization criterion.

3 Methodology

Our approach to predicting sentiment graphs on the given datasets is grounded on BARTABSA (2.2). We adopt and extend their framework to not only be able to predict aspect-based sentiment but also structured sentiment. For this we introduce a new output grammar which is able to model and represent the sentiment tuples as required for this task.

We adapt their sequential pointer representation for *Triplet Extraction* (aspect, opinion, sentiment) to the task at hand.

3.1 Extensions to the Output Grammar

We extend the expressive power of their output grammar in several ways to account for the increased target complexity of the structured sentiment task: Each entity (can be a or o for Triplet Extraction) for structured sentiment analysis can...

- be optional
- consist of arbitrarily many discontiguous parts
- be “source”/“holder” of the sentiment

Defining such an enhanced output grammar which can model these properties enables us to unambiguously represent the sentiment tuples required for the structured sentiment task. In the following we adapt the notion of BARTABSA from “aspect term” to the sentiment *target* and “opinion term” to the polar *expression*.

3.1.1 Entity Absence

First we account for *optional absence* of entities by introducing a special token for each type of entity indicating the begin of its respective entity. This prevents ambiguity if an entity is absent and transforms the previous example from subsubsection 2.2.1 to the following sequence:

$$\ldots, \text{TGT}_{BEG}, t_i^s, t_i^e, \text{EXP}_{BEG}, e_i^s, e_i^e, s_i^p, \ldots$$

where XYZ_{BEG} is the unique class token for each entity type indicating the begin of entity XYZ, t_i and e_i are *integers* referring to token positions in the input sequence and s_i^p is the sentiment class token (see Figure 1 for example). This way if, e.g., there is no target term to be predicted for the i^{th} sentiment tuple the sequence becomes

$$\ldots, \text{EXP}_{BEG}, e_i^s, e_i^e, s_i^p, \ldots$$

and thereby stays unambiguous as it is still well defined that the tokens between e_i^s and e_i^e resemble the expression term and not the missing target term. This can be clearly interpreted although e_i^s and e_i^e are the first two predicted indices of the i^{th} sentiment tuple which were allocated to target entity indices before.

3.1.2 Entity Splitting

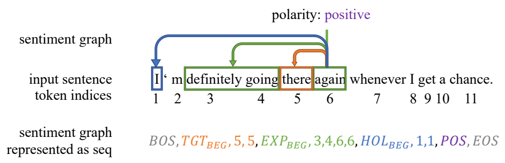
Second we allow for *split entities* as described in subsection 4.3 by further extending the output grammar. After every begin-of-entity special token (XYZ_{BEG}) we do not only allow for a single start^s and end^e index tuple but arbitrarily many such tuples. Therefore a two-part target term in the i^{th} triplet is represented like this:

$$\ldots, \text{TGT}_{BEG}, t_i^{s1}, t_i^{e1}, t_i^{s2}, t_i^{e2}, \text{EXP}_{BEG}, e_i^s, \ldots$$

In this syntax the first part of the i^{th} target term can be found between t_i^{s1} and t_i^{e1} and the second part between t_i^{s2} and t_i^{e2} (see Figure 1 for example). Every entity can be modeled by arbitrarily many such pointer tuples and due to the special tokens (XYZ_{BEG}) indicating transitions between the predicted entities the association between pointer indices and entity type stays unambiguous.

3.1.3 Sentiment Source

Last we enable the prediction of the *sentiment holder* or source. Given the previous modifications to the output grammar this can be easily achieved by introducing a new special token HOL_{BEG}.



polarity: positive
sentiment graph
input sentence
token indices
I ‘ m definitely going there again whenever I get a chance.
1 2 3 4 5 6 7 8 9 10 11
sentiment graph
represented as seq
$BOS, TGT_{BEG}, 5, 5, EXP_{BEG}, 3,4,6,6, HOL_{BEG}, 1,1, POS, EOS$

Figure 1: Exemplary input sentence, its associated sentiment graph (target, expression, holder, sentiment) as well as the sentiment graph modeled using our grammar.

3.1.4 Constructing the Output Grammar

Combining all our extensions to the output grammar introduced above we are able to unambiguously represent structured sentiment tuples in a sentence as a sequence consisting of pointers and special tokens.

