UDC 004.8

DOi 10.30837/ bi.2021.2(97).03

Yana Daniiel[1], Kostiantyn Onyshchenko[2], N. Kameniuk[3]

[1]Assistant of the Department of Software engineering,
Kharkiv National university of Radio electronics, Kharkiv, Ukraine,
yana.daniiel@nure.ua, ORCID ID: 0000-0002-3895-0744

[2]Assistant of the Department of Software engineering,
Kharkiv National university of Radio electronics, Kharkiv, Ukraine,
kostiantyn.onyshchenko@nure.ua, ORCID ID: 0000-0002-7746-4570

[3]Master's student of Riga Technical University, Latvia, nataliia.kameniuk@gmail.com

USAGE OF LSTM MODELS FOR NATURAL LANGUAGE UNDERSTANDING

The problem of emotion classification is a complex task of language interpretation. In this work, a number of existing solutions for emotional classification problem were considered. The evaluation of performance of the considered models was conducted. The model for emotion classification in three-sentence conversations is proposed in this work. The model is based on smileys and word embeddings with domain specificity in state of art conversations on the Internet. The model performance is evaluated and compared with language processing model BERT. The proposed model is better at classifying emotions than BERT (F1 78 versus 75). However, modern performance of models for language representation did not achieve the human performance due to the complexity of natural language. There is a variety of factors to consider when choosing the word embeddings and training methods to design the model architecture.

NATURAL LANGUAGE PROCESSING, NEURAL NETWORK, NATURAL LANGUAGE

Данієль Я.Д., Онищенко К.Г., Каменюк Н. Використання LSTM-моделей для обробки природної мови. Проблема класифікації емоцій є однією із важливих завдань інтерпретації мови. У даній роботі було розглянуто ряд існуючих рішень проблеми емоційної класифікації. Проведено оцінку продуктивності розглянутих моделей. Запропоновано модель класифікації емоцій у розмовах із трьох речень. Модель заснована на смайлах і використанням слів із оглядом на специфіку сучасного спілкування в в Інтернеті. Продуктивність моделі оцінюється та порівнюється з моделлю обробки мови BERT. Запропонована модель краще класифікує емоції, ніж BERT (F1 78 проти 75). Однак, сучасне виконання моделей мовного представлення не досягло продуктивності людини через складність природної мови. Існує ряд факторів, які слід враховувати при виборі вбудовування слів і методів навчання для проектування архітектури моделі.

ОБРОБКА ПРИРОДНОЇ МОВИ, НЕЙРОННА МЕРЕЖА, ПРИРОДНА МОВА

Даниель Я.Д., Онищенко К.Г., Каменюк Н. Использование LSTM-моделей для обработки есстественного языка. Проблема классификации эмоций является одной из важнейших задач интерпретации языка. В данной работе был рассмотрен ряд существующих решений проблемы эмоциональной классификации. Проведена оценка производительности рассматриваемых моделей. Предложена модель классификации эмоций в разговорах из трех предложений. Модель основана на смайлах и использованием слов с учетом специфики современного общения в Интернете. Производительность модели оценивается и сравнивается с моделью обработки языка BERT. Предлагаемая модель лучше классифицирует эмоции, чем BERT (F1 78 против 75). Однако, современное исполнение моделей речевого представления не достигло производительности человека из-за сложности естественной речи. Есть ряд факторов, которые следует учитывать при выборе встраивания слов и методов обучения для проектирования архитектуры модели.

ОБРАБОТКА ПРИРОДНОГО ЯЗЫКА, НЕЙРОННАЯ СЕТЬ, ПРИРОДНЫЙ ЯЗЫК

Introduction

Over the past few decades, the amount of text data produced by humanity has grown exceedingly. One of the reasons behind this fact is that we actively exchange information and publish our thoughts on websites and social media.

Such unstructured data is represented in arbitrary form and is often complemented by emojis, which makes it difficult for a computer to categorize and derive meaning from the source. On this basis, the challenge to teach computers to properly process this information arose.

