УДК 004:62-52:004.853 https://doi.org/10.15407/pp2022.03-04.289

FEATURES OF BUILDING RECOMMENDATION SYSTEMS BASED ON NEURAL NETWORK TECHNOLOGY USING MULTITHREADING

Nataliia Komleva, Svitlana Zinovatna, Vira Liubchenko, Oleksandr Komlevoi

У статті розглянуто питання створення прикладного програмного забезпечення з використанням багатопотоковості. Розроблено рекомендаційну систему, призначену для надання рекомендацій туристам щодо готелів. Для створення рекомендацій побудовано модель оцінювання готелів, що описуються об'єктивними та суб'єктивними аспектами. До об'єктивних аспектів відносяться характеристики готелів, які можуть бути представлені у вигляді бінарного значення (наявність чи відсутність) чи кількісної оцінки характеристик готелю (кількість поверхів, площина тощо). Серед суб'єктивних характеристик найціннішими є текстові відгуки туристів, які аналізуються з визначенням певного емоційного забарвлення; відгук може бути класифікований як позитивний, нейтральний чи негативний. Розробку рекомендаційної системи виконано з використанням мови програмування Python та супутніх бібліотек.

Ключові слова: рекомендаційна система, нейронна мережа, паралельні обчислення, інженерія програмного забезпечення.

The article is devoted to the creation of a recommendation system for tourists regarding hotels using a neural network based on a multilayer perceptron. The work uses the mechanism of parallelization of the training sample of the neural network. To check the quality of the provided recommendations, the average absolute and root mean square errors, accuracy and completeness were used. The results of the experiments showed that when analyzing 10 html pages with descriptions of hotels, the metrics of root mean square error and accuracy gave the best results at 500,000 epochs of neural network training when using 8 processors.

Keywords: recommendation system, neural network, parallel computing, software engineering.

Introduction

In recent decades, the interest of many developers and users has been focused on automating the performance of various tasks, regardless of which subject area they belong to. Collecting and pre-processing information, calculating and effectively presenting results, performing a chain of logical conclusions, converting data formats – all this and much more is done with the help of automated software tools.

Recommender systems are designed to increase user satisfaction by providing helpful recommendations for various entities – goods, movies, audio, books, news content, promotions, etc. Of course, in this case, the recommender system should have available data describing the user and the entities to which the recommendations are projected [1, 2]. Content-oriented filtering, collaborative filtering, or clustering methods are usually used when creating a recommender system.

One of the classes of content analysis methods, the work of which can be automated and organized in such a way as to increase awareness of the level of opinions about particular objects or processes, is the analysis of the tonality of the text. The tonality of any text is understood as the kind of emotional coloring of this text, which shows the author's attitude to some event, object, or process. The tonality analysis is based on analyzing emotionally colored vocabulary and its components [3].

Using sentiment analysis methods allows you to solve such tasks as determining the author's emotional state during the creation of the text and the author's relationship to a particular object mentioned in the text. At the same time, theoretical definitions of the characteristics of sentiment analysis have quite specific practical applications in practice [4].

The most common practical example of encouraging sentiment analysis methods is evaluating the quality of services and/or certain goods based on textual user reviews. The importance of such information received by the user about a particular product or service varies depending on the degree of importance of the service or product to the user. Therefore, in the case of the need to make a complex decision for the user, it is advisable to obtain recommendations and reference information. At the same time, the sharp increase in the number of goods and services, as well as the people who use them, leads to the impossibility of manual processing of data arrays. Therefore, such processes should be automated.

Purpose and tasks

The work aims at studying the peculiarities of building a recommender system created using a neural network based on a multilayer perceptron, using multithreading, as well as analyzing the speed of its operation depending on the dimension of parallelization.

To achieve this aim, the following main tasks of the research are defined:

– to analyze the latest studies and publications in which the peculiarities of the development of recommender systems are given;

- build an evaluation model for the object described by objective and subjective features;
- among text vectorization algorithms, choose an algorithm for determining the emotional coloring of text responses;
- determine the data for training a convolutional neural network, formalize the process of determining the tonality of responses using multi-threading;
- to consider the application of the recommendation system using the example of creating recommendations for choosing a hotel, detailing the requirements for the functional characteristics of the corresponding software application, and designing the system architecture;
- perform a comparison of the training duration of a neural network on one and several processors;
- discuss the research results and conclude the conducted research's effectiveness.