First we sort all sentiment tuples per sentence ascending by their token index in the input sentence. For this we only look at the index of the start token of each entity. We first sort the tuples by the start index of the target term, if they are equal by their expression term and lastly on equality we fall back to their holder term. This is done the same way in BARTABSA and PASTE as it is shown that predicting the sentiment tuples of a sentence in a strict order boosts the performance of the final model (Yan et al., 2021; Mukherjee et al., 2021).

A sentiment graph for an input sentence is modelled as follows using our output grammar (example in Figure 1): After the BOS-token we encode the sentiment tuples in the order described above. First if the i^{th} sentiment tuple contains an target term we insert an TGT_{BEG}-token followed by a sequence of tuples of start (t_i^s) and end (t_i^e) indices, one for every discontiguous part of the target term. We repeat this for the expression and holder terms of the sentiment tuple. The representation of the i^{th} sentiment tuple is completed by its sentiment polarity token (pos, neu, neg). This process is repeated for every sentiment tuple of the input sentence. After the last sentiment polarity token the output sequence is closed with an EOS-token.

If a sentence has no sentiment tuples it is represented by the empty sequence $[BOS, EOS]$.

3.2 Output Length Regularizer

Predicting the structured sentiment graph for an input sentence using our above (3.1.4) defined output grammar is naturally very sensitive to the placement of the EOS-token. If our model predicts the EOS-token e.g. only a single token too early the last sentiment tuple — consisting of target, expression, holder and sentiment class — becomes incomplete as it lacks at least the sentiment class for the entire tuple. Therefore it can no longer be correctly interpreted or converted to the sentiment graph structure. A missplaced EOS-token can lead us to miss an entire sentiment tuple or even more if the EOS-token is off by more than the length of a tuple.

During our experiments we noticed some models being prone to overfitting to common positions of the EOS-token (also see Newman et al. (2020)) in the train set (5.1). When analysing the predictions of such a model during training we found that often the EOS-token was predicted not only at the correct location in the sequence, but also once more, earlier at the most-common EOS-token position in the train set. Usually this is right after the BOS-token. As we convert the output sequence to the sentiment graph up to the first EOS-token (3.1.4) this results in an empty graph. Such an incorrect sequence consisting of an otherwise correctly predicted sentiment unit with a target and an expression exemplarily looks like this:

$$[BOS, EOS], t_i^s, t_i^o, \mathrm{EXP}_{BEG}, e_i^s, e_i^e, s_i^p, EOS$$

In this case the TGT_{BEG}-token got erroneously replaced by an overfitted EOS-token, thereby stopping the output sequence early — as indicated by the gray font. As can be clearly deducted from Figure 1 such a token replacement would result in an empty sequence and thereby no extracted sentiment graph. We cannot just ignore/fix such a misplaced EOS-token automatically because we cannot decide whether such an error occurred at all or which begin-of-entity (XYZ_{BEG}) got replaced by the first EOS.

Therefore we have to prevent these errors from occurring in the first place. To encourage the model to predict the EOS-token at the correct location only, we extended the loss function to make it more sensitive to the placement of the EOS-token.

We realize this by introducing an additional loss component $\mathcal{L}_{\mathcal{RE}}$. The predictions $\hat{y}$ of our model during training have the shape $\mathbb{R}^{s \times \hat{t}}$, where $\hat{t}$ and s is the length of the prediction $\mathcal{P}$ and source sequence $\mathcal{S}$ respectively, as we predict t *target* indices pointing to tokens in the source sequence of length s (including special tokens).

Now, along every column $j \in 1 \ldots \hat{t}$ of $\hat{y}$ we calculate the softmax to decide the token probability distribution at prediction step j. From this matrix we now extract the row $o = \hat{y}_i \in \mathbb{R}^{\hat{t}}$, $i \in 1 \dots s$ corresponding to the EOS-token. Then, $\text{softmax}(o)$ at position j represents the probability that the j-th token in the target sequence is an EOS-token over all positions in the target sequence.

