DLSU Research Congress 2024
De La Salle University, Manila, Philippines
June 20 to 22, 2024

Aspect-Based Sentiment Classification on Filipino Tweets

John Paul Guzman[1,*], Dr. Charibeth Cheng[2], and Dr. Angelyn Lao[1]
[1] *Department of Mathematics and Statistics, De La Salle University*
[2] *Department of Software Technology, De La Salle University*
**Corresponding Author: john_paul_d_guzman@dlsu.edu.ph*

Abstract: Sentiment analysis involves extracting opinions from text, while aspect-based sentiment classification (ABSC) determines sentiments toward specified aspects in a given text. Knowledge about the specific target of the opinion allows for a more fine-grained analysis of sentiments, resulting in richer insights that can lead to further applications. This paper introduces a model for the ABSC task that is adaptable to texts in various languages and topics. Moreover, we outline techniques to overcome the challenges associated with multilingual ABSC, such as working with limited data. Our model was evaluated on four datasets to demonstrate its capacity to learn ABSC in different languages and topics. Three benchmark datasets in English were used to compare our model with existing ones. Additionally, we created a multilingual dataset in English and Filipino that consists of tweets published during the COVID-19 pandemic. Our model achieved slight improvements over similar models in the benchmark datasets. Meanwhile, its performance in the multilingual dataset is comparable to that observed in the benchmark datasets.

Key Words: machine learning; sentiment analysis; aspect-based sentiment classification

1. INTRODUCTION

Social media allows people to share thoughts and opinions on a massive scale, generating large amounts of data that can be analyzed for valuable insights (Ghani et al., 2019). A common method for extracting information from text is sentiment analysis, which refers to the process of extracting opinions or sentiments from bodies of texts (Pang, Lee, et al., 2008). Aspect-based sentiment analysis (ABSA) is a fine-grained approach to sentiment analysis where the target of the sentiment is taken into account (Liu, 2012). Knowledge about the specific target of the opinion allows for a more fine-grained analysis of sentiments. For instance, smartphone reviews may contain sentiments about various aspects of the phone like battery life, camera, and screen quality. Aspect-based sentiment classification (ABSC) is a task within ABSA that involves classifying the sentiment polarity towards an aspect, which can be either *positive*, *negative*, or *neutral*. More formally, given a sentence $S = w_1 w_2 \dots w_k$ defined as a sequence of words w_i and an aspect target A which is a subsequence of S, the task is to determine the sentiment polarity towards the aspect A.

Aspect-level analysis introduces new challenges due to its fine-grained nature (Nazir et al., 2020). We illustrate these using examples taken from Pontiki et al. (2014). Let us consider the sentence: "*The fish is fresh, but the variety of fish is nothing out of the ordinary.*" Here, the sentiment polarity and context words related to "*fish*" are *positive* and "*fresh*," respectively. On the other hand, the corresponding sentiment polarity and context words for "*variety of fish*" are *negative* and "*nothing out of the ordinary*," respectively. Another challenge involves aspects with long-term dependencies. Consider the sentence: "*The manager claimed that he could not compensate us for anything on the bill, which just shows the lack of sophistication from the entire group.*" In this example, the target "*manager*" and the context words "*lack of sophistication*" are separated by a large distance in the sentence.

Numerous machine learning methods have been proposed in order to tackle ABSC. Traditional machine learning models, such as the Support Vector Machine (SVM), have been used for ABSC (Kiritchenko et al., 2014). However, traditional machine learning models require rich knowledge-bases in order to function. For instance, Kiritchenko et al. (2014) utilizes a combination of a lexicon, a part-of-speech tagger, and a dependency parser to generate the features required by the SVM

classifier. The dependence on engineered features is a problem because they are difficult to produce. Deep learning models are employed to tackle ABSC in an end-to-end manner, i.e., without the need for engineered features (Zhou et al., 2019).

In this paper, we introduce a deep learning model for the ABSC task that is adaptable to texts in various languages and topics. Section 2 discusses preliminary concepts, the architecture of our model, the datasets used in experiments, and the training hyperparameters. Section 3 showcases the performance of our model and the comparisons to other ABSC models. Lastly, Section 4 covers conclusions and potential areas for future work.