Natural language processing (NLP) is widely used for text data analysis and classification [11].

The core aspects of language understanding include three parameters, which are morphology, semantics, and syntax. Morphology is the study of word or statement structure; semantics is the study of meaning, reference, or truth; syntax is the study of how words and morphemes combine to form larger units such as phrases and sentences [3].

Modern models consider all three parameters. The models are using data-driven approach through machine learning and deep learning.

The goal of this work is to consider existing solutions for text data processing in terms of emotional classification and propose the model that can solve such task.

1. Problem statement

The semantic evaluation problem of emotion classification will be considered in this work. The deep learning approach will be based on Keras and TensorFlow – the

Python frameworks. These frameworks have ready to use functions for rapid optimization, model prediction, model tuning and many more. This will allow to achieve approximate state of the art results with an original model architecture. BERT will also be implemented to obtain a current state of the art model in natural language understanding. The received results will be evaluated and compared [4].

The theoretical fundamentals of emotional classification models will be discussed. The practical details of the proposed model's training and BERT on unseen test data will be presented. The obtained results will be evaluated and compared. The differences between the models will be illustrated.

It should be considered that the proposed model will be implemented based on Keras and TensorFlow. The learned models will be applied to a specific pre-defined dataset to solve the semantic evaluation problem of emotion classification. The models will be trained on four CPU cores. These conditions apply specific limitations on the scope of the project.

2. Analysis of existing solutions

1) Machine Learning

Machine learning is a modelling approach focused on finding underlying patterns in a dataset. An algorithm with learning function is applied to rich dataset. The parameters of model are modified to enhance the predictive performance of a model. The unseen data is used to evaluate the trained model.

a) 3.1.1 RNN

Recurrent Neural Network (RNN) is a class of artificial neural networks, where connections between nodes form a directed sequential graph. By this, the sequential nature of input can be considered when making output predictions [12].

A set of feedback weights contained in a hidden state vector is computed at every step in the sequence that pass information from earlier time points. The ability to predict to which class the sequence belongs to is provided by the model reformulation. This will allow to incorporate new time dependency by using the final recurrent hidden state vector to make softmax probability predictions.

$$\hat{y} = \varphi\left(Vh_p\right), \tag{1}$$

where

$$h_p = \sigma\left(Ux_p + Wh_{p-1}\right). \tag{2}$$

The parametrization W of the model is comprised of the weight matrices U, V and W. The hidden layer is dependent on earlier states.

The Fig. 1 provides more information about the recurrence mechanism applied.

New hidden state vector is computed at each time step. The vector is trained to pass forward the most significant information for solving the problem through gradient descent with an appropriate loss function [12].

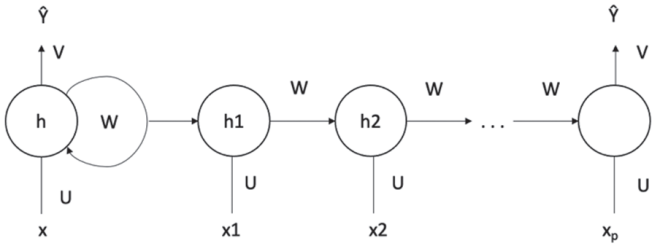

Fig. 1. The unfolding concept

b) LSTM

Long Short-Term Memory (LSTM) networks have two enhancements comparing to previously considered RNNs. The first enhancement is that each time step passes a hidden state vector and a local context vector to the next recurrent node. The second enhancement is that long short-term memory network contains a set of gating mechanisms (Fig. 2). These mechanisms provide the ability to the model to decide which data to pass forward in recurrence. These enhancements allow the LSTM model to learn long-term dependencies in the sequence more stably.

