An Efficient Model for Sentiment Analysis of Electronic Product Reviews in Vietnamese *

Suong N. Hoang¹,², Linh V. Nguyen¹, Tai Huynh¹,², and Vuong T. Pham¹,³

¹ Kyanon Digital, Ho Chi Minh City, Vietnam
{suong.hoang,linh.nguyenviet,tai.huynh,vuong.pham}@kyanon.digital
https://kyanon.digital

² Advosights, Ho Chi Minh City, Vietnam
{suong.hoang,tai.huynh}@advosights.com
https://advosights.com

³ Saigon University, Ho Chi Minh City, Vietnam
vuong.pham@sgu.edu.vn

Abstract. In the past few years, the growth of e-commerce and digital marketing in Vietnam has generated a huge volume of opinionated data. Analyzing those data would provide enterprises with insight for better business decisions. In this work, as part of the Advosights project, we study sentiment analysis of product reviews in Vietnamese. The final solution is based on Self-attention neural networks, a flexible architecture for text classification task with about 90.16% of accuracy in 0.0124 second, a very fast inference time.

Keywords: Vietnamese · sentiment analysis · electronics product review.

1 Introduction

Sentiment analysis aims to analyze human opinions, attitudes, and emotions. It has been applied in various fields of business. For instance, in our current project, Advosights, it is used to measure the impact of new products and ads campaigns through consumer’s responses.

In the past few years, together with the rapid growth of e-commerce and digital marketing in Vietnam, a huge volume of written opinionated data in digital form has been created. As the result, sentiment analysis plays a more critical role in social listening than ever before. So far, human effort is the most common solution for sentiment analysis problems. However, this approach generally does not result in the desired outcomes and speed. Human check and labeling are time consuming and error-prone. Therefore, developing a system that automatically classifies human sentiment is highly essential.

While we can easily find a lot of sentiment analysis researches for English, there are only a few works for Vietnamese. Vietnamese is a unique language

* Supported by Kyanon Digital
and it differs from English in a number of ways. To apply the same techniques that work for English to Vietnamese would yield inaccurate results. This has motivated our systematic study in sentiment analysis for Vietnamese. Since our project, Advosights, initially served a well-known electronics brand, we decided to focus our study on electronic product reviews. Broader scopes will be studied in future works.

Our initial approach was to build a sentiment lexicon dictionary. Its first version was based on some statistical methods [4,5,6] to estimate the sentiment score for each word from a list, collected manually based on Vietnamese dictionaries. This approach did not work well because the dataset came from casual reviews, that were practically spoken language with a lot of slang words and acronyms. This fact made it almost impossible to build a dictionary that cover all of those words. We then tried to use a simple neural network to learn sentiment lexicons from corpus automatically [12]. This also did not work well because some words in Vietnamese have same morphology, but they have different meanings in different contexts. For example, the words “đánh” in two sentences “nhìn đánh quả” and “đánh quả cù” have different meanings. But by using the dictionary, they have the same sentiment score.

Some machine learning-based approaches have been studied. For examples, CountVectorizer and Term Frequency–Inverse Document Frequency (Tf-idf) were used for word representations. Support Vector Machine (SVM) and Naive Bayes were used as classifiers. However, the results were not very encouraging.

We also investigated various types of recurrent neural networks (RNNs) such as long short-term memory (LSTM) [1], Bi-Directional LSTM (biLSTM) [2] or gated recurrent unit (GRU) [9], etc. Although some of them achieved pretty good accuracy, the models were heavy and had very long inference time. Our final model is based on the Self-attention neural network architecture Transformer [16], a well known state of the art technique in machine translation. It provided top accuracy and has very fast inference time when running on real data.

The paper is organized as follows. In section 2, some description of self-attention is provided for motivation. In section 3, our architecture is presented. The experiments are described in section 4. Finally, conclusions and remarks are included in section 5.

2 Background

Inspired by human sight mechanism, Attention was used in the field of visual imaging about 20 years ago [3]. In 2014, a group from Google DeepMind applied Attentions to the RNN for image classification tasks [7]. After that, Bahdanau et al. [8] applied this mechanism to encoder-decoder architectures in machine translation task. It became the first work to apply Attention mechanism to the field of Natural Language Processing (NLP). Since then, Attention became more and more common for the improvement in various NLP tasks based on neural networks such as RNN/CNN [10,11,14,15,19].
In 2017, Vaswani et al. first introduced Self-attention Neural Network [16]. The proposed architecture, Transformer, did not follow the well-known idea of recurrent network. This paper paved the way and Self-attention have become a hot topic in the field of NLP in the last few years. In this section, we describe their approach in detail.