2. METHODOLOGY

2.1 Preliminary Concepts

A Recurrent Neural Network (RNN) is a type of neural network with cyclical dependencies (Goodfellow et al., 2016). In Natural Language Processing (NLP) tasks, documents are often divided into token sequences, which are then processed by an RNN one token at a time. Tang, Qin, Feng, et al. (2016) have shown that RNNs can be somewhat effective in ABSC.

A well-known issue with RNNs is that they struggle with learning long-term dependencies (Hochreiter et al., 2001; Pascanu et al., 2013). One reason that contributes to this difficulty is the problem of vanishing and exploding gradients (Bengio et al., 1994). Gated RNNs, such as the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), were introduced to mitigate this issue (Cho et al., 2014; Hochreiter & Schmidhuber, 1997). These RNNs typically do not constitute the entire neural network but instead serve as a component of a larger network. This specific component is referred to as an RNN cell.

Another technique for managing long-term dependencies is the use of attention mechanisms. Attention mechanisms enable RNNs to selectively focus on various positions within an input sequence, which helps alleviate the challenge of learning long-term dependencies (Bahdanau et al., 2015). The attention mechanism takes three pieces of data, namely the query, keys, and values, and then produces a summary of relevant information from an input sequence (Niu et al., 2021). The key is a representation of the sequence that encodes the positional information of each feature. The query represents the information that we aim to retrieve from the sequence. Meanwhile, the value represents the information associated with each feature. Wang et al. (2016) introduced an ABSC model that incorporates an attention mechanism, resulting in notable improvements over the previously established baseline for deep learning models.

Utilizing external memory is yet another technique for overcoming the issue of learning long-term dependencies. Many neural networks have a limitation in that they can only retain a limited amount of information at a time. Memory networks resolve this limitation by storing all the pertinent data in external memory, which they can read from and write to (Weston et al., 2015). This approach enables the network to store and manipulate data for extended periods. Many memory networks share common components (Graves et al., 2014; Mao et al., 2019; Tang, Qin, & Liu, 2016):

- A memory matrix that holds a collection of feature vectors within its columns.
- An addressing mechanism that is an attention mechanism over the memory matrix.
- Reading and writing mechanisms that read from and write to the memory matrix.
- A controller RNN that drives the various mechanisms.

Tang, Qin, and Liu (2016) introduced a memory network, called MemNet, which achieved a significant improvement over previous deep learning models. Since then, a number of memory networks, such as Recurrent Attention on Memory (RAM) and Attentive Neural Turing Machine (ANTM), have been developed that further improve the performance of deep learning models for ABSC (Chen et al., 2017; Mao et al., 2019).

2.2 Embedding Models

A word embedding is a vector representation of a word that preserves some of its semantic information. In particular, words with similar meanings have close vector representations. The use of pre-trained word embeddings has been known to boost performance in various NLP tasks (Turian et al., 2010). In this study, we use the Bidirectional Encoder Representations from Transformers (BERT) model to generate embeddings with a dimensionality of $d_w = 768$ (Devlin et al., 2019).

In our experiments with the English datasets, we employed the uncased BERT model pretrained on the BooksCorpus and English Wikipedia (Devlin et al., 2019). For the multilingual dataset, we conducted experiments using three pre-trained BERT models: RoBERTa Tagalog (RoBERTa-Tl), Multilingual BERT (mBERT), and Cross-lingual Language Model-RoBERTa (XLM-R) (Conneau et al., 2020; Cruz & Cheng, 2022; Devlin et al., 2019). The RoBERTa-Tl model was pretrained on a collection of Filipino text known as the TLUnified dataset (Cruz & Cheng, 2022). On the other

hand, mBERT was pretrained on Wikipedia articles from the top 100 languages with the largest Wikipedias (Devlin et al., 2019). Lastly, XLM-R was pretrained on the Common Crawl corpus in 100 languages (Conneau et al., 2020). RoBERTa-Tl was chosen because a significant portion of our dataset is in Filipino, while mBERT and XLM-R were selected for their effectiveness on multilingual tasks (Szolomicka & Kocon, 2022).