The gating mechanisms of LSTM contain an input gate, a context gate, forgetting gate and an output gate, which are activated matrix manipulations. The manipulations are based on gating weights optimally learned through training. The following relationships define the gates and their associated weights [1].

$$f_p = \sigma(x_p U^f + h_{p-1} W^f), \tag{3}$$

$$i_p = \sigma(x_p U^i + h_{p-1} W^i), \tag{4}$$

$$o_p = \sigma(x_p U^o + h_{p-1} W^o), \tag{5}$$

The functions are recurrent to hidden state of the previous time step and the current input data at this time.

A candidate $c_p^{\sim}$ for the context state $c_p^{\sim}$ is computed by

$$c_p^{\sim} = \tanh(x_p U^c + h_{p-1} W^c). \tag{6}$$

The previous context information filtered by the forgetting gate and present information from input gate in an equation filtered through the context gate form the context state.

The gating machinery provides an ability to determine which long-term and short-term data to filter and pass to the final output representation [10].

The hidden state representation of the sequence h_{p-1} is computed as a combination of the filtered output from the output gate and current context information.

$$h_p = o_p \tanh(c_p). \tag{7}$$

The received output state can be used for prediction. This output can be used in the same way as the hidden states were used in RNN architecture equation (1) considered earlier.

On this basis, LSTM can represent complex sequences and make stable predictions.

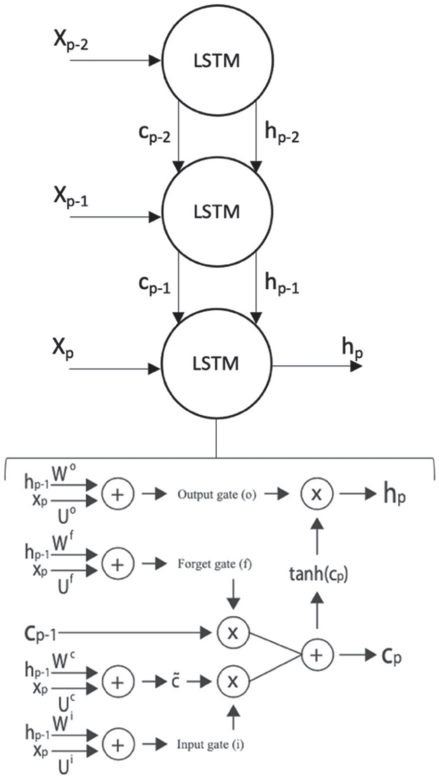

Fig. 2. The LSTM node

c) Bidirectionality

It should be considered that not all words can be predicted by the words before them.

In the sequences like “bow tie” and “bow ring”, “bow” is contextually dependent both on the word sequences before and after them. This is a challenge for regular recurrent networks. The solution for this challenge is bidirectionality, which means reversing direction of the sequence and feeding it to network. Both resulting hidden states are concatenated, which is a standard practice for many language models [10].

In Fig. 3, the simple RNN from Fig. 1 is extended to work bidirectionally.

The states h in Fig. 3 can be from regular nodes or LSTM. The network with the reverted states is a copy from the original network. The resulting hidden states are concatenated into form

$$h_p = (h^{\rightarrow}_p, h_1^{\leftarrow}) \quad (8)$$

The final hidden state vector can be used in the same way as for RNN through equation (8). The bidirectional models can be modelled by doubling the number of weights. This method is often applied to improve model representation and predictions due to its ability to add contextual information to language models [5].

2) Vector Representation of Language

Machine Learning models represented above have real valued vectors xi. The language is represented as vectors which is an essential step in using neural networks as emotional classifiers. The words w are gathered in a vocabulary to form the bag of words [2].