2.1 Attention

The first description of Attention Mechanism in Machine Neural Translation [8] was well known as a process to compute weighted average context vectors for each state of the decoder $s_i$ by incorporating the relevant information from all of the encoder states $h_j$ with the previous decoder hidden state $s_{i-1}$, which is determined by a alignment weights $\alpha_{ij}$ between each encoder state and previous hidden state of the decoder, to predict next state of the decoder. It can be summarized by the following equations:

$$ c_i = \sum_{j=1}^{n} \alpha_{ij} h_j $$

$$ \alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{n} exp(e_{jk})} $$

$$ e_{ij} = a(s_{i-1}, h_j), \text{where } a(s_{i-1}, h_j) \text{ is a function to compute the compatibility score between } s_{i-1} \text{ and } h_j. $$

**Scaled Dot-Product Attention:** Let us consider $s_{i-1}$ as a query vector $q$. And $h_j$ now duplicated, one is key vector $k_j$ and the other is value vector $v_j$ (in current NLP work, the key and value vector are frequently the same, there for $h_j$ can be considered as $k_j$ or $v_j$). The equations outlined above generally look like:

$$ c = \sum_{j=1}^{n} \alpha_j v_j $$

$$ \alpha_j = \frac{exp(e_j)}{\sum_{k=1}^{n} exp(e_k)} $$

In [16] paper, Vaswani et al. using the scaled dot-product function for the compatibility score function

$$ e_j = a(q, k_j) = \frac{qk_j^T}{\sqrt{d_{model}}} $$

where $d_{model}$ is dimension of input vectors or $k$ vector ($q, k, v$ have the same dimension as input embedding vector).
Self-attention: Self-attention is a mechanism to apply Scaled Dot-Product Attention to every token of the sentence for all others. It means for each token, this process will compute a context output that incorporates information of itself and information about how it relates to others tokens in the sentence.

By using a linear feed-forward layer as a transformation to create three vectors (query, key, value) for every token in sentence, then apply the attention mechanism outlined above to get the context matrix. But it seems very slow and takes a bunch of time for whole process. So, instead of creating them individually, we consider $Q$ is a matrix containing all the query vectors $Q = [q_1, q_2, ..., q_n]$, $K$ contains all keys $K = [k_1, k_2, ..., k_n]$, and $V$ contains all values $V = [v_1, v_2, ..., v_n]$. As the result, this process can be done in parallel [16].

$$Attention(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_{model}}} \right)V$$

(6)

Multi-head Attention Instead of performing Self-attention a single time with $(Q, K, V)$ of dimensions $d_{model}$. Multi-head Attention performs attention $h$ times with $(Q, K, V)$ matrices of dimensions $d_{model}/h$, each time for applying Attention, it is called a head. For each head, the $(Q, K, V)$ matrices are uniquely projected with different dimensions $d_q$, $d_k$ and $d_v$ (equal to $d_{model}/h$), then self-attention mechanism is performed to yield an output of the same dimension $d_{model}/h$ [16]. After all, outputs of $h$ heads are concatenated, and apply a linear projection layer once again. This process can be summarized by the following equations:

$$MultiHead(Q, K, V) = \text{Concat}(head_1, head_2, ..., head_h)W^O$$

(7)

where $head_i = Attention(QW^Q_i, KW^K_i, VW^V_i)$

Where the projections are parameter matrices $W^Q_i \in \mathbb{R}^{d_{model} \times d_q}$, $W^K_i \in \mathbb{R}^{d_{model} \times d_k}$, $W^V_i \in \mathbb{R}^{d_{model} \times d_v}$, $W^O \in \mathbb{R}^{hd_v \times d_{model}}$.

2.2 Positional Information Embedding Representation

Self-attention can provide context matrix containing information about how a token relates with the others. However, this attention mechanism still has limit, losing positional information problem. It does not care about the order of tokens. That means outputs of this process is invariant with the same set of tokens with order permutations. So, to make it work, neural networks need to incorporate positional information to the inputs. Sinusoidal Positional Encoding technique is commonly used to solve this problem.

Sinusoidal Position Encoding: This technique was proposed by Vaswani et al. [16]. The main point of this technique is to create Position Encoding
(PE) using sinusoidal and cosinusoidal functions to encode the position. The PE function can be written by the following equation:

\[ PE(position, 2i) = \sin\left(\frac{position}{10000^{2i/d_{model}}}\right) \]  

\[ PE(position, 2i + 1) = \cos\left(\frac{position}{10000^{2i/d_{model}}}\right) \]  

where \( position \) starts from 1 and \( i \) is \( i^{th} \) dimension of \( d_{model} \) dimensions. It means that for each dimension of the positional encoding corresponds to a different sinusoid.