Analysis of literary sources and problem formulation

The work [5] presented a recommender system that calculates product tonality scores based on different levels of analysis using hybrid deep learning techniques to determine polarity-based tonality characteristics. But the study used mainly subjective features such as customers' feedback and ratings; only the product cost is used from objective features.

The authors of [6] described the individual methods used to evaluate the review using sentiment analysis and found that to overcome the shortcomings of the particular techniques, it is possible to combine them to improve the effectiveness of sentiment classification. However, it is indicated that the proposed methods have performance problems, which should be solved in the future.

In [7], there was introduced sentiment analysis based on recursive neural networks using deep learning to optimize recommendations based on sentiment analysis performed on different reviews taken from various social networking sites; however, the narrow task of recommending places that are close to the current location is solved location of the user, a significant part of the attention is focused on solving the spatial task.

The work [8] defined a methodology for calculating the weighted sentiment value using the Sentiment Intensity Analyzer from the NTLK library, but no objective factors are used. In addition, parallelization of calculations is not proposed to increase the speed of obtaining results.

The task of sentiment analysis (tonality analysis) consists of three stages:

- preliminary analysis of textual data;
- conversion of text into a valid feature space;
- tonality classification using machine learning methods.

Preliminary analysis of textual data means:

- removal of stop words;
- segmentation;
- bringing words to a single norm;
- marking parts of speech.

Two methods are most often used to convert text into a valid feature space:

- bag of words [9];
- Word2Vec [10].

Both models use methods based on statistical information about the words of the text. This approach creates a vector with a length equal to the number of words used in all analyzed texts for each object.

In the last step of sentiment analysis, the most suitable machine learning method is selected and applied. It classifies and determines whether the text message reflects a positive, neutral, or adverse opinion. There are many classification methods. The most common classification methods are the support vectors, gradient boosting, naive Bayes, etc.

The process of deep learning can be divided into the training process and the inference process. We should point out that a single node often cannot meet its performance requirement in large-scale deep neural network training. Therefore, the training process often is designed in parallel nodes. The increase in the number of layers of the neural network and the complexity of the algorithm model has brought challenges to the parallelization of deep learning [11].

Formalization of the process of building a recommendation system

For the assessment of a complex object, the application of an aspect-oriented approach is proposed. In general, the description of such an object can consist of objective O and subjective S features. The characteristics of objective features are measured on a binary (true/false or present/absent) or quantitative (numeric types) scale.

In general, the evaluation model is represented by sets of evaluations of objective and subjective features:

$$H = (\ <Co_i \subseteq O,\ Cs_j \subseteq S>\ ), \tag{1}$$

where Co_i represents a set of evaluations of objective features, i=1,...,N, N is a number of objective features; Cs_j represents a set of assessments of subjective features, j=1,...,M, M is a number of subjective features.

Fig. 1 shows a generalized scheme for building of recommender system for complex objects. As you can see, the first stages are studying the evaluation object, analyzing its components, and internal relationships. The result of

the analysis is a set of assessment features. When building an assessment model, features are divided into objective and subjective features, and a set of acceptable assessments is determined.