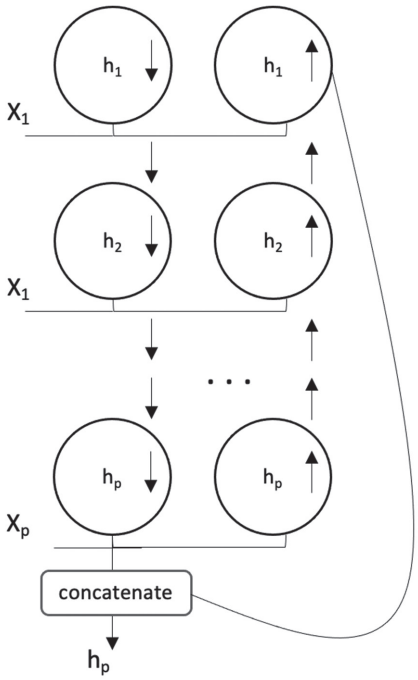

Fig. 3. Bi-directional RNN

$$V = \{w_i : i \in 1, ..., N\} \quad (8)$$

The phrase is divided into one-word hot vectors, which do not provide any extra information about the context. The vectors are also computationally hard to use as a single vector is a single word.

a) Word Embeddings

A more efficient solution is to pass a lower dimensional vector $x_i \in Rd$ ($d<N$) which also contains language information about the word wi. Less computational power is required to process such vector by a classificational model. This vector contains more language information than one-word hot vector. This type of lower dimensional representation is called word embedding.

Statistical language modelling is the other language feature. The words are used in conjunction with each other. This can get a lot of information about the semantic and syntactical use of a word through the context [9].

Principal Component Analysis (PCA) is a classic dimensional reduction technique, which is based on singular value decomposition of the co-occurrence matrix. This approach is does not need linguistic rules to be fed to the system, which makes it unsupervised. It considers the text corpus in a whole, in different context, which also makes it computationally expensive.

This can be solved by using a feed forward neural network to predict the n-gram probabilities of the words in the vocabulary given the context that came before [9].

Trainable d-dimensional random initialized vector represents each word and n previous words are used as a context. These vectors are concatenated and fed through several hidden layers. The output layer is a softmax probability over the vocabulary of all the probabilities to meet each word by the context words. The network is learning

linguistically valuable information in the matrix of d-dimensional vectors while training to predict n-grams.

On this basis, words embedding can be used as the sole input to other machine learning models for real NLP tasks, such as machine translation, and semantic sentence parsing. The Fig.4 illustrates a scheme of such feed forward neural network.

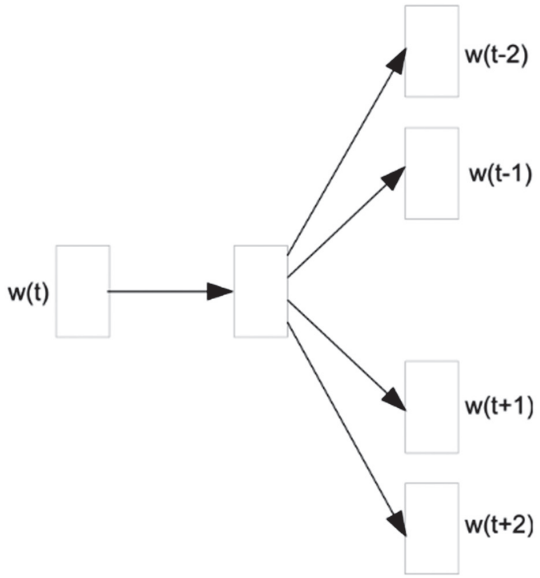

Fig. 4. Bi-directional RNN

It should be considered that this model is expensive to train. To increase the context window for the network to learn, there is a need to increase the number of input nodes, and the task of emotional classification requires a lot of context.

3) GloVe

The skip-gram model is another proposed RNN approach which is efficient on smaller datasets with better context information. RNNs efficiently model rich context information due to the fact they have a sequence memory of the previously met words. The model starts with the opposite probability comparing to the n-gram prediction, because this is the probability of different context words. The basis of this probability is a target word [6].