Due to the significant disparity between the domains of the pretraining corpus and the FHT dataset, we performed additional pretraining steps on a new corpus. In other words, we fine-tune the embedding models on a new corpus and refer to this as domain adaptation (Howard & Ruder, 2018). These models were pretrained on the Masked Language Modeling (MLM) task (Devlin et al., 2019). MLM is a self-supervised task that relies on unannotated text data. In this task, the input to the model is tokenized text, where each token has a probability of being replaced with a mask, and the output is the same text with no masks. The pretraining objective is to predict the original tokens that were masked based on the surrounding context. Through this process, BERT acquires language representations that prove useful for various NLP tasks (Rogers et al., 2021).

We used the tweets gathered by Chan et al. (2022) as our corpus for domain adaptation, excluding those that appear in the test set of our dataset. Each tweet is preprocessed by removing URLs, hashtags, mentions, and emojis, and then converting it to lowercase. The domain adaptation hyperparameters were adjusted to be compatible with a NVIDIA L4 GPU, as detailed in Table 1. Although RoBERTa-Tl, mBERT, and XLM-R were all initially pretrained on cased text, we chose to use uncased text due to the inconsistent casing observed in tweets.

Table 1. Domain adaptation hyperparameters

Hyperparameter	RoBERTa-Tl	mBERT	XLM-R
MLM probability	0.15		
Optimizer	Adafactor (Shazeer & Stern, 2018)		
Number of steps	10,000		
Sequence length	100	100	80
Batch size	150	128	100

2.3 Network Architecture

Our model is an RNN equipped with an attention mechanism and external memory. The model takes the strings *aspect_term* and *text*, and then outputs a probability distribution over three categories: *negative*, *neutral*, and *positive*. The *aspect_term* and *text* are passed through an embedding layer that uses the word embeddings from Subsection 2.2.

The embedding layer produces an embedding matrix $E \in \mathbb{R}^{d_w \times k}$, where d_w is the length of the word embeddings, and k is the sequence length. We let $E(i)$ denote the i-th column of E, which is the embedding for the i-th word in the input string. When the input string contains more than k words, any excess words are discarded. Conversely, if the input string has fewer than k words, we pad the remaining columns of matrix E with zero vectors. In addition to E, the embedding layer generates a binary mask $\mu \in \{0,1\}^k$, where $\mu(i)$ is 0 if the i-th column of E is padding and 1 otherwise. We refer to E_{txt} and E_{asp} as the embedding matrices for the *text* and *aspect_term*, respectively, while μ_{txt} and μ_{asp} are the corresponding binary masks.

The binary masks are useful for operations that take padding into consideration. For instance, the aspect representation $A \in \mathbb{R}^{d_w}$ is computed as the average of aspect embeddings, excluding the padding; i.e.,

$$A = \sum_{i=1}^{k} \frac{\mu_{asp}(i)\, E_{asp}(i)}{\Sigma_{j=1}^{k} \mu_{asp}(j)}.$$

The text embedding matrix E_{txt} and mask μ_{txt} along with the aspect representation A are fed into our network. Our network utilizes multiple types of memory to enhance the capabilities of the controller, where each type of memory is analogous to a tape in a multi-tape Turing machine. Hence, we refer to our network as the Multi-tape NTM (MNTM). The concept behind our network involves increasing the controller's memory capacity to enhance the overall capacity of the network.

The network employs two types of memory. The first type of memory is the long-term memory matrix $L = E_{txt} \in \mathbb{R}^{d_w \times k}$. This functions as a repository for the text embedding matrix E_{txt} and remains constant over time. The second type of memory is the short-term memory matrix $S \in \mathbb{R}^{d_w \times k}$, which is initialized with zeros. Short-term memory operates as a transient storage for data subjected to manipulation over time, much like how humans employ scratch paper for lengthy calculations.

