2023 IEEE 23rd International Symposium on Computational Intelligence and Informatics (CINTI) | 979-8-3503-4294-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/CINTI59972.2023.10381977

Enhancing huBERT Model with Additional Convolutional and BiLSTM Layers for Hungarian Sentiment Analysis

Dávid Maafi
John von Neumann Faculty of Informatics
Obuda University
Budapest, Hungary

Miklós Sipos
John von Neumann Faculty of Informatics
Obuda University
Budapest, Hungary
sipos.miklos@nik.uni-obuda.hu

***Abstract*—This paper explores the enhancement of the huBERT model's classification efficiency by integrating convolutional and BiLSTM layers. The authors train this modified model on categorically labeled user reviews from two websites, assessing performance via a confusion matrix and the F1 score. Results offer insight into the hybrid architecture's efficiency in text classification and provide groundwork for future investigations into improving machine learning methods in language processing.**

***Index Terms*—Natural Language Processing, Text Classification, huBERT, BERT, Convolutional Neural Networks, BiLSTM, Machine Learning, Sentiment Analysis**

I. Introduction

Sentiment analysis, also known as opinion mining, is a natural language processing (NLP) technique used to determine and evaluate the emotional tone, attitude, or sentiment expressed within a piece of text. Its primary purpose is to understand whether the text conveys a positive, negative, or neutral sentiment. [1] Sentiment analysis has gained widespread popularity in both academic research and the business world, largely driven by the growing volume of opinionated text generated by internet users. [2] In this paper, the authors investigate how computers can interpret and distinguish between linguistic sentences based on certain features. To do this, the first step to understand how each machine learning method works, and then use this knowledge to apply to neural networks. Currently, the most popular model is BERT [3], which fortunately now has a pre-learned version in Hungarian called huBERT [4], [5]. BERT can also perform classification, but not very efficiently, so an additional convolution layer and a BiLSTM layer will be added to this model to see how efficient it can be by combining different architectures. User reviews as data are used to train the model, which are scraped from different websites. The evaluations are categorized into 3 groups based on their scores, and the corresponding texts are then cleaned using different filters. After training, the model can classify text objects into these 3 categories. The already trained model undergoes testing using a dedicated portion of our database to assess training effectiveness and model performance. The evaluation employs a confusion matrix and the F1 score, revealing the rate of incorrect outcomes and model performance, respectively. Ultimately, the paper presents a summary of the outcomes and potential avenues for future research.

II. Methodology

A. Database preparation

Initially, data collection for the model is necessary, which can be subsequently utilized for teaching purposes. The challenge lies in obtaining labeled data, and no pre-existing English database was found. Consequently, a custom database was generated by retrieving evaluations from the following websites: www.port.hu/forum and www.maps.google.com.

The process of downloading data involves web scraping. Three distinct libraries were identified for this task: BeautifulSoup, Selenium, and Scrapy. Initially, Selenium was tested, and a scraper was developed, but encountered several issues. Selenium serves as a testing tool that effectively simulates web browsing. When the program is executed, it follows the provided instructions while visually navigating web pages. The first challenge encountered pertained to handling pop-ups such as cookie acceptance prompts and various notifications. Cookies required a one-time acceptance, while notifications were recurrently displayed with each new page load. To address this, timers were implemented to resolve these issues. Subsequently, scraping commenced, but it became apparent that extracting reviews from thousands of pages using this method could take several days. Exploring the other two options, BeautifulSoup initially seemed promising, so development continued here as well. Fortunately, the logic from prior attempts proved applicable, and when dealing with requests, the management of cookies and notifications posed no challenges. Discrepancies emerged during the examination of data across certain websites. Given the limitations of BeautifulSoup in handling these variances, an exploration of Scrapy commenced. In the evaluation, Scrapy emerged as the most versatile scraper among the options considered, facilitating the seamless extraction of ratings from websites.

Utilizing the Google Maps website, the site's API is employed for retrieving ratings. However, a limitation exists

wherein only a maximum of five ratings can be randomly acquired from each website. Detailed documentation on this usage can be found on Google's website. The sole challenge encountered lies in the requirement to specify the search area's center or radius, as narrowing it down to a specific country isn't feasible. Coordinates for major cities were stored, and ratings for each location within a 10km radius were obtained. Ultimately, a complete query wasn't executed due to the associated API costs, necessitating an alternative approach. Discovering an alternate data source was imperative, since the majority of reviews on port.hu consist of positive comments. (Figure 1) Nonetheless, it is crucial for the model to have a relatively balanced input database. Consequently, Amazon [6] was selected, and reviews of English-language books were downloaded.