GloVe is the other unsupervised learning algorithm for obtaining vector representations for words. The difference of GloVe from the skip-gram and feed forward word embedding is the fact that GloVe is trained on a loss function. The loss function considers both local co-occurrences from the n-length context windows and global count-based co-occurrence probabilities from the text corpus. This allows to encode more of the language features comparing to PCA. This ability is provided since including only local context information does not give enough information to features about the frequency the words occur in rare contexts [7]. The loss function has slightly richer context information embedded in the vectors comparing to skip-gram model.

4) BERT

Pre-trained word embeddings in language are easy to use and flexible. Their drawback is that one vector can represent only one word. This means homonyms or phrasal verbs, like ‘key’ or ‘get’ lose their language information. This can be mitigated by using RNN or LSTM to carry information over sequences to create a context for a single word, but these networks cannot provide rich context data to predict context-heavy language constructions.

One of the proposed approaches to solve this issue is BERT (Bi-Directional Representations from Transformers). In its core, the transformer relies on attention for its representation to improve the contextual awareness [2].

The weighted representation of all hidden state vectors is trained at the same time with the recurrence relationships. This allows the network to be aware of the previous hidden states and do not rely solely on the final hidden state representation when making predictions.

The transformers apply attention to underlying word vectors from the original sequence to extract relevant language features. This is called text encoding. Since transformers do not use any recurrent relations, this allows to conduct training in parallel [2].

On this basis, BERT model is applicable to solve NLP tasks. BERT pre-trained weights can be downloaded via the Internet and used for transfer learning. This will allow to compare BERT with the other models in the task of emotion classification.

a) Structure of the model

BERT is a sequence-to-sequence language representation model, which is efficient for NLP tasks solution. A sequence of language information X = (I0, . . . , In) as inputs and outputs a contextualized vector representation H = (h0, . . . , hn) of the elements of the input sequence.

During the pre-processing phase, the model splits words into word-parts. The placeholder tags are inserted before the sequence and after sentences, however this does not allow input vectors to directly correspond to the underlying words in the sequence.

Due to the design of BERT model, the output presentation h0 becomes a distributed representation of the underlying sequence. This sequence is similar to the final hidden state of RNN. A classification model can be obtained when adding an extra hidden layer to the BERT model and activating this model with a softmax function [11].

$$\hat{y} = \varphi\left(Vh_0\right) \tag{9}$$

Encoders, which are stacking nodes, are the other part of this language representation framework. Encoders are used to create encoded text representation (Fig. 5).

Every encoder layer abstracts language patterns from an input sequence. It allows to form more complicated patterns as the information flows up the layers. The first encoder layer L receives language inputs and the last

encoder layer outputs the final encoded language information [11].

$$H_1 = Encoder(X)$$
$$\ldots \quad (10)$$
$$H_L = Encoder\,(H_{L-1})$$

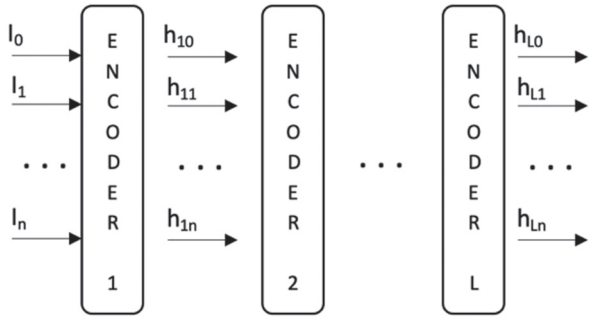

Fig. 5. Layers of the model

The first vector H of the final H representation is the basis for classification task. Two sub-processes form each encoder layer. As the first step, the inputs to the encoder are passed through a multi-head self-attention layer. This layer uses a series of matrix manipulations to extract language information from the data inputs. The definition of this process is multi-head self-attention (Fig. 6).

