AMEX AI Labs at SemEval-2022 Task 10: Contextualized fine-tuning of BERT for Structured Sentiment Analysis

Pratyush Sarangi, Shamika Ganesan, Piyush Arora, Salil Rajeev Joshi
American Express AI Labs, Bangalore, India
{pratyush.k.sarangi|shamika.ganesan|piyush.arora1|salilrajeev.joshi}@aexp.com

Abstract

We describe the work carried out by AMEX AI Labs on the *structured sentiment analysis* task at SemEval-2022. This task focuses on extracting fine grained information w.r.t. to source, target and polar expressions in a given text. We propose a BERT based encoder, which utilizes a novel concatenation mechanism for combining syntactic and pretrained embeddings with BERT embeddings. Our system achieved an average rank of 14/32 systems, based on the average scores across seven datasets for five languages provided for the monolingual task. The proposed BERT based approaches outperformed BiLSTM based approaches used for structured sentiment extraction problem. We provide an in-depth analysis based on our post submission analysis.

1 Introduction

In this paper we present the work done by AMEX AI Labs on the *structured sentiment analysis* monolingual task at SemEval-2022 (Barnes et al., 2022). Structured sentiment analysis (SSA) focuses on performing fine grained analysis and extracting opinion tuples from a given input text (Barnes et al., 2021). An example is presented in Fig. 1.

Input sentence: *Some others give the new UMUC 5 stars - don't believe them*

Expected outputs are two tuples:

- **Tuple1** - ("source": *"Some others"*, "target": *"the new UMUC"*, "polar expression": *"5 stars"*)
- **Tuple2** - ("source": *""*, "target": *"them"*, "polar expression": *"don't believe"*).

Aspect based sentiment analysis is a popular and widely researched topic in the NLP community (Wagner et al., 2014; Pontiki et al., 2015; Schouten and Frasincar, 2015). However, the SSA task goes beyond traditional aspect based sentiment

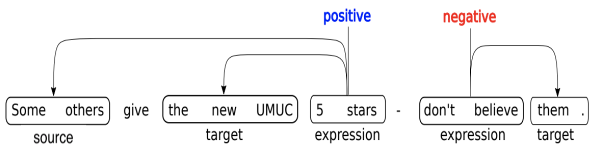


Figure 1: Sample sentence, source: (Barnes et al., 2021)

mining. The main challenges in SSA task are as follows:

- Focus is on correct **span detection** of source, targets, and polar expression elements as shown in Fig. 1.
- Correctly identifying **relationships** between source, target, and polar expression elements, as illustrated in the example above.
- Building a robust, generalized approach that is applicable *across domains* as well as *across languages*.

Most of the state-of-the-art (SOTA) approaches have explored joint learning, span based multi-task learning, BiLSTM encoders or CRF based approaches for SSA related tasks (Barnes et al., 2021; Li et al., 2019; Zhao et al., 2020; Chen et al., 2020). We propose a contextualized BERT based approach called 'CBERT', for capturing context information effectively and address the main challenges for SSA task as discussed above.

The main contributions of our work are:

- We propose a BERT based encoding approach and combine syntactic and pre-trained embeddings using a novel concatenation approach to extract and find opinion tuples effectively.
- We performed a detailed error analysis highlighting the main challenges of SSA task and suggest future directions for building a more efficient and effective system for SSA.

The outline of this paper is as follows: Section 2 describes the prior work in the area of SSA. Section 3 presents an abstract overview of our proposed system. Section 4 provides a detailed description of our CBERT system. Section 5 describes the datasets, tools and resources used for the experimental setup. Section 6 present the main results of the proposed system and describes an in-depth error analysis. Section 7 describes the main conclusions and leanings.

2 Background

Structured Sentiment Analysis (SSA) aims at capturing the natural hierarchical structure of different aspects, their corresponding sentiments in a sentence and the correlation between them. Almars et al. (2017) leveraged a hierarchical tree structure on the aspect terms, analyses the opinions on those aspect terms and connects them to the corresponding evidence for the same. Choi et al. (2006) identify two types of opinion-related entities — expressions of opinions and sources of opinions — along with the linking relation that exists between them. To identify the joint existence of opinion terms, they had utilized integer linear programming approach. Over years, this task was recognized further as Aspect Based Sentiment Analysis which focuses more on the precise identification of aspects and their corresponding sentiments (Chen et al., 2014), (Liu et al., 2012). Other intermediate works also focus on transition based end-to-end opinion extraction approaches (Zhang et al., 2019) which ignore the polarity classification subtask. Target Sentiment Analysis, which is another modified form of SSA focuses on extracting only sentiment targets and classifying their polarity (Jiang et al., 2011), (Mitchell et al., 2013).