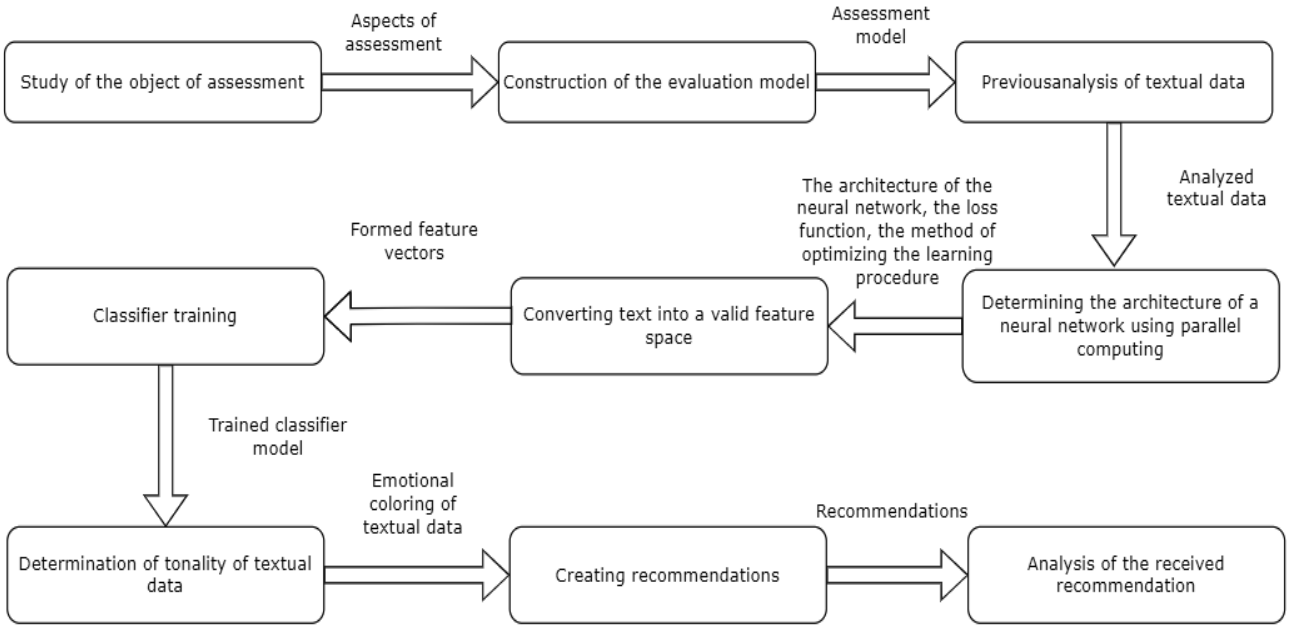

Figure 1. Scheme of recommender system building

Next, a preliminary analysis of text data is performed – reviews, messages, descriptions, etc. In this processing, marked data, i.e., marked as positive or negative, and dictionaries are fed to the input of the block. Dictionaries are used to normalize words further and remove stop words. Despite being significant in human communication, stop words are words that do not provide helpful information when analyzing the text's tonality.

A neural network transforms the processed text data into feature space and determines tonality. The neural network architecture determines the topology of connections between neurons, the number of layers and neurons in each layer, the learning method, etc. Multithreaded calculations are proposed to optimize the neural network training procedure. The number of processors involved in calculations affects the way of control, information transfer between individual neurons, and their synchronization.

For the text to be submitted to the classifier's input, it needs to be vectorized. Depending on the chosen technology, the text is matched with a set of signs presented in the form of numbers during vectorization. The following algorithms can be used for vectorization: Topicmodeling, GloVe, One-hot encoding, SVD, FastText, and others.

But the most widespread is the Word2vec algorithm, which provides an opportunity to determine the degree of closeness of the word values of the analyzed text depending on the context of the words. To assess the degree of proximity of words represented by vectors, cosine similarity is used, which for two vectors A and B is calculated according to the standard formula:

$$Similarity = \frac{(A,B)}{|A||B|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (2)$$

During the training of the neural network, the task of the Word2vec algorithm is to maximize the cosine similarity between the vectors of those words that are in contexts that are similar in meaning and, conversely, to minimize the cosine distance between words that are not located next to each other in the context.

The neural network of the Word2vec algorithm is trained using the backpropagation method; that is, the weights of the hidden layer and then the input layer are adjusted. The result of Word2vec is vector coordinates for certain words.

The loss function describes how far the neural network model is from making perfect recommendation predictions for the given data. If so, check the quality of the provided recommendations, the recommended forecast predicted, and the actual estimate that was issued was calculated; then we have the following statistical metrics:

$$MAE = \frac{1}{n}\sum |\, predicted - actual\,|, \quad (3)$$

$$RMSE = \sqrt{\frac{1}{n}\sum (predicted - actual)^2}, \quad (4)$$

where n is a number of recommended objects. The values of the *MAE* and *RMSE* metrics reflect the accuracy of the recommendations provided to the user.