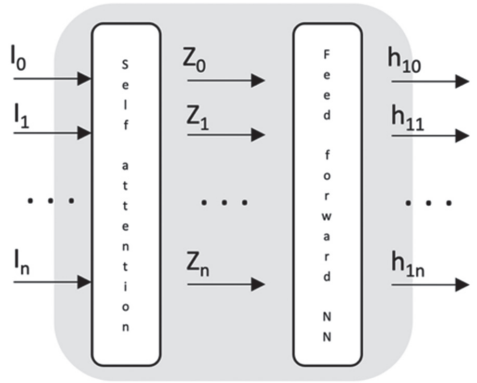

Fig. 6. The encoder dissection

The second sub-layer receives the outputs of the first layer after these outputs are residually connected and normalized. The second sub-layer retrieves the most valuable information from the attention layer. This data is residually connected and normalized again. After this, the outputs are sent onwards to the following encoder layer.

3. Theoretical propositions

This section is dedicated to proposition of a model for classifying the last phrase of three-turn conversations into any of pre-defined emotions. These emotions are Happy, Angry, Sad and Others. The context is defined by previous two sentences. The datasets are classified beforehand by the emotion they represent, since model training should be performed on high-quality data.

This task is solved by creation of neural network architecture. BERT is used as a state of art language representation model. The proposed model and BERT are compared in terms of their performance on the problem.

Table 1

Data sample

id	Turn 1	Turn 2	Turn 3	
112	It's cool meme	Ha-ha, yes	☺	Happy
154	I don't think it is a good idea	You better think twice	No, I won't!	Angry
186	In my hometown there are many chestnuts	Where is it?	In Kyiv	Others

The pre-processing is divided into two steps. The first one is data cleaning, which stands for defining and fixing data irregularities. The second one is tokenization, which stands for mathematical mapping of text data into a suitable input for classification models.

Data cleaning provides an ability to clarify meaning in the text and boost the coverage. Coverage is measured by pre-trained embeddings as the number of words for which pre-trained embeddings exist divided by the total amount of unique words in the problem. Since our dictionary is problem dependent and GloVe embedding matrix is generalizable, it is not certain that the embedding matrix has a vector representation of a given word. Thus, data cleaning is recurrently applied to boost coverage.

In our problem, data cleaning is used to transform abbreviations into their initial state and replace symbol emojis with Unicode emojis. This allows to improve sentence representation in terms of meaning.

After data cleaning, tokenization is applied. The words are vectorized and prepared for passing into classification models. The words are separated and mapped to unique numbers to decode them later. Keras, which is used to solve the problem, has this functionality built into the library.

As a result, tokenized texts are mapped with embedding matrices which represents the sequences in vector space. The vector represented sequences are passed to LSTM models. The input must all be of equal length to allow to train the model weights to recognize features at the same places in different sequences. To solve this, the vectors are zero padded to the same length, which is constrained to 100 words per sentence.

The Figure 7 illustrates the word length distribution is the training data.

Each sentence in a conversation is a matrix with elements of padded vectors representing the sentences. The corresponding labels are one-hot vectors. The index of the element 2 represents what emotion the training example is connected to. The indexes are as follows: 0 – Angry, 1 – Sad, 2 – Happy, 3 – Others. The dataset is ready for processing.

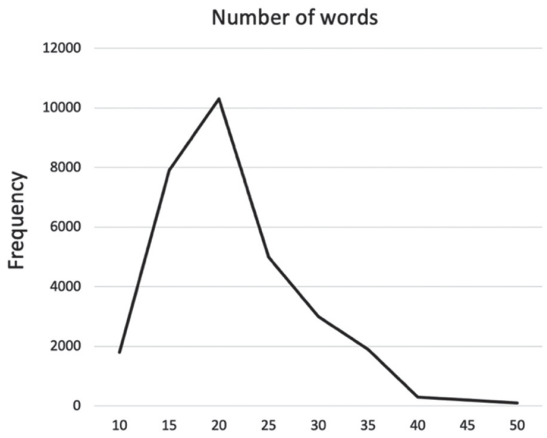

Fig. 7. Word lengths per sentence

1) Proposed model

The proposed model (Fig. 8) is built as follows. The standard LSTM outputs straight to a prediction layer and is filtered through a softmax function. The result is a probability vector. We use GloVe embeddings, trained on text corpus represented by Wikipedia data. The word embeddings of dimension 100 and LSTM layer of dimension 128 with built in dropout are used by the model.