Yang and Cardie (2012) is a strong baseline which suggests that the use of Conditional Random Fields could enhance the aspect terms prediction, given a smaller dataset with annotations, for training. During this time, some prominent research works discussed the advantage of semantic dependency parsing (Dozat and Manning, 2018), (Kurtz et al., 2020) which was leveraged for SSA task and for the establishment of the state-of-the-art SSA model which was proposed by (Barnes et al., 2021). This work formulates the SSA task as a dependency parsing problem and predicts all tuple components as a dependency graph. We aim to extend this aforementioned work and propose a modified architecture to improve the results across various language datasets.

3 Model Overview

For the SSA task, prior works have deployed separate models for detecting various elements (source, target and polar expression) (Barnes et al., 2022). However, this results in information regarding prediction of individual elements not being passed on to other element predictions.

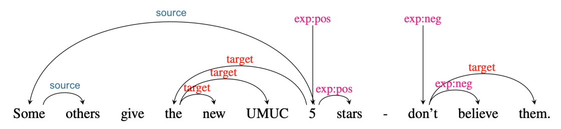


Figure 2: Sample sentence : head-first representation (Barnes et al., 2021)

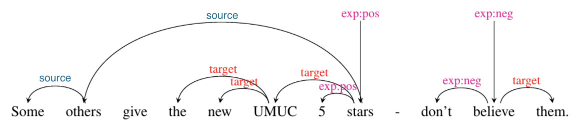


Figure 3: Sample sentence : head-final representation (Barnes et al., 2021)

For leveraging a joint learning technique, (Barnes et al., 2021) suggest two sets of approaches, *viz.*, head-first and head-final representation. *Head-first* considers spans of opinion elements using the first token of each element, *i.e.*, head token first, as shown in Fig. 2. *Head-final* considers the last token, *i.e.*, head token last, as shown in Fig. 3. Our proposed architecture was inspired by this joint learning technique and is elaborated in Section 4.

SSA task involves prediction of opinion elements along with the relationship between them, referred to as arc prediction. In this paper, we provide a comparison between two methods - VBERT, which uses only BERT embeddings and CBERT, which uses BERT embeddings along with concatenation of head and dependent BiLSTM blocks, detailed in Section 4. A comparison of results from these two approaches are provided in Section 6.

4 System Description

In this section we provide a detailed description of our proposed CBERT system. Fig. 4 presents the main architecture, comprising of seven main components, detailed below respectively.

4.1 Encoding Block

The *BERT context* shown in Fig. 4 represents BERT's role as an encoder. Our proposed model uses a BERT base multilingual cased[1] model (Devlin et al., 2018) which has t transformer layers to calculate the fine-tuned BERT vector from the input tokens, $x = (x_1, x_2, x_3, ..., x_n)$ of n padded sequence length. The output BERT fine-tuned vector is shown as, $b = (b_1^t, b_2^t, ..., b_n^t) \in R^{n*dim_b}$ where dim_b denotes it's dimension.