The advantages of this technique is it can add positional information for sentences longer than those in the training dataset.

3 Our Approach

3.1 Model architecture

We proposed a simple model using a single modified 12 heads Self-attention block (See Fig 2), described below.

Original Sinusoidal Position Encoding [16] used “adding” operation to incorporate positional informations as a input. That means while performing Self-attention, representation informations(Word Embeddings) and positional informations(Positional Embeddings) have the same weights (these two information are equal).

\[ z = \text{Embedding} + PE \]  

In Vietnamese, we assumed that the positional information has more contributions to create contextual semantics than representation informations. Therefore, we used “concatenate” operation to incorporate positional informations. That made representation informations may have a different weights with positional informations during the transformation process.

\[ z = \text{Concat(Embedding, PE)} \]  

We added a block inspired by paper “Squeeze-and-Excitation Networks”, Hu et al. [18] for the average attention mechanism and the gating mechanism by stacking a GlobalAveragePooling1D layer then forming a bottleneck with two fully-connected layers (see Fig. 1). The first layer is dimensionality-reduction layer with reduction ratio \( r \) (in our experiment default is 4) with a non-linear activation and then the second layer is dimensionality-increasing layer to return the result to \( d_{model} \) dimension also with a sigmoid activation function, which scale the feature value into range \([0, 1]\). It means this layer computes how much a feature incorporates information to contextual semantics. We call this technique Embedding Feature Attention.
Where \( x \) is input of block, \( y \) is output of block. \( \sigma \) is a non-linear activation function. \( \delta \) is a non-linear activation function. \( W_{fc1}, W_{fc2} \) are trainable matrices.

![Fig. 1. Squeeze-Excitation architecture.](image1)

![Fig. 2. Self-attention Neural Networks architecture for sentiments classification task.](image2)

4 Experiments

We implemented from scratch some layers that are needed for this work, such as: Scaled-dot product Attention, Multihead Attention, Feed-Forward Network and re-trained word embeddings for Vietnamese spoken language.

All experiments were deployed on 26GB RAM, CPU Intel Xeon Processor E31220L v2, GPU Tesla K80 for 20 epochs, 64 of batch size for comparison and all neural network models used focal-crossentropy as the training loss.
4.1 Datasets

There is no public dataset for electronics product reviews in Vietnamese. We had to crawl user reviews from several e-commerce websites, such as Tiki, Lazada, shopee, Sendo, Adayroi, Dienmayxanh, Thegioididong, fptshop, vatgia. Based on our purposes, we chose some data fields to collect and store. Some data samples are presented in Table 1 below.

Table 1. Examples for crawled data from e-commerce websites.

| username | product name | category | review                          | rating |
|----------|--------------|----------|---------------------------------|--------|
| user 1   | Samsung Galaxy A8+ | điện thoại | Ytt5ya 5t55                     | 1/5    |
| user 2   | Philips E181 | điện thoại | đang chơi liên tục nhìn bị đờ đờ rồi tự nhảy lung tung. Bị như vậy là do game hay do máy v mọi người. | 1/5    |
| user 3   | Philips E181 | điện thoại | Đặt màu vàng động mà giao màu bạc | 2/5    |
| user 4   | Oppo f7      | điện thoại | Oppo f7 đang có chương trình trả trước 0% và trả góp 0% đúng không a? | 2/5    |
| user 5   | Philips E181 | điện thoại | Giá đó mà không có camera kép, Vivo V9 đẹp hơn. | 2/5    |
| user 6   | Samsung Note 7 | điện thoại | Cho em hỏi máy móc của em hay bị tắt nguồn là do sao a? | 4/5    |
| user 7   | Nokia 230 Dual SIM | điện thoại | điện thoại vs Máy dùng tốt | 4/5    |
| user 8   | Oppo f7      | điện thoại | cho em hỏi giá oppo F7 hiện tai bên mình là bao nhiêu a? | 5/5    |
| user 9   | Samsung Note 7 | điện thoại | Có màu đen ko vậy? | 5/5    |

After analyzing and visualizing, we found that the dataset was very imbalanced (see the description below) and noisy. There were some meaningless reviews (user 1 in Tab. 1). Some of them did not have sentiments (user 4, user 8 and user 9 in Tab. 1). Sometimes, the ratings do not reflect the sentiment of reviews, (see user 6 in Tab. 1). Therefore, a manual inspection step was applied to clean and label the data. We also built a tool for labeling process to made it smoothly and faster (see Fig. 3).