Metrics for evaluating the decisions made should also include Precision and Recall. Both are based on the so-called contingency table, which contains four values: TP (true-positive) reflects the correctly recommended entities; TN (true-negative) reflects the correctly not recommended entities; FP (false-positive) reflects the entities recommended to the user, but he does not need it; FN (false-negative) reflects the entities not recommended to the user but was required by him.

Precision and Recall values are calculated as:

$$Precision = \frac{TP}{TP + FP}, \tag{5}$$

$$Recall = \frac{TP}{TP + FN}. \tag{6}$$

Sentiment analysis using multithreading

Quite often, convolutional neural networks are used to analyze the tonality of texts [12]. Initially, convolutional neural networks were used for image recognition, but later they began to be actively used in prediction, classification, and modification tasks.

The architecture of the convolutional neural network is unidirectional (only the direct direction of propagation of the activation signals); the activation functions are selected at the user's request. A neural network has many layers. The backpropagation method is used for training [13].

The idea of parallel computing is based on the fact that most tasks can be divided into a set of smaller tasks that can be solved simultaneously. Usually, parallel computations require the coordination of actions. At the same time, several threads can run in parallel and not interfere with each other. When solving problems with the help of neural networks, parallelization can be carried out in different ways: at the level of the learning phase, with the distribution of the training sample, at the level of the layer of the neural network, at the level of the neurons themselves or their weights. These methods can be used both individually and in combination.

For any neural network, the most resource-intensive task is the stage of its training. Often, the training sample size for a problem solved using neural networks can be large enough. In a single-stream system, training vectors will be sent to the network one at a time. In a parallel system, these training vectors can be divided between several processors, and the number of processors can be adjusted. Each processor requires a full copy of the neural network. The peculiarity of this learning method is that different values are applied to the input of neural networks, thereby changing the response results. At the same time, after their correction, the scales are immediately sent to other streams, after which the results are obtained.

Thus, processors can learn simultaneously on different training samples (Fig. 2).

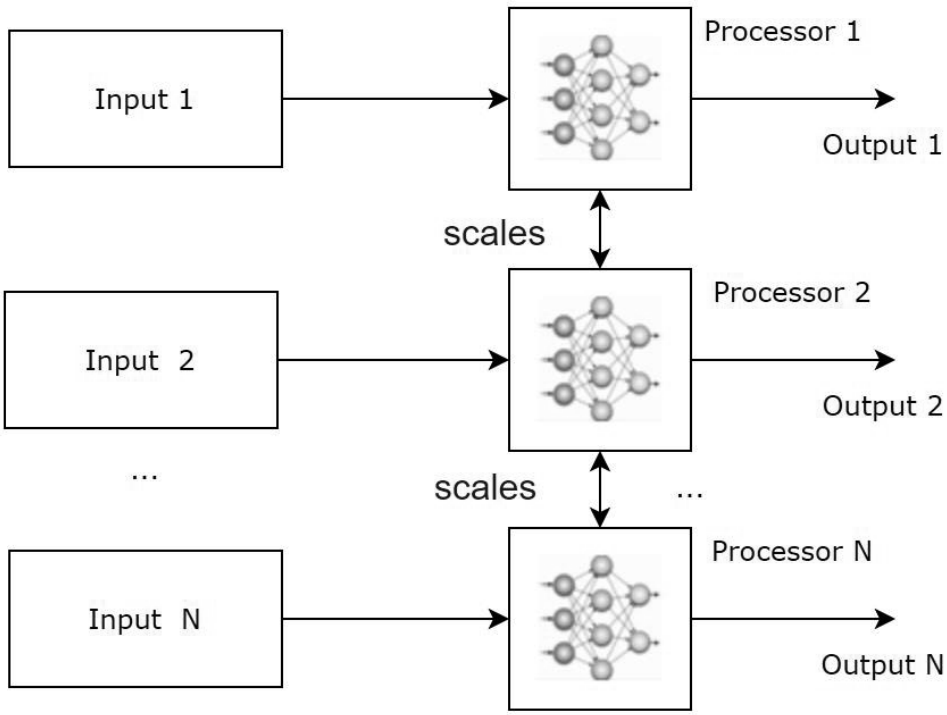

Figure 2. Parallelization of the training process

A difficult task for building a neural network architecture is determining the number of internal layers. Adding too many layers causes the network to forget the gradient and dramatically increases the number of weights needed to train the network. The problem of forgetting can be solved using activation functions of the ReLU (Rectified Linear Unit) family of functions. The issue of increasing the number of weights for training leads to an increase in training time and the possibility of retraining the network model. To solve this problem, a specific correlation between the values of the elements of the vectorized text is used, which allows not to create a fully connected neural network and, thus, reduces the number of weights used for training.