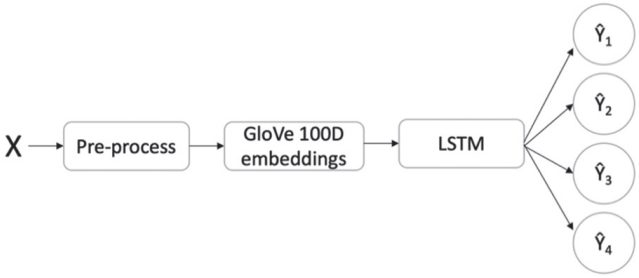

Fig. 8. The proposed model architecture

The model passes several improvements, which allow to define features with high impact on model performance. This impact is quantitively represented by F1 score. Such features are isolated and used to create simple networks.

The first improvement is to separate the phrases of the conversation and pass each one to LSTM nodes separately. Thus, the model is provided by a clearer vision of sentence differences.

The second improvement is to extract emojis and smileys from text for further separate processing. The emojis are represented with Emoji2Vec embeddings, which provides new language information to the model. Emoji2Vec is a 300-dimensional representation of Unicode smileys. Further concatenation of information contained in emojis with the data from the rest of the model is passed through a final dense layer. This allows the model to weight the relative importance of smileys in classifying the emotion. The rectified linear unit is the activation function in this dense layer.

BERT is used due to its applicability in wide range of problems. The model is hosted on TensorFlow hub. TensorFlow is a Google pre-trained machine learning repository.

It provides an ability to download an implementation which has to do with the following: the number of encoded layers, the number of heads in the multi-head attention, the number of dense layers in the feed forward network and the number of used training samples.

On the contrary to GloVe embeddings, BERT uses word-piece embeddings and follows different pre-processing and tokenizing which can be downloaded with the ready to use pre-trained model. This allows to not to perform pre-processing of the sentences but directly pass sentences to BERT.

4. Experimental results

The table 2 illustrates the comparative micro-averaged F1 results for the models on emotion classification. The performance of the models was considered by evaluation metric on Happy, Angry and Sad emotions. The model performance on emotion class Others was ignored in this evaluation.

The LSTM model achieved overall micro-averaged F1 as 0.616. This model required minimal data pre-processing and GloVe word embeddings. The word embeddings were trained on Wikipedia data. It can be seen in the table, that the model achieved the worst results on Happy conversations and the best results on Angry conversations. The shortcoming of the model is the inability to distinguish the Happy, Angry and Sad emotion classes from the emotion class Others.

Table 2

F_1 scores

Model	Emotion F_1			Micro F_1
	Happy	Sad	Happy	
LSTM	0.523	0.601	0.724	0.616
BERT	0.687	0.799	0.771	0.752
SS-LSTM	0.556	0.**818**	0.784	0.719
Proposed Model	**0.784**	0.767	**0.811**	0.787

BERT model achieved better results in macro F1 comparing to LSTM due to presence of extra context information. The Angry emotion class was predicted as 0.771 F1 points. The Sad emotion class was also predicted better in F1 points comparing to LSTM. The proposed model F1 scores are shown in the table. The model has shown the best results in Angry and Happy emotion class.

The confusion matrix of the proposed model is shown in table 3. The matrix illustrates the distribution between the emotion labels in the test dataset. The most confusion comes from distinguishing the Others emotion class from Happy, Angry and Sad emotion classes. Angry and Sad emotion classes are never evaluated by the model as Happy and vice versa. The confusion between predicting Angry and Sad labels is rarely present. On this basis, the

focus should be put on prediction improvement between Others emotion class and the other three emotion classes.