We calculate the cross entropy between this resulting vector o and the correct position for the EOS-token in the target sequence.This loss addition penalizes high-probabilities for EOS-tokens in the wrong locations:

$$\mathcal{L}_{\mathcal{RE}}(y, \hat{y}) = \text{CE}(t, \text{softmax}(\hat{y}_i)) \qquad (9)$$

where t is the length of the target sequence y — thereby the correct position of the EOS-token — and i is the vocabulary index of EOS.

We add our Output Length Regularization to the Cross Entropy loss:

$$\mathcal{L}(y, \hat{y}) = \text{CE}(y, \hat{y}) + \mathcal{L}_{\mathcal{RE}}(y, \hat{y}) \qquad (10)$$

4 Experiments

We base our model on two different but related architectures depending on the subtask. For both tasks, we build a model similar to BARTABSA (Yan et al., 2021)[1] consisting of a transformer encoder-decoder model augmented with a pointer layer for predicting indices in the input sequence.

We always train for 75 epochs and select the best model based on the best performance on the validation set. This strategy is in line with BARTABSA but we increased the maximum number of epochs from 50 to 75 as we found in preliminary experiments that mT5 sometimes was able to improve slightly on the validation set beyond 50 epochs. If we train on multiple datasets at the same time we also validate on a concatenation of their respective validation sets. We finetune the models once with and once without the Output Length Regularization (3.2). We select model dependent learning rates, learning rate schedules and optimizers as suggested in their respective original papers or BARTABSA. Therefore we finetune BART models using Adam with a peak learningrate of $5e^{-5}$ in a triangular learning rate schedule and mT5 using AdaFactor with a constant learningrate schedule and a learningrate of 0.001.

The results are evaluated using the Sentiment Graph F_1 introduced by Barnes et al. (2021). For this metric they calculate the true positives by averaging the overlap for exact token-level matches between predicted and gold spans over all three entities. To obtain precision and recall they now divide the number of correctly predicted tokens by the total number of predicted tokens and gold tokens respectively.

4.1 Subtask 1: Monolingual

For the monolingual task we divide between English and non-English datasets. For the English datasets we select a pretrained monolingual English BART model as our transformer architecture as suggested in BARTABSA. As there are no BART models publicly available for all non-English languages from our datasets, we choose a pretrained mT5 model for those instead which was pretrained on 101 languages including all languages present in our datasets. We do not train a separate mT5 for every single dataset or language since finetuning mT5 on small datasets lead to significant instabilities during training (5.1) which we counteract by concatenating all datasets available to us.

We compare our approach against the baselines provided by the task organizers[2]. They provide a sequence labelling approach which consists of three separate BiLSTM models, one each for extracting the holders, targets, and expressions followed by another BiLSTM-based relation prediction model. On top of the concatenation of these three outputs a classification task is trained to predict whether two predicted elements are related or not. The second baseline is a graph parsing model described by Barnes et al. (2021). It works by modelling the sentiment tuples as dependency graphs and then predicting those using the neural dependency parser introduced by Dozat and Manning (2018).

4.2 Subtask 2: Crosslingual

In the crosslingual setting it is important for our model to be able to generalize and transfer well between languages. Therefore we once again select mT5 as our basemodel and train it on English and Norwegian at the same time as these are the only languages available for training in this setting. We train on both available languages at the same time since we expect this to improve generalization between languages.

[1]For our experiments we reuse their codebase making use of torch, transformers and fastNLP.

[2]https://github.com/jerbarnes/semeval22_structured_sentiment

4.3 Datasets

The datasets provided for this challenge can be split into five languages: English (darmstadt (Toprak et al., 2010), mpqa (Wiebe et al., 2005), opener (Agerri et al., 2013)), Spanish (opener), Catalan (multibooked (Barnes et al., 2018)), Basque (multibooked) and Norwegian (norec (Øvrelid et al., 2020)). All datasets are structured identically consisting of a source sentence and its annotated sentiment graphs, which are represented by their sentiment tuples (target, expression, holder, polarity) as introduced in section 1. Additional complexity is added by the fact that every sentiment tuple entity can be either completely missing or even consist of arbitrarily many discontiguous parts as described in subsubsection 3.1.4: e.g. in "[...] have degraded the image of the University severely" the opinion term consists of two parts: "degraded" and "severly".