- Corpus have only 2 labels (positive and negative).
- Total 32,953 documents in labeled corpus:
  - Positives: 22,335 documents.
  - Negatives: 10,618 documents.
Next, to make the dataset balanced, we duplicated some short negative documents and segmented the longer ones. In final result we have over 43,500 documents in corpus with 22,335 positives and 21,236 negatives.

Using for training models, we splitted corpus into 3 sets as following: training set: 27,489, validation set: 6,873, test set: 8,591.

4.2 Preprocessing

For automatic preprocessing, we mainly used available researches. We applied a sentence tokenizer\cite{21} for each documents. All links, phone numbers and email addresses were replaced by “urlObj”, “phonenumObj” and “mailObj”, respectively. Words tokenizer from Underthesea\cite{20} for Vietnamese was also applied.

4.3 Embeddings

We used fastText\cite{13} model for word embeddings. In many cases, users may type a wrong word accidentally or intentionally. fastText deals with this problem very well by encoding at the characters level. When users type wrong or very rare words or out-of-vocabulary words, fastText still can represent those words with an embedding vector that most similar to word met in trained sentences. This has made fastText become the best candidate to represent user inputs.

There had been no fastText pre-trained model for Vietnamese spoken language. Therefore, we trained fastText model for Vietnamese vocabulary as embedding pre-trained weights from a corpus over 70,000 documents of multi-products reviews crawled from ecommerce sites mentioned in subsection 4.1 with no label. Rare words that occur less than 5 times in the vocabulary were removed. Embedding size was 384. After training, we had 5,534 vocabularies in total.
4.4 Evaluation results

We used the same word embeddings as mentioned above for all models and evaluated all models on test set which has 8591 documents. To demonstrate the significance of our model, we compare our model with 6 base line RNNs models such as Long-Short Term Memory (LSTM), Gated Recurrent Units (GRU), bidirectional LSTM, bidirectional GRU, stacked bidirectional LSTM and stacked bidirectional GRU with the following configurations.

- Vanilla LSTM and GRU: 1 layer with 1,024 units.
- Bidirectional model of LSTM and GRU: 1 layer with 1,024 units in forward and 1,024 units in backward.
- Stacked bidirectional model of LSTM and GRU: 2 stacked layers with 1,024 units in forward and 1,024 units in backward for each layer.

Table 2 shows that our model gave the best inference time with top accuracy in test set. Also, in fact, this model ran in production have shown good prediction than the top of baseline models, stacked Bidirectional Long-short Term Memory, especially with complex sentences such as “giá cao như này thì t mua con ss gala S7 cho r”, “quảng cáo lm lố vl”, “với tôi thì trong tầm giá nv vẫn có thể chấp nhận đk” or “Nhưng vì đây là dòng điện thoại giá rẻ, nên cũng k thể kì vọng hơn đc.” (See Fig. 4, Fig. 5)

| Methods         | Avg.inference time (s) | Macro-f1 (%) |
|-----------------|-------------------------|--------------|
| LSTM            | 0.4748                  | 48.9(23)     |
| bi-LSTM         | 0.9373                  | 90.0(05)     |
| stacked bi-LSTM | 1.7967                  | 90.1(32)     |
| GRU             | 0.3738                  | 48.9(23)     |
| bi-GRU          | 0.5863                  | 88.9(25)     |
| stacked bi-GRU  | 1.4830                  | 89.9(72)     |
| **Self-attention** | **0.0124**           | **90.1(64)** |

5 Conclusion

In this paper we demonstrated that using Self-attention Neural Network is faster than previous state of the art techniques with the best result in test set and achieved exceptionally good results when ran in production (Predictions make sense to human in unlabeled data with very fast inference time).

For future work, we plan to extend stacked multi-head self-attention architectures. We are also interested in seeing the behaviour of the models explored in this work on much larger datasets (beyond the electronics product reviews) and more classes.
Fig. 4. Stacked bidirectional Long-Short term memory for Sentiments Analysis in Vietnamese examples

marketing thu trong rat tuyet, trong khi cach tren tai thoi tro ai, qua te.
nhan chung thu su day la dong dt cua nhung thong xuat hien cua hang.
noi chung la k the hao chap nhan due nhung vi vo trong hang dt lan

[ANALYZING...]
+ marketing thu trong rat tuyet. => [POSITIVE]
+ nhung khi cach tren tai thoi tro ai. qua te. => [NEGATIVE]
+ nhan chung thu su day la dong dt cua nhung thong xuat hien cua hang. => [NEGATIVE]
+ noi chung la k the' hao chap nhan duoc nhung vi vo trong hang dt lan

{'NEGATIVE': 3, 'POSITIVE': 1}

choi game cuc ki kho chiu, kho chiu kam khung, men nhin thu bi giet lag.
nhung noi chich xai de' luot web hay facebook thu rat ok.