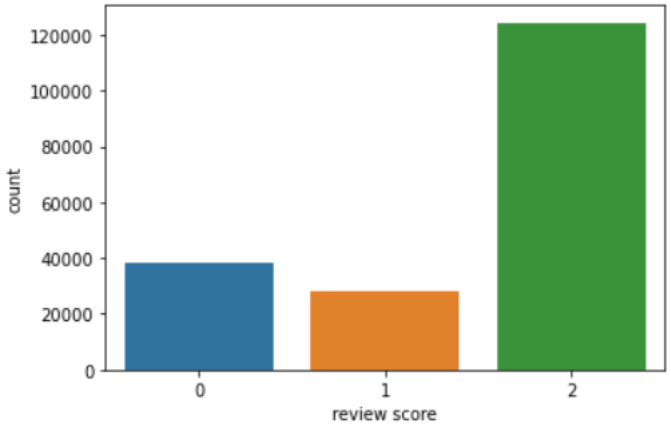

Fig. 1. Distribution of port.hu ratings

B. Data preparation and processing

To facilitate the processing of text and associated scores by our model, the database preparation is essential. The data originates from two distinct sources, necessitating an initial merging process. This merging procedure involved the initial filtration of scores from port.hu, followed by precise numerical identification. During the filtering, consideration was given to various aspects, with paramount importance placed on exclusively examining movie ratings with fewer than 1000 reviews. This choice was informed by previous experience, as instances with a higher count often led to user interactions or spam. Within the examined film dataset, ratings were grouped by users, and only the earliest chronological rating from each user was retained. These resulting reviews predominantly leaned towards the positive spectrum, prompting the substitution of neutral and negative reviews from the Amazon database. Within this database, each review featured a 'vote' attribute, signifying the number of received likes. Utilizing this attribute, reviews were ranked in descending order, and selections were made accordingly. Subsequently, the next phase involved translating nearly 300,000 reviews from English to English using Google's Translate API. This process was expedited through parallelization, resulting in a 16x speed increase. The final step in merging the two databases entailed harmonizing the evaluation scores into a uniform range. Port.com ratings span from 1 to 10, while those on Amazon range from 1 to 5.

Following the merge, the next step involved progressing to text processing. In this stage, the following components of the program needed to be written: extracting usernames, emails, web addresses, HTML tags, special characters, character repetitions, numbers and frequently repeated words, then afterward replacing emoticons with words.

In this context, most problems have been resolved through the application of regex operations. Initially, using regex posed challenges, but proficiency was acquired gradually. The primary obstacle encountered was determining the correct sequence of methods. An illustrative error was removing special characters prior to substituting emoticons with words, resulting in the removal of all emoticons, which contained valuable information for assessment analysis.

The texts must undergo conversion into a format interpretable by the model, employing the WordPiece tokenizer as described in the WordPiece paper [7]. WordPiece operates by analyzing word components through their occurrences in its training dataset. It preserves the most commonly appearing word components in its lexicon and employs this lexicon for the transformation process. To initiate tokenization, it becomes crucial to establish the maximum sentence length, as all sentences will be either extended or shortened to match this predetermined length (Figure 2).

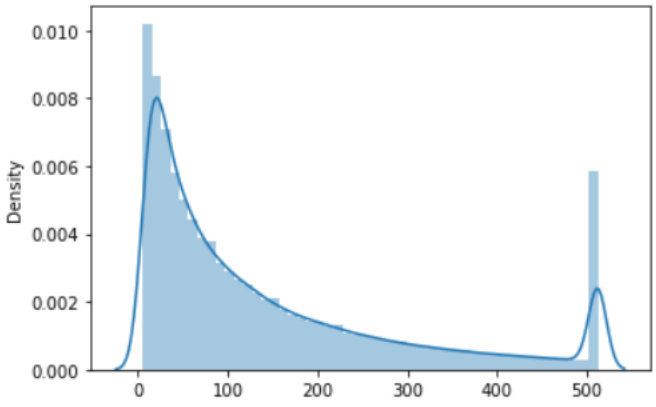

Fig. 2. Distribution of sentence lengths

C. Creating the model

Initially, the process of connecting the layers posed no significant challenges. However, despite error-free layer connections, certainty regarding the impact of actions like matrix permutation on data quality remained elusive. After extensive testing and subsequent research, the model construction was successfully accomplished.