4.4 Dataset Sampling

During preliminary experiments we observed stability problems while finetuning mT5 (5.1) which we were only able to mitigate by training on the concatenation of different datasets and sampling the training dataset. We were able to address this instability by undersampling the training set in such a way that *at most* one fifth of the training set consists of empty samples. We implemented this sampling by first ensuring this ratio for all datasets separately and concatenating them afterwards. We kept this sampling strategy for all experiments using mT5 except when training only on the concatenation of the English datasets where we found that a maximum of one fourth empty samples achieved better results on the validation set.

We ran the same experiments with BART and found it to be not nearly as susceptible to a sampling strategy compared to mT5. Nevertheless we found that BART is able to gain small improvements on the validation set only when training on the concatenation of all English datasets without Output Length Regulatization while sampling the training datasets such that at most half of the samples are empty. So we sampled the training dataset only in this specific case when finetuning BART.

5 Results & Evaluation

5.1 Training Instability of mT5

While finetuning mT5 on our datasets during preliminary runs we noticed that the model is not only generally very unstable during training but also that the resulting performance is very sensitive to different dataset splits. When training on individual training datasets including all samples the model was not able to achieve sentiment-F_1-scores above 5% on the validation set. For this it did not matter whether we included the Output Length Regularization (3.2) or not. We only managed to achieve competitive results using mT5 after we significantly undersampled the empty sentiment tuples in the dataset as described in subsection 4.4. We believe this to be already first signs of EOS-token location overfitting (3.2) as all empty samples are represented by the sequence $[BOS, EOS]$.

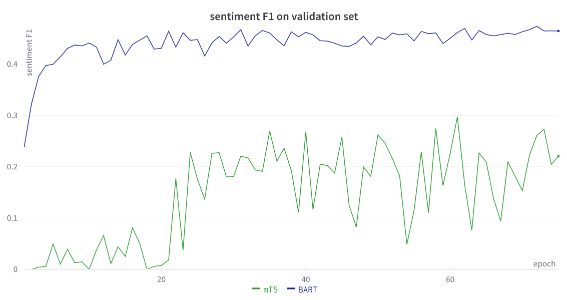


Figure 2: Comparison of two representative English validation set sentiment-F_1 curves for BART and mT5 during training. The models got evaluated in each of the 75 training epochs.

Training instabilities of mT5 become evident when looking at the F_1-scores on the validation set during training as representatively plotted in Figure 2 (green). For comparison we rerun the same setup using BART (blue). While the training loss of the mT5-models descends during training without large jumps or spikes the F_1-score on the validation set oscillates heavily. Although all mT5-models we trained contained such frequent and huge jumps in the F_1-score, there became no common pattern apparent between different runs. Even the slightest change e.g. different dataset sampling can result in a completely different F_1-score curve on the validation set. At the same time the loss during training on the same dataset remained smooth. This is not surprising given the fact that a wrongly placed EOS-token can invalidate the entire prediction as described in subsection 3.2, while in comparison according to the cross entropy loss a higher probability for the EOS-token at the e.g. second position only slightly decreases the measured performance. In other words the final evaluation metric is way more sensitive to the placement of the EOS-token

Table 1: Submitted results for the monolingual subtask as described in subsection 4.1.

		ours		baselines	
dataset	model	sent-F_1	place	seq. label	graph parser
opener_en	BART	0.651	12	0.33	0.52
mpqa	BART	0.338	11	0.02	0.12
ds_uni	BART	0.417	6	0.06	0.20
average		0.469	—	0.14	0.28
opener_es	mT5	0.504	17	0.24	0.50
norec	mT5	0.280	18	0.20	0.36
multib_ca	mT5	0.517	16	0.34	0.52
multib_eu	mT5	0.439	18	0.37	0.55
average		0.435	—	0.29	0.64
overall avg		0.449	16	0.22	0.40

than the training objective. This is the core issue we are addressing by introducing the additional loss component (3.2). We did not observe a similar phenomenon during training of any BART model.

5.2 Subtask 1: Monolingual

As we approached the monolingual setting using two different base-models we also analyse them separately. Overall for this monolingual subtask we placed 16^{th} out of 31 teams on the leaderboard at the time the challenge ended.