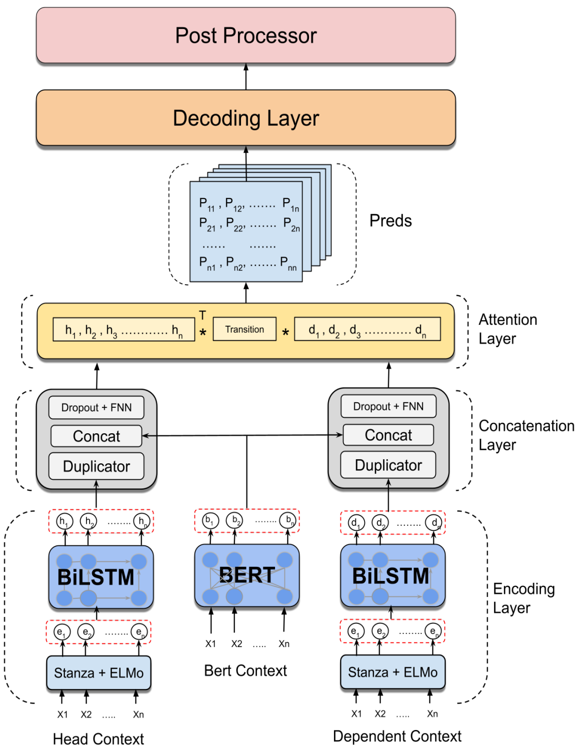


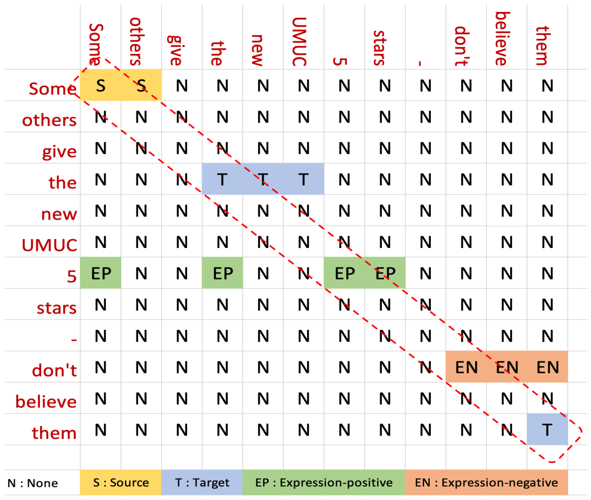
Figure 4: CBERT Architecture

The *Head Context* and *Dependent Context* as shown in Fig. 4, represent independent BiLSTM blocks where we fine-tune the syntactic and pretrained context in our CBERT system. However, in the VBERT system we do not have the aforementioned independent BiLSTM blocks (for more details refer Section A.4).

The input embedding in the BiLSTM block is a concatenation of Stanza[2] pretrained syntactic vectors (Qi et al., 2020) (U-pos, Lemma, Depparse) and ELMo[3] (Peters et al., 1802) embedding represented as, $e = (e_1, e_2, ..., e_n) \in R^{n*dim_e}$ where dim_e is the dimension of the input embedding. After passing these vectors through two independent identical BiLSTM blocks, we get the head and dependent fine-tuned vectors $h = (h_1^r, h_2^r, ..., h_n^r) \in R^{n*dim_c}$ and $d = (d_1^r, d_2^r, ..., d_n^r) \in R^{n*dim_c}$ where dim_c is the dimension of the fine-tuned vector and r is the number of recurrent layers in the BiLSTM block.

4.2 Target Encoding

We follow a head-first encoding (Barnes et al., 2021) which entails adding a label map to each position in the head-dependent 2D mapping, each label showing which category from ***(Expression-positive, Expression-negative, Source, Target, None)*** it belongs to. In the 2D mapping as shown in Fig. 5, heads are shown as rows, dependents as columns and a non ***None*** label shows if head-dependent relation exists or not. For denoting span the starting token of a certain span becomes the head and the other tokens as the dependents, as shown in Fig. 5, where *"the"* is linked to *("the","new","UMUC")*. Similarly, the head token of ***(Expression-positive, Expression-negative)*** labels point to the head tokens of ***(Source, Target)***, if a relation between them exists. For eg., *"5"* being linked to *"the"* as shown in Fig. 5. We also experimented with alternative encoding approaches mentioned in Section A.2



	Some	others	give	the	new	UMUC	5	stars	-	don't	believe	them
Some	S	S	N	N	N	N	N	N	N	N	N	N
others	N	N	N	N	N	N	N	N	N	N	N	N
give	N	N	N	N	N	N	N	N	N	N	N	N
the	N	N	N	T	T	T	N	N	N	N	N	N
new	N	N	N	N	N	N	N	N	N	N	N	N
UMUC	N	N	N	N	N	N	N	N	N	N	N	N
5	EP	N	N	EP	N	N	EP	EP	N	N	N	N
stars	N	N	N	N	N	N	N	N	N	N	N	N
-	N	N	N	N	N	N	N	N	N	N	N	N
don't	N	N	N	N	N	N	N	N	N	EN	EN	EN
believe	N	N	N	N	N	N	N	N	N	N	N	N
them	N	N	N	N	N	N	N	N	N	N	N	T

N : None   S : Source   T : Target   EP : Expression-positive   EN : Expression-negative

Figure 5: Encoding Structure

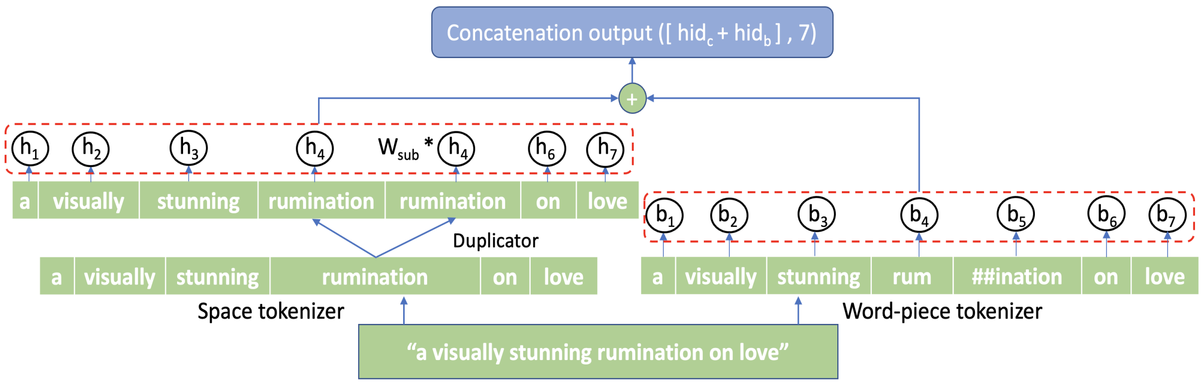
4.3 Concatenation layer

In this layer, we combine the BERT fine-tuned vector with the head and dependent fine-tuned vectors separately. The head and dependent vectors are based on a space based tokenizer while the BERT