The initial step involves the creation of word embeddings. To mitigate the high computational cost of training BERT, a pre-trained model named huBERT is employed for word embedding generation. This model comprises 12 encoder layers,

producing an output matrix with a defined maximum length of x 768 elements as dictated by the tokenizer. Consequently, the BERT model represents each word in a 768-dimensional space.

Subsequently, the word embeddings are incorporated into three expanded convolutional grids. The advantage of utilizing an expanded convolutional grid lies in its capacity to analyze sentences from various perspectives. By employing a small dilation coefficient, word relationships in close proximity can be extracted, while a larger coefficient allows for the representation of distant sentence relations. Each convolutional layer features 64 filters, a kernel size of 3x768, and dilation coefficients of 1, 2, and 3. The notation 3x768 signifies the examination of three words at a time, extending this process.

Through testing, the quest for a more precise model led to experimentation with dilated kernels and variations in kernel size. (Figure 3) It was determined that employing a dilated kernel yielded superior model accuracy. After obtaining the results from each convolutional layer, the next step involves their concatenation, resulting in a matrix of size n x m. Due to the excessive size and noise in the current matrix, it is passed through a max pooling layer, utilizing a 2x2 kernel. This process yields a matrix of dimensions N/2 x M/2.

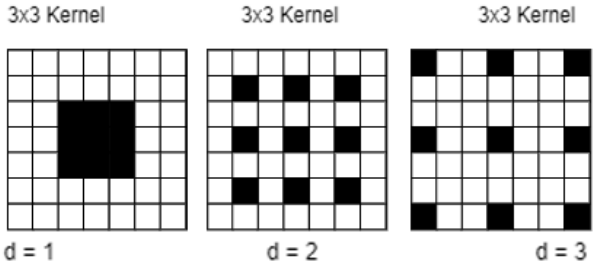

Fig. 3. Dilated 3x3 kernels (left to right: d=1, d=2, d=3)

The matrix proceeds to a bidirectional LSTM layer, offering the benefit of evaluating each vector both from the right and the left. Comprising 128 LSTM units and featuring a dropout rate of 0.2, this layer is characterized by its bidirectional processing. The output from the BiLSTM layer is then flattened to create a one-dimensional vector, which subsequently undergoes processing within a pre-connected neural network.

To enable the model to infer the text's sentiment value, the content of the previously mentioned pre-connected neural network is directed to a fully connected layer employing a sigmoid activation function. This activation function transforms the output into a probability, indicating the likelihood of the text belonging to a specific class. The predicted sentiment value corresponds to the class with the highest probability.

When selecting the optimization function, consideration was given to both SGD and AdamW, ultimately favoring AdamW due to its alignment with BERT. Additionally, a learning rate scheduler was incorporated into the model. This scheduler gradually reduces the learning rate as the number of steps advances, ensuring the optimization process converges efficiently towards the minimum. While this enhances model accuracy, it does extend the learning duration.

D. Training

Defining parameters for the teaching process posed significant challenges. Initially, overly large parameters were selected for the CNN and LSTM layers, resulting in inaccurate teaching and excessive time consumption.

The teaching process extended beyond 30 hours, and validation accuracy, for instance, failed to exceed 34%, ultimately declining further. Notably, the loss rate exhibited minimal change throughout the teaching process. Subsequently, adjusting the parameter values for these two layers led to substantial improvements, which will be elaborated upon in the following section. Utilizing PyTorch Lightning's built-in functionality, learning rates for training were determined, yielding the subsequent outcomes:

Following the adjustment of suitable parameters, the model underwent 8 epochs of training, spanning approximately 28 hours. At the conclusion of each epoch, the training process evaluated the model's accuracy and saved it if improvements were detected. Lastly, a module was crafted for making predictions, drawing upon the preceding details, and the final outcome was stored in an external file. Plotting the data from the file, the following curve can be seen. (Figure 4)