For mT5 we report the results for finetuning on all datasets on all languages at the same time using our Output Length Regularization, as training mT5 became slightly more stable on a larger dataset.

For the English results we finetune BART with our Output Length Regularization only on the respective dataset the model was tested on except when testing on the darmstadt dataset. For darmstadt we trained BART on a combination of all English datasets and without Output Length Regularization. If we train only on the darmstadt dataset itself and with Output Length Regularization like for the other datasets, we achieve an F_1-score of only 0.389. We chose this strategy for all BART runs in general and the darmstadt dataset in specific as it resulted in the highest scores on the respective validation datasets. The results for the other datasets when training BART on all english datasets at the same time is included in Table 2.

In Table 1 we report our performance on the monolingual task and compare it against the baselines provided by the task organizers. On English datasets using BART we achieved in general a considerably higher placement (6,11,12) than our over all placement (16) and are able to comfortably beat the employed baselines (subsection 4.1). This indicates that our method works comparatively better using BART as base model than mT5 (16, 17, 18, 18) where our approach consistently beats the sequence labelling approach but matches the performance of the graph parser only on some datasets.

Table 2: Comparison of a single BART and a single mT5 trained on all English datasets at the same time.

	sent-F_1	
dataset	BART	mT5
opener_en	0.493	0.471
mpqa	0.326	0.159
darmstadt_uni	0.365	0.218
average	0.395	0.283

5.2.1 BART vs. mT5 Performance

To further evaluate this presumed performance discrepancy between BART and mT5 we compare our approach using both base models on the English datasets as this is the only language both models have in common. We train both models using the Output Length Regularization and only differing by their respective optimal sampling strategy as laid out in subsection 4.4. For a fair comparison we train both models on all datasets at the same time as a larger dataset reduces training instabilities of mT5 (5.1). The results are visible in Table 2.

We find the performance to be drastically dropping overall when we switch from a BART to an mT5 model. The overall validation loss for both models in every epoch is reported in Figure 2. We assume this drop is hugely driven by the training instabilities (5.1) we observed, although we also suspect the differences in pretraining (2.1) to lead to this discrepancy. This explains our comparatively better placement in Table 1 when we are able to use BART instead of mT5 to solve the task.

5.2.2 Ablation: Output Length Regularization

In order to evaluate how well our novel Output Length Regularization (3.2) is able to improve model performance we finetune a separate BART for every English dataset once with and once with-

Table 3: Comparison of performance for models trained with and without length regularization. BART models were trained on the dataset they are tested on while mT5 models were trained on all languages at the same time.

		sent-F_1	
model	dataset	w/ lenreg	w/o lenreg
BART	opener_en	0.651	0.635
	mpqa	0.338	0.320
	darmstadt_uni	0.389	0.320
	average	0.459	0.425
mT5	opener_es	0.504	0.410
	norec	0.280	0.251
	multibooked_ca	0.517	0.374
	multibooked_eu	0.439	0.353
	average	0.435	0.347
	overall average	0.449	0.367

Table 4: Average number of predicted sentiment tuples per sentence compared to actual average number of sentiment tuples per sentence in the testset.

		lenreg		
	dataset	w/	w/o	testset
BART	opener_en	1.77	1.88	1.73
	mpqa	0.21	0.29	0.24
	darmstadt_uni	0.31	0.47	0.41
mT5	opener_es	2.02	1.49	2.33
	norec	1.28	0.77	0.97
	multibooked_ca	1.39	1.00	1.56
	multibooked_eu	1.19	0.76	1.43

out length regularization. We repeat this setup for mT5 but train a single common mT5 on all datasets together as we found mT5 to be more stable during training with increasing dataset sizes.

We compare the results in Table 3. It is apparent that for all runs for both base models on all datasets and languages the sentiment F_1 score improves when adding our length regularization. Therefore we conclude that our Output Length Regularization (3.2) does indeed help the model learn where to place the EOS-token and thereby decide how many sentiment tuples are present in the sentence. This results in better predictions for this task especially for the mT5 model.