[1] https://huggingface.co/bert-base-multilingual-cased

[2] https://stanfordnlp.github.io/stanza/index.html

[3] https://allenai.org/allennlp/software/elmo



Concatenation output ([hid_c + hid_b] , 7)
h1 h2 h3 h4 W_sub * h4 h6 h7
a visually stunning rumination rumination on love
Duplicator
a visually stunning rumination on love
Space tokenizer
b1 b2 b3 b4 b5 b6 b7
a visually stunning rum ##ination on love
Word-piece tokenizer
"a visually stunning rumination on love"

Figure 6: Concatenation of fine-tuned BERT with head and dependent fine-tuned vectors

vector is based on word-piece (Wu et al., 2016) tokenizer. Some of these space delimited tokens are further split by word-piece into sub-tokens. For eg., as shown in Fig. 6, the parent token *"rumination"* is split into sub-tokens *("rum","##ination")*. So we duplicate the parent token's vector,v as weighted vectors in the Duplicator as shown in Fig. 4. Higher weight is assigned to base-token *"rum"* and lower to sub-token *"##ination"*, as we want to put more significance on the base-token of the word for identifying span offsets correctly. Initializing sub-tokens with relatively low weighted vector instead of null vector helps us carry the base-word context till the sub-word. Following are the vector equations,

$$v = (v_1, v_2, ..., v_n)$$
$$v_{sub} = (W_{sub} * v_1, W_{sub} * v_2, ..., W_{sub} * v_n)$$
$$v_{base} = (W_{base} * v_1, W_{base} * v_2, ..., W_{base} * v_n)$$

Here v represents the parent token vector, v_{sub} the sub-token vector and v_{base} represents the base-token vector. The lower constant weight for sub-tokens is denoted by $W_{sub} \in [0, 1]$ compared to $W_{base} = 1$ assigned to base-token. The weight is decided after doing multiple runs with variations in W_{sub} as shown in Fig. 8. We choose $W_{sub} = 0.4$ empirically, as we get higher average scores in our experiments on the dev set.

4.4 Attention layer

The Concatenation layer outputs are then passed into the Attention layer which uses a transition matrix of $Shape(U) = dim_h * labels$. It transforms the head and dependent inputs to a 3D tensor matrix predicting the head-dependent relation arcs and labels jointly as represented in Fig. 4 as the Attention layer. The output is $Preds \in R^{n*n*labels}$ where $labels$ is the number of predicted labels and n is the padded sequence length. (for more details refer Section A.3)

4.5 Loss Function

We use a weighted cross-entropy function where the weights are assigned for each of the labels, l with respect to the target label frequency. We found out that the frequency of ***None*** label vs other labels was unbalanced. Hence, we conducted different runs with the following weight distribution $w = (w_{None}, 1, 1, 1, 1)$ by varying w_{None}. We choose the weights $w = (0.8, 1, 1, 1, 1)$ empirically, as we get higher average scores in our experiments on the dev set, as shown in Fig. 9.

$$loss_n = -w_{y_n} * log(\frac{exp(x_{n,y_n})}{exp(\sum_{l=1}^{L} x_{n,l})})$$
$$loss = -\sum_{n=1}^{N} \frac{loss_n}{\sum_{n=1}^{N} w_{yn}}$$

where x_n is the input probabilities, y_n is the corresponding target label, N is the batch length of input, w are the weights and L is the number of labels.

4.6 Decoding

In this subsection, we describe the methodology behind decoding the Attention layer output into the opinion pairs or tuples. As shown in Algorithm 1, in lines 2 to 12, we move along the main diagonal to find the predicted spans with labels of each category. Each of these spans have their starting offset denoted by the head pointing to itself as shown in Fig. 5. The end offset of a span is the last consecutive occurrence of the same tag as the head. For the next step shown in lines 13 to 22, we look at expression head's dependents to find if any of them are heads to other spans. This signifies a relation arc between spans.