The addressing mechanism consists of three heads: a read head on L, a read head on S, and a write head on S. Each head takes a query $q \in \mathbb{R}^{d_w}$ and outputs a vector of weights w, given by

$$w = softmax(scores(q, L)). \quad \text{(Eq. 1)}$$

The $scores$ function is computed as follows:

$$scores(q,\ L)\ = \widehat{scores}(q,\ L) - \left(M\ *\ \neg\mu_{txt}\right),$$
$$\widehat{scores}(q,\ L)\ = \left\{v_h^{\intercal} tanh\left(W_h[q; L(i)] + b_h\right): i\ =\ 1, \ldots, k\right\},$$

where M is a large positive number, $\neg$ is element-wise binary negation, $[\cdot;\cdot]$ denotes vector concatenation, and $W_h \in \mathbb{R}^{k \times 2d_w}, b_h \in \mathbb{R}^k, v_h \in \mathbb{R}^k$ are learned parameters for each head. Our model utilizes one query vector q_L for long-term memory and another q_S for short-term memory. We let w_{rL} denote the weights generated by the read head on L with the query q_L. Furthermore, w_{rS} and w_{wS} denote the weights generated by the read and write heads on S with the query q_S.

The reading and writing mechanisms utilize the aforementioned weights to interact with memory. First, we extract the read vectors r_L and r_S from each type of memory, given by

$$r_L = \sum_{i=1}^{k} w_{rL}(i)\ L(i)\ , \text{and}\ r_S = \sum_{i=1}^{k} w_{rS}(i)\ S(i). \quad \text{(Eq. 2)}$$

Then, update the short-term memory S using an erase operation followed by an add operation

$$S(i) \leftarrow S(i)\left[1\ -\ w_{wS}(i)\ e\right] + w_{wS}(i)\ a,$$

where e is the erase vector, and a is the add vector.

Note that the same key matrix, L, is applied to all the read and write heads, as indicated in (Eq. 1). In contrast, different value matrices are used based on the type of memory, as shown in (Eq. 2). Our approach maintains consistent positional representations among the different types of memory. This gives us a way to address both types of memory while keeping their contents separate. As a result, information from the text embeddings is not lost over time in long-term memory. Meanwhile, we retain the capability to manipulate semantic cues within the short-term memory.

Our controller RNN is a stacked LSTM or GRU, an example of which is illustrated in Fig. 1. The query, erase, and add vectors are produced from the previous controller output h_{t-1} as follows:

$$\left[q_L; q_S; e; a\right] = sigmoid\left(W_c h_{t-1} + b_c\right),$$

where $W_c \in \mathbb{R}^{4d_w \times d_h}$ and $b_c \in \mathbb{R}^{4d_w}$ are learned parameters, and d_h is the dimension of the controller outputs. The succeeding controller output h_t is computed by the controller RNN

$$h_t = RNN\left(\left[h_{t-1}; A; r_L; r_S; x_t\right]\right),$$

where A is the aspect representation, x_t is the current word embedding, and r_L and r_s are the read vectors. For the padding tokens, we forego passing them to the controller and set $h_t = 0$. The controller takes various pieces of information into account, including aspect, contents of memory, and individual words being processed. It then constructs output representations that are used for classification and for driving various mechanisms. The interactions between memory and controller are visually depicted in Fig. 2.

We apply an attention mechanism over the controller outputs, similar to the approach of Wang et al. (2016). The attention weights att are calculated as

$$att\ =\ softmax\left(w^{\intercal} tanh\left(\left[W_h H; W_a V_a\right]\right) - \left(M\ *\ \neg\mu_{txt}\right)\right),$$

where $H = \left[h_1, \ldots, h_k\right] \in \mathbb{R}^{d_h \times k}$, $V_a = [A, \ldots, A] \in \mathbb{R}^{d_w \times k}$, and $w \in \mathbb{R}^{d_w + d_h}$, $W_h \in \mathbb{R}^{d_h \times d_h}$, $W_a \in \mathbb{R}^{d_w \times d_w}$ are learned parameters. Finally, we produce a probability distribution p using the approach of Mao et al. (2019):