Originally we introduced the Output Length Regularization to fix an overfitted EOS-token at the second position for the mT5 model as exemplarily indicated by the colors in subsection 3.2. This led the mT5 model to predict too few (commonly zero) sentiment tuples. When analysing the differences in length of the output sequences (see Table 4) we found that for the mT5 model it consistently increased the number of sentiment tuples predicted by the model and thereby almost always moves the average number of predicted tuples per sentence closer to the average number of sentiment tuples present in the dataset splits. Only for norec the average number of predicted tuples on the testset overshoots the average number of sentiment tuples on the testset.

Table 5: Performance of BART per dataset (columns) when finetuning only on a single dataset (rows). All models trained with Output Length Regularization.

	op_en	mpqa	ds_uni
opener_en	0.651	0.007	0.195
mpqa	0.008	0.338	0.050
darmstadt_uni	0.300	0.005	0.389
all English	0.493	0.326	0.365

5.2.3 Crossdomain Performance

The crosslingual subtask of this challenge primarily evaluates how well a model is able to generalize between different languages and different domains at the same time. We also analyze the performance of our BART model when we change only the target domain by crossdomain zero shot evaluating on a different english dataset. Therefore we finetune a separate BART model with Output Length Regularization on each of the three English datasets separately and then use each model to predict all other English datasets.

In Table 5 we find BART to be able to generalize between darmstadt_uni and opener_en albeit this comes at the cost of a significant performance loss in both directions. Meanwhile mpqa does not seem to be similar enough to either of the other two English datasets for the model to output meaningful crossdomain predictions. This can be explained as both darmstadt_uni and opener_en datasets consist of reviews of universities and hotels respectively, while mpqa is a dataset focused around political opinion expression. It is likely that the model benefits from the more similar phrasing used in both review datasets compared to political opinions.

Table 6: Submitted results for the crosslingual subtask as described in subsection 4.2. We finetuned mT5 with Output Length Regularization on all datasets at once.

dataset	sent-F_1	place
opener_es	0.315	14
multibooked_ca	0.259	15
multibooked_eu	0.243	14
average	0.272	15

Table 7: Comparison of an mT5 trained with and without Output Length Regularization (3.2).

	sent-F_1	
dataset	w/ lenreg	w/o lenreg
opener_es	0.315	0.245
multibooked_ca	0.259	0.105
multibooked_eu	0.243	0.131
average	0.272	0.160

5.3 Subtask 2: Crosslingual

For the crosslingual task (Table 6) we predict structured sentiment tuples in Spanish, Catalan and Basque without prior training on any sentiment annotations in any of these languages. We again finetune an mT5, but for this subtask on all languages at once which are not in the set of target languages: English & Norwegian. Training on a larger dataset consisting of multiple languages not only stabilizes training but also possibly helps the model to generalize between languages to predict structured sentiment in unseen languages. Compared to the monolingual subtask where mT5 was trained using sentiment annotations in the target languages (Table 1), here mT5 loses on average over 40% of performance in this zero shot crosslingual setting. Therefore we are able to show that despite a significant performance loss pointer prediction models are able to zero shot generalize between languages and domains at the same time.

5.3.1 Ablation: Output Length Regularization

Again we evaluate our Output Length Regularization by finetuning two mT5 on all English and Norwegian datasets at the same time, once with and once without Output Length Regularization. We find the same results as already described in 5.2.2: Output Length Regularization is able to improve the zero shot crosslingual generalization significantly as can be found in Table 7. Finetuning mT5 for pointer prediction using this length regularization increases the performance by 70% averaged over all target datasets.

6 Related Works

The PASTE framework (Mukherjee et al., 2021) also uses pointer networks to solve the task of aspect-based sentiment analysis. Instead of Transformers, they use an LSTM (Hochreiter and Schmidhuber, 1997) as decoder and two additional Bi-LSTMs as pointer networks — one for pointing to aspect and opinion term each.

Peng et al. (2020) propose an approach to extract opinion triplets from text in a generative manner without using pointers. However, their output grammar is also less flexible meaning that all of the entities (target, aspect, and sentiment) have to be present in each triple. They also introduced one of the data sets used to train BARTABSA and PASTE.

Apart from structured sentiment analysis, pointer networks were also successfully applied to various further NLP tasks where it is beneficial to directly transfer parts of the input sequence to the output. This includes automatic summarization (See et al., 2017; Enarvi et al., 2020) and entity extraction (Nayak and Ng, 2020).