Algorithm 1 Algorithm for Decoding the scores to tuples

Input: $Preds$, the 3D tensor output which contains the predicted label with maximum class probability for every head-dependent token pair.
Output: $Arcs$, which contain all the possible Source-Target-Expression tuples.

```
Require: $Preds \neq null$
 1: Initialize $Tar$, $Src$, $Exp$ and $Arcs$ sets to empty
 2: while Left index $l \leq length(Preds)$ and Right index $l < r \leq length(Preds)$ do
 3:     if $Preds(l,l) = T$ and all $Preds(l,r) = Preds(l,r-1)$ then
 4:         Add tokens $(t_l, ...t_r)$ to $Tar$
 5:     end if
 6:     if $Preds(l,l) = S$ and all $Preds(l,r) = Preds(l,r-1)$ then
 7:         Add tokens $(t_l, ...t_r)$ to $Src$
 8:     end if
 9:     if $Preds(l,l) \in (EN, EP)$ and all $Preds(l,r) = Preds(l,r-1)$ then
10:         Add tokens $(t_l, ...t_r)$ to $Exp$
11:     end if
12: end while
13: while Left index $l \leq length(Preds)$ and Right index $r \leq length(Preds)$ do
14:     if $Preds(l,l) \in (EN, EP)$ and $Preds(l,l) = Preds(l,r)$ and $l \neq r$ then
15:         if $t_r \in Tar$ then
16:             $Arcs \leftarrow (Tar(t_r), Exp(t_l))$
17:         end if
18:         if $t_r \in Src$ then
19:             $Arcs \leftarrow (Src(t_r), Exp(t_l))$
20:         end if
21:     end if
22: end while
```

4.7 Post processing

For post processing, in the generated tuples from the decoding layer, we remove overlapping Source or Target spans from edges of Expression spans. As shown in Example 1, "experience" is removed from the polar expression.

Example 1: *Thank you for such a wonderful experience.*
Raw tuple - ("source": "", "target": "*experience*", "polar expression": "*wonderful experience*")
Post processed tuple - ("source":"", "target": "*experience*", "polar expression": "*wonderful*").

5 Experimental Setup

5.1 Datasets & Evaluation Metric

The shared task has seven datasets across five languages that are used for final evaluation and ranking of submitted systems MPQA (Wiebe et al., 2005), NoRec (Øvrelid et al., 2020), Multibook (Barnes et al., 2018), Opener (Agerri et al., 2013) and Darmstadt (Toprak et al., 2010). We submitted our system for monolingual task, where we fine-tuned our model for each individual dataset mentioned above.

The experiments conducted as a part of this work use Sentiment-F1 (SF1) score (Barnes et al., 2021) as the evaluation metric, as used in SemEval-2022 Shared Task 10. This metric defines true positive as an exact match for each element of the opinion terms, averaging the overlap in predicted and annotated spans for each element across source, target and polar expressions.

5.2 Experimental Settings

We use the Pytorch[4] implementation of BERT base multilingual cased model pretrained on the 104 languages. The model has $t = 12$ transformer layers, the hidden size dim_h is 768 and 12 self-attention heads. The padding length used for encoding is 128. We use ELMo embeddings which are domain-general with 256-dimensions, pre-trained with $800M$ tokens and $28M$ parameters. The Pytorch BiLSTM implementation has hidden dimension of 64, number of layers as 2 and dropout probability as 0.3. The Concatenation layer has a output dimension as 32 and dropout probability of 0.2. The batch size is set to 16 for Train and 8 for devset. We adopt an Adam optimizer (Kingma and Ba, 2014) with learning rate of $2e-5$ and 500 warmup steps. After conducting train-dev runs for maximum 100 epochs we select best epoch based on Sentiment-F1 score on dev-set. A detailed view of these experiments have been included in Fig. 7 in Appendix. We conduct these experiments on a DGX server consisting of 8 Nvidia Tesla V100-SXM2 with 16GB V-RAM. We use multiple GPU's using the DataParallel[5] Pytorch class.

6 Experimental Results and Analysis

The detailed results of final submission are as shown in Table 1. The proposed CBERT and VBERT models performed relatively better than BiLSTM based head-first and head-final approaches. CBERT performed 9% relatively better than VBERT, supporting the hypothesis that the concatenation of contextualized information with syntactic information is more effective.

Our system achieved an average rank of 14/32 systems, based on the average scores across seven datasets. An in-depth analysis reveals that although

[4]https://huggingface.co/transformers/v1.2.0/
[5]https://pytorch.org/docs/stable/generated/torch.nn.DataParallel.html

Dataset	Head-first	Head-final	VBERT	CBERT	*Median*	*Best System*
Opener_en	0.524	0.542	0.576	**0.634**	0.623	0.760
MPQA	0.218	0.218	0.218	**0.283**	0.367	0.447
Darmstadt_unis	0.181	0.209	0.271	**0.320**	0.342	0.494
Opener_es	0.480	0.552	0.537	**0.595**	0.539	0.722
NoRec	0.254	0.272	0.315	**0.343**	0.329	0.529
Multibook_ca	0.529	0.543	0.595	**0.634**	0.525	0.728
Multibook_eu	0.501	0.536	0.564	**0.559**	0.478	0.739
Average Scores	0.384	0.410	0.439	**0.481**	0.458	0.631

Table 1: Results on the test set across seven datasets for five languages, scores of the final system is in bold face

our model performed quite good for five datasets over the median system, for two datasets *MPQA* and *Darmstadt_unis* our proposed system did not perform relatively well, thus hampering the average scores. The best system scores are also relatively quite high, than our proposed CBERT system. One of the potential reasons for relatively lower performance for two datasets could be the existence of excessive number of neutral labels in these two datasets, which was not separately handled in the experimental setup, as discussed in Section 3.2.