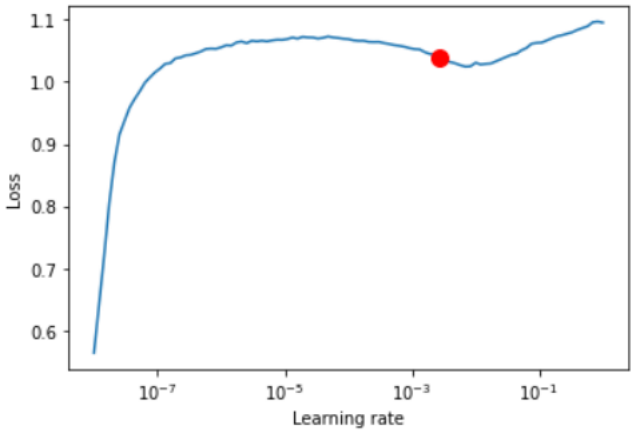

Fig. 4. Optimization of the learning curve

E. Testing

The primary challenge encountered during testing pertained to configuring setup.py correctly. Notably, the process was initiated at the project's conclusion. Following minimal refactoring, reasonable results were achieved in testing the application.

III. Evaluation

Upon completing model training, loss and accuracy values were stored, and a graphical representation was generated.

The graph (Figure 5) illustrates variations in model accuracy across validation and training datasets throughout the training process. Notably, the accuracy consistently increases, as anticipated. Surprisingly, from the outset, validation accuracy

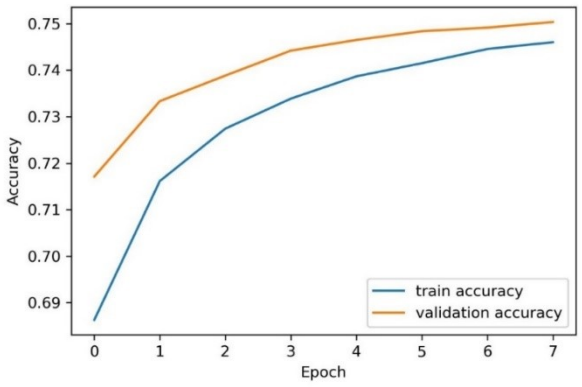

Fig. 5. Accuracy functions

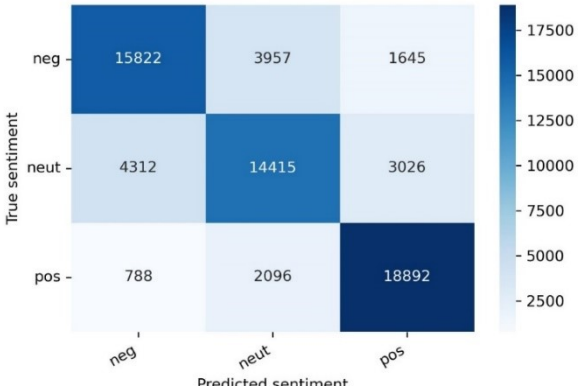

Fig. 7. Confusion matrix

surpasses training accuracy. This outcome can be attributed to a random selection of items within the validation dataset, including a substantial number of positive and negative ratings. This approach prevented the neutral class, which was responsible for most inaccuracies, from dragging down overall accuracy. While it's feasible to extend training on the data, significant improvement is unlikely, as the functions plateau with increasing steps. A comparison of losses is presented in the subsequent figure.

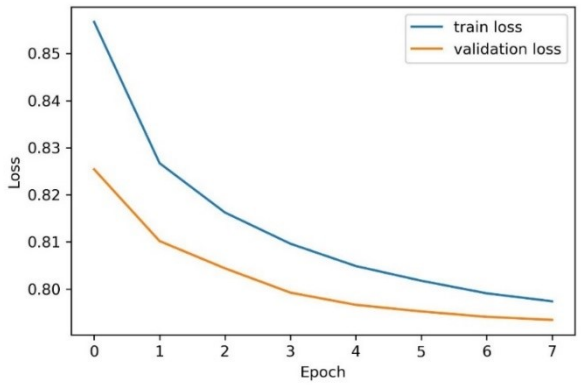

Fig. 6. Loss functions

Once again, the unexpected loss during teaching exceeds that during validation (Figure 6). The rationale for this phenomenon appears to align with the one mentioned earlier. Testing the trained model was subsequently conducted using a dedicated database to assess its actual performance. It's worth emphasizing that elements from the entire database were randomly selected, leading to an unbalanced dataset. In the initial scenario, a comparison was made between the estimated and actual evaluation values, as illustrated in the subsequent figure.