$$\left.\begin{aligned} p\ &=\ softmax\left(W_s h^*\ +\ b_s\right), \\ h^*\ &=\ tanh\left(W_r r_s\ +\ W_x h_k\right), \\ r_s\ &=\ Hatt^{\intercal}, \end{aligned}\right\} \quad \text{(Eq. 3)}$$

where $W_s \in \mathbb{R}^{3 \times d_h}$, $b_s \in \mathbb{R}^3$, $W_r \in \mathbb{R}^{d_h \times d_h}$, and $W_x \in \mathbb{R}^{d_h \times d_h}$ are learned parameters. The entries of p represent the predicted probabilities for *negative*, *neutral*, and *positive* sentiment, respectively. The predicted label is given by the $argmax$ of p. For example, if $p = (0.2, 0.5, 0.3)$, then the predicted sentiment polarity is *neutral*. Once this network is trained, it can perform ABSC on new documents. This is where potential applications arise.

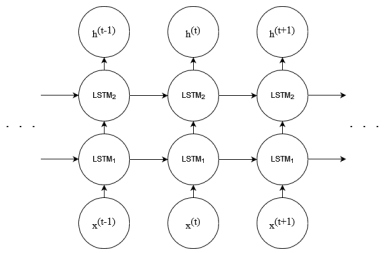

Fig. 1. Stacked LSTM. In this example, two LSTM cells are used to produce the variable $h^{(t)}$. The first LSTM takes the input $x^{(t)}$ and generates an output, which is then used as input by the second LSTM to produce $h^{(t)}$.

2.4 Datasets

We utilize three benchmark datasets in English that are commonly used in ABSC studies (Chen et al., 2017; Kiritchenko et al., 2014; Mao et al., 2019; Tang, Qin, & Liu, 2016; Tang, Qin, Feng, et al., 2016). The Laptops and Restaurants datasets originate from the

SemEval2014 Task 4 Subtask 2 (Pontiki et al., 2014). Both datasets contain reviews collected from various online review platforms, such as Yelp and TripAdvisor, about laptops and restaurants. The third benchmark dataset is the Tweets dataset collected by Dong et al. (2014). It contains tweets about various celebrities, products, and companies.

We refer to our multilingual dataset as the FHT dataset (Guzman, 2024). This dataset consists of tweets posted between December 4, 2020, and June 4, 2021 (Chan et al., 2022). These tweets were filtered to ensure they originate from the Philippines based on geolocation and contain at least one of the specified health-related keywords: "*covid-19*," "*covid*," "*coronavirus*," "*corona*," "*tb*," "*tuberculosis*," "*WorldTBDay*," or "*TBFreePh*." Aspect terms were selected based on topics frequently discussed during the pandemic, including but not limited to: different vaccine brands, travel bans, community quarantine, the use of masks/face shields, hospital capacity, government agencies, and isolation facilities (Guzman, 2024). We excluded aspects that may relate to tuberculosis, such as "*cough*," "*ubo*," and "*fatigue*." In total, 590 aspects were identified (Guzman, 2024).

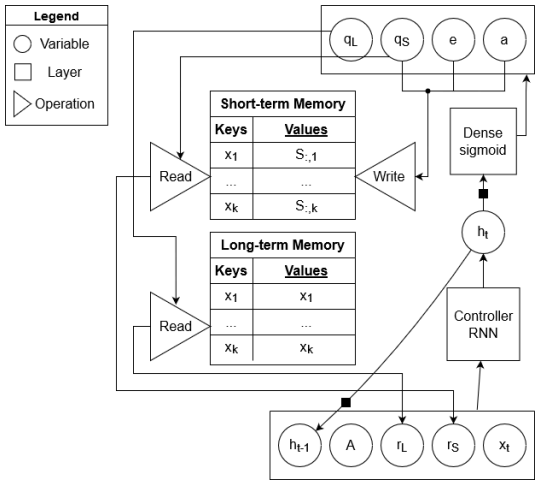

Fig. 2. Interactions between memory and controller. The circular shapes represent variables, while groupings of these circles signify vector concatenation. Arrows indicate dependencies on specific variables, with an arrow featuring a black rectangle indicating a one-step time delay. Squares represent different network layers, while triangles denote memory operations. At each timestep, the memory operations are performed using the previous controller output. Then, we input the vectors retrieved from memory, along with the aspect and the current token embedding, in order to produce the succeeding controller output.