Following is a qualitative analysis to identify main challenges in the CBERT system.

- **Boundary Detection:** It was observed that there were differences in the source, target and polar expression boundaries based on punctuation marks, stopwords or adjectives. For eg., *It is really very basic.* contains polar expression as *very basic* in its gold annotations whereas CBERT predicted as *really very basic.*

- **Tuple Formation:** In sentences that contain a single target with multiple polar expressions, CBERT predicts them as a single tuple. For eg., *The rooms are clean and functional.* In this sentence, there is one target, *the rooms* and its two polar expressions are *clean* and *functional.* Gold annotations provided for the sentence have two separate tuples for polar expressions whereas CBERT predicts them as a single tuple.

- **Differences in Gold Annotation:** In the datasets, some data annotations exist only for texts which have an explicit indication towards the theme of the dataset. For instance, the *NoRec* dataset is a collection of restaurant reviews. Hence, the data which do not have an explicit indication towards restaurant related comments are not annotated. However, CBERT does not distinguish between themes and produces predictions for such texts as well. For eg., *As a gold member I enjoyed an upgrade to a very large room with lots of floor space .* does not have any annotations. However, CBERT predicts Source as *I*, Target as *room* and Polar Expressions as *enjoyed, very large, lots of floor space*.

A detailed set of examples are shown in Table 2.

7 Conclusion and Future Scope

This paper discusses the proposed system and experimental results on SSA task at SemEval-2022. The proposed model called CBERT, assumes a joint learning technique using BERT-base multilingual model embeddings along with contextualized embeddings created using pretrained Stanza vectors and pretrained ELMo embeddings. The arc prediction between opinion term elements are achieved using an Attention layer. The proposed CBERT approach performed relatively better than BiLSTM approaches and VBERT approach. We find that using BERT based encoding, along with concatenation of contextualized information with syntactic information is more effective for SSA task.

For improving CBERT, leveraging 2D CRFs (Zhu et al., 2005) can be explored for better span detection over longer texts. Different joint learning architecture on top of CBERT's encoding like SDRN (Chen et al., 2020) can be leveraged. Approaches to fine-tune models for specific tasks (Zhao et al., 2020) is also a potential area for exploration.

References

Rodrigo Agerri, Montse Cuadros, Sean Gaines, and German Rigau. 2013. OpeNER: Open polarity enhanced named entity recognition. In *Sociedad Es-*

pañola para el Procesamiento del Lenguaje Natural, volume 51, pages 215–218.

Abdulqader Almars, Xue Li, Xin Zhao, Ibrahim A. Ibrahim, Weiwei Yuan, and Bohan Li. 2017. Structured sentiment analysis. pages 695–707.

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.

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.

Shaowei Chen, Jie Liu, Yu Wang, Wenzheng Zhang, and Ziming Chi. 2020. Synchronous double-channel recurrent network for aspect-opinion pair extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6515–6524, Online. Association for Computational Linguistics.

Zhiyuan Chen, Arjun Mukherjee, and B. Liu. 2014. Aspect extraction with automated prior knowledge learning. In *ACL*.

Yejin Choi, Eric Breck, and Claire Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 431–439, Sydney, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent Twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 151–160, Portland, Oregon, USA. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Robin Kurtz, Stephan Oepen, and Marco Kuhlmann. 2020. End-to-end negation resolution as graph parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 14–24, 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.

Kang Liu, Liheng Xu, and Jun Zhao. 2012. Opinion target extraction using word-based translation model. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, page 1346–1356, USA. Association for Computational Linguistics.

Margaret Mitchell, Jacqui Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1654, Seattle, Washington, USA. Association for Computational Linguistics.

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.

ME Peters, M Neumann, M Iyyer, M Gardner, C Clark, K Lee, and L Zettlemoyer. 1802. Deep contextualized word representations. arxiv 2018. *arXiv preprint arXiv:1802.05365*, 12.

Maria Pontiki, Dimitrios Galanis, Harris 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.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. *arXiv preprint arXiv:2003.07082*.

Kim Schouten and Flavius Frasincar. 2015. Survey on aspect-level sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):813–830.

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.

Joachim Wagner, Piyush Arora, Santiago Cortés Vaíllo, Utsab Barman, Dasha Bogdanova, Jennifer Foster, and Lamia Tounsi. 2014. Dcu: Aspect-based polarity classification for semeval task 4. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 223–229.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2-3):165–210.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-Markov conditional random fields. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1335–1345, Jeju Island, Korea. Association for Computational Linguistics.