7 Conclusion

In this work, we adapted BARTABSA, a pointer network based on BART, for the task of structured sentiment analysis. In particular, we introduced a new output grammar which is able to model the increased complexity of this task by taking into account new entity types, split entities and missing entities in sentiment tuples.

We also experimented replacing BART with an mT5 network to allow for input sequences in languages other than English. We found that using the approach of BARTABSA it is possible to swap out BART for another base model but in the case of mT5 this comes with a significant performance hit which we suspect is mainly driven by training instabilities we encountered. Moreover, we introduced a output length regularizer to reduce overfitting to common sequence output lengths from the trainset. We found this to be very beneficial consistently on all data sets and to increase relative performance by up to 70%.

Acknowledgements

This research project is funded by the Bavarian Research Institute for Digital Transformation (bidt), an institute of the Bavarian Academy of Sciences and Humanities. The author is responsible for the content of this publication.

References

Rodrigo Agerri, Montse Cuadros, Sean Gaines, and German Rigau. 2013. OpeNER: Open polarity enhanced named entity recognition. In *Sociedad Española para el Procesamiento del Lenguaje Natural*, volume 51, pages 215–218.

Jeremy Barnes, Toni Badia, and Patrik Lambert. 2018. MultiBooked: A corpus of Basque and Catalan hotel reviews annotated for aspect-level sentiment classification. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Jeremy Barnes, Robin Kurtz, Stephan Oepen, Lilja Øvrelid, and Erik Velldal. 2021. Structured sentiment analysis as dependency graph parsing. 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 3387–3402, Online. Association for Computational Linguistics.

Jeremy Barnes, Andrey Kutuzov, Laura Ana Maria Oberländer, Enrica Troiano, Jan Buchmann, Rodrigo Agerri, Lilja Øvrelid, Erik Velldal, and Stephan Oepen. 2022. SemEval-2022 task 10: Structured sentiment analysis. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, Seattle. Association for Computational Linguistics.

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

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Seppo Enarvi, Marilisa Amoia, Miguel Del-Agua Teba, Brian Delaney, Frank Diehl, Stefan Hahn, Kristina Harris, Liam McGrath, Yue Pan, Joel Pinto, Luca Rubini, Miguel Ruiz, Gagandeep Singh, Fabian Stemmer, Weiyi Sun, Paul Vozila, Thomas Lin, and Ranjani Ramamurthy. 2020. Generating medical reports from patient-doctor conversations using sequence-to-sequence models. In *Proceedings of the First Workshop on Natural Language Processing for Medical Conversations*, pages 22–30, Online. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

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.

Rajdeep Mukherjee, Tapas Nayak, Yash Butala, Sourangshu Bhattacharya, and Pawan Goyal. 2021. PASTE: A tagging-free decoding framework using pointer networks for aspect sentiment triplet extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9279–9291, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tapas Nayak and Hwee Tou Ng. 2020. Effective modeling of encoder-decoder architecture for joint entity and relation extraction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8528–8535.

Benjamin Newman, John Hewitt, Percy Liang, and Christopher D. Manning. 2020. The EOS decision and length extrapolation. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 276–291, Online. Association for Computational Linguistics.

Xuan-Phi Nguyen, Shafiq Joty, Steven Hoi, and Richard Socher. 2020. Tree-structured attention with hierarchical accumulation. In *International Conference on Learning Representations*.

Lilja Øvrelid, Petter Mæhlum, Jeremy Barnes, and Erik Velldal. 2020. A fine-grained sentiment dataset for Norwegian. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5025–5033, Marseille, France. European Language Resources Association.

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.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

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.

Cigdem Toprak, Niklas Jakob, and Iryna Gurevych. 2010. Sentence and expression level annotation of opinions in user-generated discourse. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 575–584, Uppsala, Sweden. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz 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.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2):165–210.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*, 1(2):270–280.

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.

Ashima Yadav and Dinesh Kumar Vishwakarma. 2020. Sentiment analysis using deep learning architectures: a review. *Artificial Intelligence Review*, 53(6):4335–4385.

Hang Yan, Junqi Dai, Tuo Ji, Xipeng Qiu, and Zheng Zhang. 2021. 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.