[ANALYZING...]
+ choi game cuc ki kho chiu, kho chiu kam khung. => [NEGATIVE]
+ nhung khi cach tren tai thoi tro ai. qua te. => [NEGATIVE]
+ nhung khi cach tren tai thoi tro ai. qua te. => [NEGATIVE]
+ nhung khi cach tren tai thoi tro ai. qua te. => [NEGATIVE]
+ noi chung thu su day la dong dt cua nhung thong xuat hien cua hang. => [NEGATIVE]
+ noi chung la k the' hao chap nhan duoc nhung vi vo trong hang dt lan

{'NEGATIVE': 4, 'POSITIVE': 2}

gia co nhu nay thi t xua con ss gala S7 cho r. quang co tan la vl. sieu te

[ANALYZING...]
+ gia co nhu nay thi t xua con ss gala S7 cho r. => [NEGATIVE]
+ quang co tan la vl. => [POSITIVE]
+ sieu te => [NEGATIVE]

{'NEGATIVE': 2, 'POSITIVE': 1}

mean inference times: 1.7967446978953542

Fig. 5. Self-attention Neural Network for Sentiments Analysis in Vietnamese examples
Acknowledgment

We thank our teammates, Tran A. Sang, Cao T. Thanh, and Ha H. Huy for helpful discussions and supports.

References

1. Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. In Neural Computation, 9(8):1735–1780, 1997.
2. Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. In IEEE Transactions on Signal Processing, 45(11):2673–2681, 1997.
3. Werner X. Schneider. An Introduction to “Mechanisms of Visual Attention:A Cognitive Neuroscience Perspective. URL: https://pdfs.semanticscholar.org/b719/918bdf2e71571a3cb2a6aaec3f1b6a9e6.pdf., 1998.
4. Andrea Esuli and Fabrizio Sebastiani. Senti-wordnet: A publicly available lexical resource for opinion mining. In Proceedings of LREC, volume 6, pages 417–422, 2006.
5. Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In Proceedings of LREC, volume 10, pages 2200–2204, 2010.
6. Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In Proceedings of SemEval-2013., 2013.
7. Volodymyr Mnih et al. Recurrent Models of Visual Attention. In Neural Information Processing Systems Conference (NIPS), 2014. arXiv preprint arXiv:1406.6247, 2014.
8. Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. accepted in International Conference on Learning Representations (ICLR), 2015. arXiv preprint arXiv:1409.0473, 2014.
9. Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv preprint arXiv:1412.3555, 2014.
10. Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. Computing Research Repository (CoRR), 2016. arXiv preprint arXiv:1601.06733, 2016.
11. Jiakun Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. Advances in Neural Information Processing Systems 29, pages 289–297, Curran Associates, Inc., 2016.
12. Duy Tin Vo and Yue Zhang. Don’t Count, Predict! An Automatic Approach to Learning Sentiment Lexicons for Short Text. URL:https://www.aclweb.org/anthology/P16-2036, 2016.
13. Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov Enriching Word Vectors with Subword Information. arXiv preprint arXiv:1607.04606, 2016.
14. Filippos Kokkinos and Alexandros Potamianos. Structural attention neural networks for improved sentiment analysis. arXiv preprint arXiv:1701.01811, 2017.
15. Michal Daniluk, Tim Rocktaschel, Johannes Welbl and Sebastian Riedel. Frustratingly short attention spans in neural language modeling. arXiv preprint arXiv:1702.04521, 2017.
16. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 5998–6008, Curran Associates, Inc., 2017. arXiv preprint arXiv:1706.03762, 2017. URL:http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf, 2017.

17. Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. arXiv preprint arXiv:1705.03122, 2017.

18. Jie Hu, Li Shen, Samuel Albanie, Gang Sun, Enhua Wu. Squeeze-and-Excitation Networks. arXiv preprint arXiv:1709.01507, 2017.

19. Yi Zhou, Junying Zhou, Lu Liu, Jiangtao Feng, Haoyuan Peng, and Xiaoqing Zheng. RNN-based sequence-preserved attention for dependency parsing. URL:https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17176, 2018.

20. Vu Anh et al. Underthesea. URL: https://github.com/undertheseanlp/underthesea.

21. Natural Language Toolkit. URL: https://www.nltk.org/.