Two human annotators were hired to label the sentiment polarity towards an aspect term in a given tweet. These annotators are native Filipino speakers who are fluent in English and have backgrounds in jobs requiring communication skills in both languages, such as teaching. For each tweet-aspect pair, they were instructed to choose from four options: *positive* when the sentiment towards the aspect is positive, *negative* when the sentiment towards the aspect is negative, *neutral* when the sentiment towards the aspect is neither positive nor negative, and *no_relation/conflicting* when there is no sentiment expressed towards the aspect or when the sentiment is conflicting.

The two sets of annotations were used to create the FHT dataset. Examples labeled as *no_relation/conflicting* were removed. We also removed examples where the annotations do not match, such as when one annotator labels a tweet-aspect pair as *positive* while another annotator labels it as *negative*. This reduces the number of available examples for our dataset but lessens the noise in labeling. After these removals, we were left with a total of $N = 6600$ examples. Finally, we randomly divided the examples into a 60-20-20 split, with 60% allocated for training, 20% for validation, and the remaining 20% for testing. We ensured that tweets occurring in one split do not appear in another. This precaution prevents instances where a tweet seen during training reappears in the test set. Our dataset is publicly available for research purposes.[1]

We assessed the inter-annotator agreement between the two annotators using Cohen's Kappa statistic κ (McHugh, 2012). This value ranges from -1 to 1, with higher values indicating stronger agreement between annotators and suggests that different annotators are likely to assign similar labels to the same data. The comparison between the two annotators resulted in $\kappa = 0.5116$, suggesting a degree of agreement but at a relatively low level (McHugh, 2012).

An overview of the datasets used in our experiments can be found in Table 2. For datasets without predefined validation sets, we created them by randomly selecting 10% from the training data.

2.5 Training Hyperparameters

The validation sets were used for early stopping and tuning hyperparameter values. We then employed these hyperparameters to train and evaluate our model. Our model was trained using a machine equipped with an NVIDIA GeForce RTX 3080 GPU with

[1] https://github.com/johnpaulguzman/FHT-dataset

8GB of VRAM, an Intel Core i7-10875H CPU, and 32GB of RAM.

The objective used during training is the cross-entropy loss with L_2 regularization and a regularization strength of $\lambda = 0.0001$. The chosen optimizer for training is Adam with an initial learning rate of 0.001 (Kingma & Ba, 2015). The embedding layer is designated as non-trainable, indicating that the word embeddings will not change throughout the training process. The sequence length k is set to 40. Dropout is applied to the variables r_s, h_k, and h^* from (Eq. 3), as well as to both the input and recurrent states of the controller cells (Srivastava et al., 2014). The dropout rate for these variables is set to 0.50. Label smoothing is applied with a parameter value of 0.15 for the Restaurants dataset and 0.09 for the other datasets (Goodfellow et al., 2016). The batch size is 128, and the training lasts for a maximum of 100 epochs with a patience parameter of 40.

Table 2. Overview of the datasets

Dataset	Negative	Neutral	Positive	Total
Laptops Train	780	414	889	2083
Laptops Valid	86	46	98	230
Laptops Test	128	169	341	638
Restaurants Train	725	570	1948	3243
Restaurants Valid	80	63	216	359
Restaurants Test	196	196	728	1120
Tweets Train	1404	2815	1405	5624
Tweets Valid	156	312	156	624
Tweets Test	173	346	173	692
FHT Train	1682	1609	594	3885
FHT Valid	584	543	230	1357
FHT Test	574	553	231	1358

3. RESULTS AND DISCUSSION

We begin by evaluating our models' performance on the benchmark datasets. Accuracy and macro F1 scores are shown in Table 3, alongside other ABSC models. We observe a slight improvement in comparison to similar RNN-based ABSC models. When comparing MTNM with ANTM+BERT, we note an improvement in mean accuracy by 0.0097 and in mean Macro F1 by 0.0093 on the benchmark datasets. Similarly, when comparing MNTM with ANTM+$BERT_L$, we observe improvements in mean accuracy by 0.0004 and in mean F1 score by 0.0046 for the benchmark datasets. Note that performance gains were achieved with the ANTM by employing a larger BERT model, $BERT_L$, which generates 1024-dimensional word embeddings. Unfortunately, we were unable to evaluate the performance of $BERT_L$ in conjunction with MNTM due to insufficient GPU VRAM.