Table 3

Confusion Matrix

TRUE	Predicted			
	Others	Happy	Sad	Angry
Others	4103	101	80	94
Happy	63	215	4	1
Sad	40	0	212	8
Angry	46	0	6	334

The idea of whether the early stopping effectiveness on outfitting prevention is got by looking at the loss function over the training steps. The decrease of training loss is present over the training steps. The validation loss is decreasing by reaching the minimum after three epochs and rising afterwards (Fig. 9).

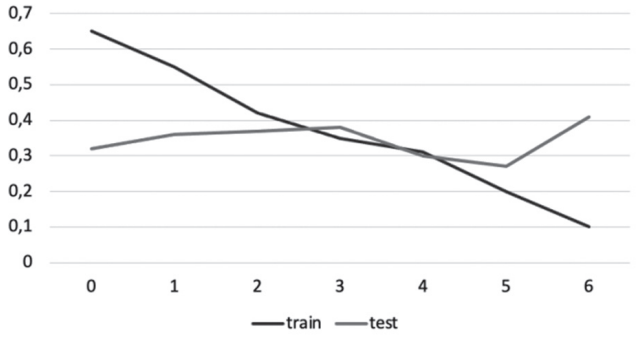

Fig. 9. Training and Validation Loss

The explanation behind this is the different distribution of emotions in the training dataset and validation dataset. This can be seen in table 4. The loss from the training set updates the gradients. The gradient updates change the validation loss in other ways. The amount of data points in the validation set is fewer comparing to the training set, which contributes to higher variance in the effects of updating the loss. On this basis, it is important to use early stopping after 3 epochs since the improvements in training loss are not translated to improvements in validation loss after this epoch.

Table 4

Emotion Class Distribution

	Others	Happy	Sad	Angry	Size
Train	.441	.98	.211	.194	29884
Validation	.793	.045	0.054	0.052	2467
Test	.882	0.039	0.061	0.044	5804

Conclusions

The problem of emotion classification is a complex task of language interpretation. The existing generic solutions for text data processing were considered in this work.

The evaluation of performance of the considered models was conducted. The proposed model shows better results comparing to generalized model with transfer learning.

There are many decisions applicable for solving the emotion classification problem. There is a variety of factors to consider when choosing the word embeddings and training methods to design the model architecture.

To date, machine learning models did not achieve the human performance in terms of language representation and emotion classification. The confusion is present in labeling complex sentence structures. Such nuances are explained by the language structure and its dynamic nature.

However, the area of capturing the language nuances is constantly evolving and new approaches are created every day.

References:

[1] *Alex Graves and Jürgen Schmidhuber*. Framewise phoneme classification with bidirec- tional lstm and other neural network architectures. Neural Networks, 18(5):602 – 610, 2005.

[2] *Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov.* Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15:1929–1958, 2014.

[3] *Y. Daniiel, K. Onyshchenko*. Implementation of Recursive Deep Learning Algorithms for Natural Language Processing. Information Systems and Technologies 2021, 2021.

[4] *I. Afanasieva, N. Golian, O. Hnatenko, Y. Daniiel, K. Onyshchenko*. Data exchange model in the internet of things concept. Telecommunications and Radio Engineering 78 (10), 2019.

[5] *Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa*. Natural language processing (almost) from scratch. CoRR, 2011.

[6] *Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov.* Bag of tricks for efficient text classification. CoRR, 2016.

[7] *Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita*. A survey of the usages of deep learning in natural language processing. CoRR, 2018.

[8] *James Allen*. Natural Language Understanding (2Nd Ed.). Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1995.

[9] *Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal*. A Sentiment-and-Semantics-Based Approach for Emotion Detection in Textual Conversa- tions. 2017.

[10] *Daniel Jurafsky and James H. Martin*. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

[11] *Christopher D.* Manning and Hinrich Schütze. Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA, USA, 1999.

The article was delivered to editorial stuff on the 27.09.2021