As can be seen in the figure (Figure 7), the model accurately matched the sentiment of the text in most cases. Most of the errors were for neutral ratings, but this is natural, as the human brain has the easiest time classifying extreme texts. After obtaining the counts of successful and unsuccessful matches per class, supplementary methods were employed to assess the model:

Precision: Precision is used to calculate the accuracy of the model. The idea is to check, for each class, how many times the class was hit (Tp), and how many times an item was falsely claimed to belong to the class of that class (Fp), and then calculate it using the equation below:

$$P_{re} = \frac{T_p}{F_p + T_p}$$

Recall: Recall is used to determine how many of the elements belonging to a class have been hit.

$$R_{ec} = \frac{T_p}{F_n + T_p}$$

F1-Score: F1-Score is the so-called harmonic average of Precision and Recall, calculated using the two techniques discussed above.

$$F = \frac{2 * R_{ec} * P_{re}}{R_{ec} + P_{re}}$$

TABLE I
MODEL METRICS

Sentiment	Metrics			
	Precision	Recall	F1-Score	Units
Negative	0.7562	0.7385	7.7472	21424
Neutral	0.7042	0.6626	0.6828	21753
Positive	0.8017	0.8675	0.8333	21776

The metrics show that the accuracy of the model is around 75%, which is acceptable. For English language applications, models with an accuracy of over 95% may be possible, but considering that the complexity of the two languages is very different, it is difficult to compare. The data in Table I would be most useful if several teachings with different parameters could be performed, so that they could be used as a basis for comparison.

IV. Conclusion

There is plenty of scope to improve the model. Initially, the database containing evaluations from port.hu was uploaded. These evaluations might contain numerous typos, necessitating the application of an autocorrect feature (similar to that found in phones) to rectify items unrecognized by the tokenizer. For reviews obtained from the Amazon website, Google's translator was utilized. It's important to note that its translations may exhibit monotonic tendencies. To ensure database diversity, consider using multiple translator applications to mix the data.

The data cleaning process aligns with basic assumptions. The stopword set, however, warrants review. Refinements have been made by eliminating unsuitable entries, but further testing and potential removal could be beneficial. Currently, data cleaning only occurs during the learning process. Regrettably, when users attempt to classify input into categories, this cleaning does not occur, rendering the identification of word relationships more challenging and consequently reducing accuracy.

Further enhancements could include data expansion, such as downloading from various sources. Scrapeable websites encompass Google Plays, Google Maps, moly.hu, and numerous forums that offer additional ratings. Presently, ratings are stored in an Excel file. Transitioning this to a database would be a significant improvement, facilitating easier updates during continuous scraping activities. Currently, the model has only been tested with two parameter sets, but there is room for expansion, allowing for the discovery of optimal parameter values.

References

[1] László Laki and Zijian Yang. "Sentiment Analysis with Neural Models for Hungarian". In: *Acta Polytechnica Hungarica* 20 (Jan. 2023), pp. 109–128. DOI: 10.12700/APH.20.5.2023.5.8.

[2] Mickel Hoang, Oskar Alija Bihorac, and Jacobo Rouces. "Aspect-Based Sentiment Analysis using BERT". In: *Proceedings of the 22nd Nordic Conference on Computational Linguistics*. Turku, Finland: Linköping University Electronic Press, Sept. 2019, pp. 187–196. URL: https://aclanthology.org/W19-6120.

[3] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[4] D. M. Nemeskey. "Natural Language Processing Methods for Language Modeling". PhD thesis. Eötvös Loránd University, 2020.

[5] Osváth Mátyás, Győző Yang Zijian, and Kósa Karolina. "Analyzing Narratives of Patient Experiences: A BERT Topic Modeling Approach". In: *Acta Polytechnica Hungarica* 20 (2023), pp. 153–171. DOI: 10.12700/APH.20.7.2023.7.9.

[6] Ruining He and Julian McAuley. "Ups and Downs". In: *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, Apr. 2016. DOI: 10.1145/2872427.2883037. URL: https://doi.org/10.1145%2F2872427.2883037.

[7] Xinying Song et al. *Fast WordPiece Tokenization*. 2021. arXiv: 2012.15524 [cs.CL].