Meishan Zhang, Qiansheng Wang, and Guohong Fu. 2019. End-to-end neural opinion extraction with a transition-based model. *Inf. Syst.*, 80:56–63.

He Zhao, Longtao Huang, Rong Zhang, Quan Lu, and Hui Xue. 2020. SpanMlt: A span-based multi-task learning framework for pair-wise aspect and opinion terms extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3239–3248, Online. Association for Computational Linguistics.

Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, and Wei-Ying Ma. 2005. 2d conditional random fields for web information extraction. In *Proceedings of the 22nd international conference on Machine learning*, pages 1044–1051.

A Appendix

A.1 Additional Experimental Settings

- In Fig. 7, each of the values correspond to SF1 scores obtained over devset, post to training on Training set (train/dev). It was observed that scores are consistent at 100 epochs.

- In Fig. 8, we ran train/dev experiments for 60 epochs to record SF1 scores by varying the sub-token weight w_{sub} to find optimal w_{sub} weights to be used in the Concatenation layer.

- In Fig. 9, we conducted train/dev experiments for 60 epochs to record SF1 scores by varying weight for ***None*** label w_{None} to find optimal weight to be used in weighted cross-entropy function.

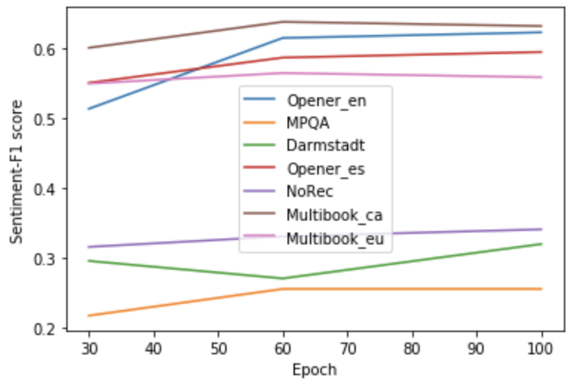


Figure 7: Graph showing Sentiment-F1 scores for our proposed model against the number of epochs.

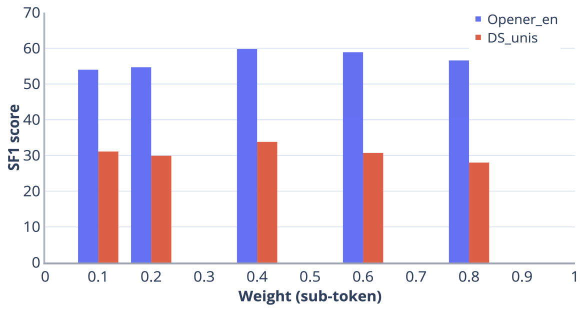


Figure 8: Sub-token weight vs devset SF1

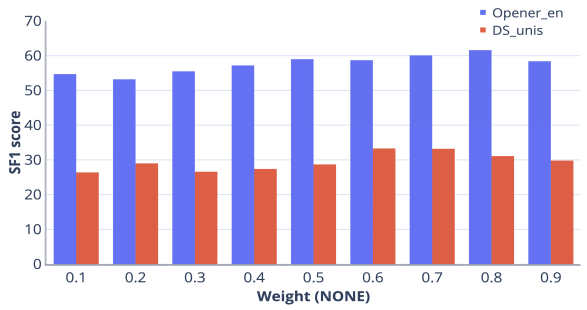


Figure 9: Weight(None) vs devset SF1

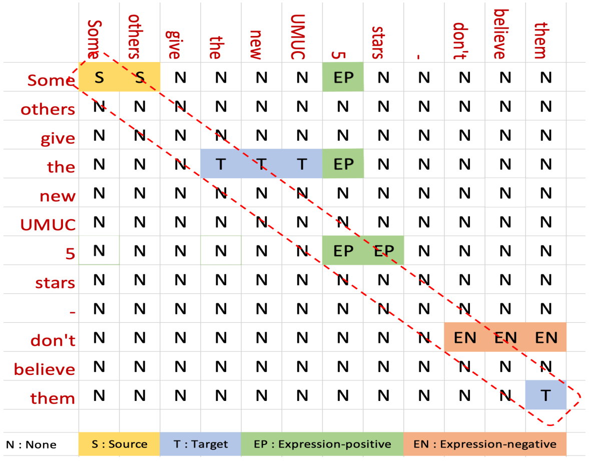
A.2 Alternative encoding schemes

We introduce another scheme for encoding where we mirror the labels of the lower triangular matrix to the upper triangular matrix along the main diagonal using results from our original encoding scheme as shown in Fig. 10. This helps us reduce the size of Attention layer's output $Preds$ by half,