Next, we evaluate the performance of the MNTM on the FHT dataset using various embedding models from Subsection 2.2. The corresponding accuracy and macro F1 scores are presented in Table 4. We first evaluated performance for each of the pretrained models: RoBERTa-Tl, mBERT, and XLM-R. Then, we evaluated the performance after the domain adaptation, as outlined in Subsection 2.2. We observed a substantial increase in performance for each model following domain adaptation. The best-performing model is the XLM-R after domain adaptation, and its metrics are comparable to those observed on the benchmark datasets.

Table 3. Model comparisons on the benchmark datasets

Model	Laptops		Restaurants		Tweets	
	Acc	Macro F1	Acc	Macro F1	Acc	Macro F1
SVM (Kiritchenko et al., 2014)	0.7049	-	0.8016	-	0.6340	0.6330
TD-LSTM (Tang, Qin, Feng, et al., 2016)	0.6810	-	0.7560	-	0.6662	0.6401
ATAE-LSTM (Wang et al., 2016)	0.6870	-	0.7720	-	-	-
MemNet (Tang, Qin, & Liu, 2016)	0.7237	-	0.8032	-	0.6850	0.6691
RAM (Chen et al., 2017)	0.7449	0.7135	0.8023	0.7080	0.6936	0.6730
ANTM+BERT (Mao et al., 2019)	0.7537	0.7189	0.8078	0.7154	0.7176	0.6921
ANTM+$BERT_L$	0.7584	0.7249	0.8249	0.7210	0.7235	0.6945
MNTM	0.7633	0.7170	0.8107	0.7234	0.7341	0.7138

4. CONCLUSIONS

Our primary focus was constructing the MNTM model for the ABSC task that is applicable to texts in various languages and topics. To assess the model's performance in multilingual ABSC, we created the FHT

dataset, comprising tweets posted in the Philippines during the COVID-19 pandemic. We also utilized three benchmark datasets in English to compare our model with existing ones. Our model demonstrated a slight improvement over similar models within the benchmark datasets. Moreover, upon domain adaptation of the XLM-R model, its performance in the FHT dataset is comparable to that observed in the benchmark datasets.

Our study uses a predefined set of aspects, so future work could focus on extending our model to perform both Aspect Term Extraction (ATE) as a joint task (Liu, 2012). Incorporating emoticons and emojis as inputs to the ABSC model may improve classification performance. Additionally, we suggest delving deeper into the effect of various RNNs as controllers. Furthermore, exploring memory networks operating with unbounded time and memory may lead to improvements. Currently, our architecture limits us to $2k$ memory cells and k timesteps to process the input.

Table 4. The performance of various embeddings with the MNTM on the FHT dataset

Embedding model	Accuracy	Macro F1
RoBERTa-Tl (Cruz & Cheng, 2022)	0.6811	0.6610
mBERT (Devlin et al., 2019)	0.6642	0.6515
XLM-R (Conneau et al., 2020)	0.6856	0.6746
RoBERTa-Tl (Adapted)	0.7172	0.6926
mBERT (Adapted)	0.7010	0.6799
XLM-R (Adapted)	0.7401	0.7199

5. REFERENCES

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. 3rd International Conference on Learning Representations, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks, 5 (2), 157–166.

Chan, E. Y., Chan, M. R. C., Ching, S. J. S., Sie, S. L., Lao, A. R., Bernadas, J. M. A. C., & Cheng, C. K. (2022). How Health Information Spreads in Twitter: The Whos and Whats of Philippine TB-data. HEALTHINF, 421–429.