Error Category	Remarks	Example
Boundary Detection	Based on Adjectives	For, *The first floor 24 hr bar was well run and no matter what time of day there was always a waiter on hand to serve .* CBERT predicts Polar Expression as *there was* whereas the annotaions contain *there was always*
	Based on Punctuations	For, *Robbed in elevator of hotel !* CBERT predicts Polar Expression as *Robbed in elevator of hotel* whereas annotations contain punctuation
	Based on Stopwords	For, *The best about this hotel is its location .* CBERT predicts Polar Expression as *The best about* whereas annotations contain *The best*
	Long Sentences	For, *There are good conference rooms with all necessary infrastructure ,good location allowing you to have nice evening , pool , restaurants , internet , 5 minutes to airport everything you need to do a business and relax after that .* CBERT predicts Source and Target efficiently but fails to detect Polar Expressions
Tuple Formation		For *Relax and enjoy* CBERT predicts two Polar Expressions separately as *Relax*, *enjoy* whereas annotations contain *Relax and enjoy* as single expression
Gold Annotations		For, *No words for this country and its people .* no annotations were provided

Table 2: Examples for Error Analysis categories

as we can always mask the lower half while applying the loss function.



	Some	others	give	the	new	UMUC	5	stars	-	don't	believe	them
Some	S	S	N	N	N	N	EP	N	N	N	N	N
others	N	N	N	N	N	N	N	N	N	N	N	N
give	N	N	N	N	N	N	N	N	N	N	N	N
the	N	N	N	T	T	T	EP	N	N	N	N	N
new	N	N	N	N	N	N	N	N	N	N	N	N
UMUC	N	N	N	N	N	N	N	N	N	N	N	N
5	N	N	N	N	N	N	EP	EP	N	N	N	N
stars	N	N	N	N	N	N	N	N	N	N	N	N
-	N	N	N	N	N	N	N	N	N	N	N	N
don't	N	N	N	N	N	N	N	N	N	EN	EN	EN
believe	N	N	N	N	N	N	N	N	N	N	N	N
them	N	N	N	N	N	N	N	N	N	N	N	T

N : None | S : Source | T : Target | EP : Expression-positive | EN : Expression-negative

Figure 10: Upper Triangular encoding

A.3 Attention layer

The output of Attention layer is represented as, $y = (y_{none}, y_{exp-pos}, y_{exp-neg}, y_{tar}, y_{src}) \in R^{n*n*labels}$ which is a 3D tensor where y_{label} is a 2D tensor corresponding to a certain label, $labels$ is the number of predicted labels and n is the padded input sequence length. The matrix transformations done in the attention module (Bi-linear label Attention) (Barnes et al., 2021) are shown below,

$$score(h_i, d_j) = h_i^T * U * d_j \text{ where}$$

$$h_i = Concat_{head}(c_i),$$
$$d_i = Concat_{dep}(c_i)$$
$$Shape(U) = dim_h * labels$$

A.4 VBERT overview

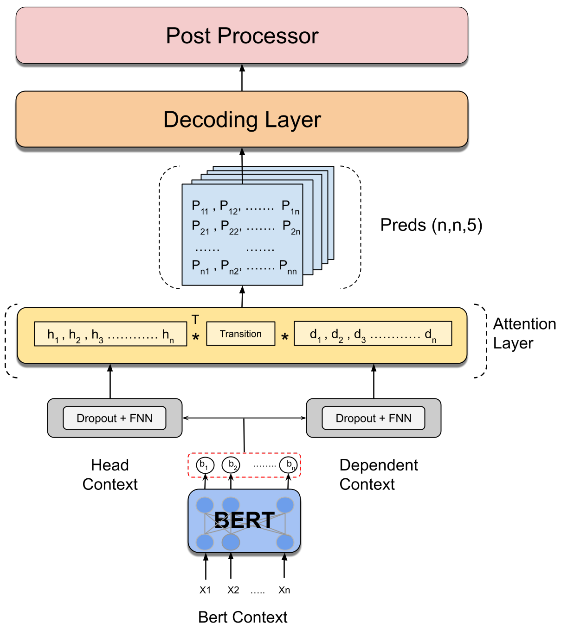

Post Processor
Decoding Layer
$P_{11}, P_{12}, \ldots\ldots P_{1n}$
$P_{21}, P_{22}, \ldots\ldots P_{2n}$
$P_{n1}, P_{n2}, \ldots\ldots P_{nn}$
Preds (n,n,5)
$h_1, h_2, h_3 \ldots\ldots\ldots\ldots h_n$
T
Transition
$d_1, d_2, d_3 \ldots\ldots\ldots\ldots d_n$
Attention Layer
Dropout + FNN
Dropout + FNN
Head Context
Dependent Context
b_1 b_2 b_l
BERT
X1 X2 ..... Xn
Bert Context

Figure 11: VBERT Architecture

This architecture shown in Figure 11 uses BERT as an encoder and splits the BERT output into head and dependent context.