Chen, P., Sun, Z., Bing, L., & Yang, W. (2017). Recurrent attention network on memory for aspect sentiment analysis. Proceedings of the 2017 conference on empirical methods in natural language processing.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzman, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2020). Unsupervised Cross-lingual Representation Learning at Scale. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.

Cruz, J. C. B., & Cheng, C. (2022). Improving Large-scale Language Models and Resources for Filipino. Proceedings of the Thirteenth Language Resources and Evaluation Conference, 6548–6555.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 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).

Dong, L., Wei, F., Tan, C., Tang, D., Zhou, M., & Xu, K. (2014). Adaptive recursive neural network for target-dependent twitter sentiment classification. Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 2: Short papers), 49–54.

Ghani, N. A., Hamid, S., Hashem, I. A. T., & Ahmed, E. (2019). Social media big data analytics: A survey. Computers in Human Behavior, 101, 417–428.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

Graves, A., Wayne, G., & Danihelka, I. (2014). Neural turing machines. arXiv preprint arXiv:1410.5401.

Guzman, J. (2024). Aspect-based sentiment analysis of Filipino COVID-19 tweets with memory networks. Retrieved from https://animorepository.dlsu.edu.ph/etdm_math/10

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9 (8).

Howard, J., & Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 328–339.

Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. 3rd International Conference on Learning Representations, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

Kiritchenko, S., Zhu, X., Cherry, C., & Mohammad, S. (2014). Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014), 437–442.

Liu, B. (2012). Sentiment analysis and opinion mining. Synthesis lectures on human language technologies, 5 (1), 1–167.

Mao, Q., Li, J., Wang, S., Zhang, Y., Peng, H., He, M., & Wang, L. (2019). Aspect-Based Sentiment Classification with Attentive Neural Turing Machines. Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence Main track, 5139–5145.

McHugh, M. L. (2012). Interrater reliability: The kappa statistic. Biochemia medica, 22 (3), 276–282.

Nazir, A., Rao, Y., Wu, L., & Sun, L. (2020). Issues and challenges of aspect-based sentiment analysis: A comprehensive survey. IEEE Transactions on Affective Computing, 13 (2), 845–863.

Pang, B., Lee, L., et al. (2008). Opinion mining and sentiment analysis. Foundations and Trends ® in information retrieval, 2 (1–2), 1–135.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. International conference on machine learning.

Pontiki, M., Galanis, D., Pavlopoulos, J., Papageorgiou, H., Androutsopoulos, I., & Manandhar, S. (2014). SemEval-2014 task 4: Aspect based sentiment analysis. Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014).

Rogers, A., Kovaleva, O., & Rumshisky, A. (2021). A primer in BERTology: What we know about how BERT works. Transactions of the Association for Computational Linguistics, 8, 842–866.

Shazeer, N., & Stern, M. (2018). Adafactor: Adaptive learning rates with sublinear memory cost. International Conference on Machine Learning.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. The journal of machine learning research, 15 (1).

Szolomicka, J., & Kocon, J. (2022). Multiaspectemo: Multilingual and language-agnostic aspect-based sentiment analysis. 2022 IEEE International Conference on Data Mining Workshops (ICDMW).

Tang, D., Qin, B., & Liu, T. (2016). Aspect Level Sentiment Classification with Deep Memory Network. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing.

Tang, D., Qin, B., Feng, X., & Liu, T. (2016). Effective LSTMs for Target-Dependent Sentiment Classification. Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics, 3298–3307.

Turian, J., Ratinov, L., & Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. Proceedings of the 48th annual meeting of the association for computational linguistics, 384–394.

Wang, Y., Huang, M., Zhu, X., & Zhao, L. (2016). Attention-based LSTM for aspect-level sentiment classification. Proceedings of the 2016 conference on empirical methods in natural language processing, 606–615.

Weston, J., Chopra, S., & Bordes, A. (2015). Memory Networks. 3rd International Conference on Learning Representations, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

Zhou, J., Huang, J. X., Chen, Q., Hu, Q. V., Wang, T., & He, L. (2019). Deep learning for aspect-level sentiment classification: Survey, vision, and challenges. IEEE access, 7, 78454–78483.