In the general case, the dictionary size can reach many words. The algorithm's operation will require a lot of time because the backpropagation method involves the calculation of the gradient in two steps. That is why it is necessary to choose algorithmic and software techniques, which include ways to optimize the word vectorization process for the possibility of quickly using large dictionaries (more than several hundred thousand words).

Results and discussion

A hotel is selected as the object for which a recommendation is issued. The recommendation system should help the tourist choose the most attractive hotel based on his request. To implement the proposed model, a software system was created, which includes subsystems for implementing business requirements and making relevant recommendations for tourists regarding hotels (Fig. 3). As you can see, the software system contains two significant parts: a subsystem for implementing business requirements (Fig. 4) and a subsystem for creating relevant recommendations (Fig. 5).

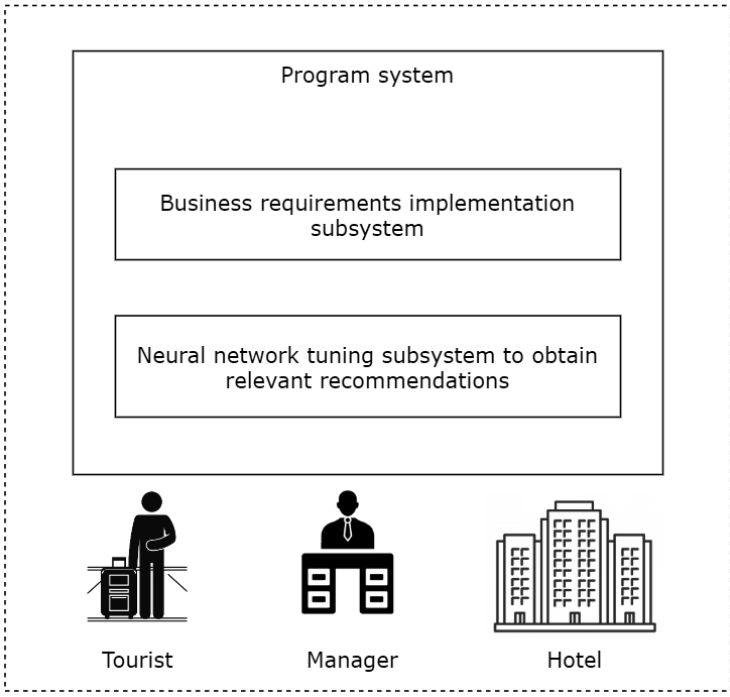

Figure 3 – Generalized structure of the software system

According to the aspect-oriented approach, when evaluating objective features, their characteristics are considered, which can be presented in the form of:

– logical value true/false (presence or absence) for a particular component of the hotel: restaurant, children's entertainment room, swimming pool, etc.;

– numerical values: the playground size, number of floors, depth of children's pool, etc.

The evaluations of subjective features are the evaluations tourists give on a particular scale. In addition, the most valuable assessments of subjective features are the textual feedback of users, which are analyzed with the determination of a certain emotional color of the feedback.

In this work, supervised learning is used to determine the emotional coloring of responses; that is, the labeled data is used for classifier building.

When developing software, the method and convenience of implementing machine learning functions and the possibility of creating parallel data processing processes depend on the chosen programming language. This paper proposes the use of the Python programming language. Program modules implemented in Python can conveniently use the free spaCy library, which specializes in implementing natural language text processing algorithms. The spaCy library provides functions for analyzing text tonality based on applying a convolutional neural network. Since the system for recommending hotels must also consider the reviews of tourists, the spaCy library is useful.

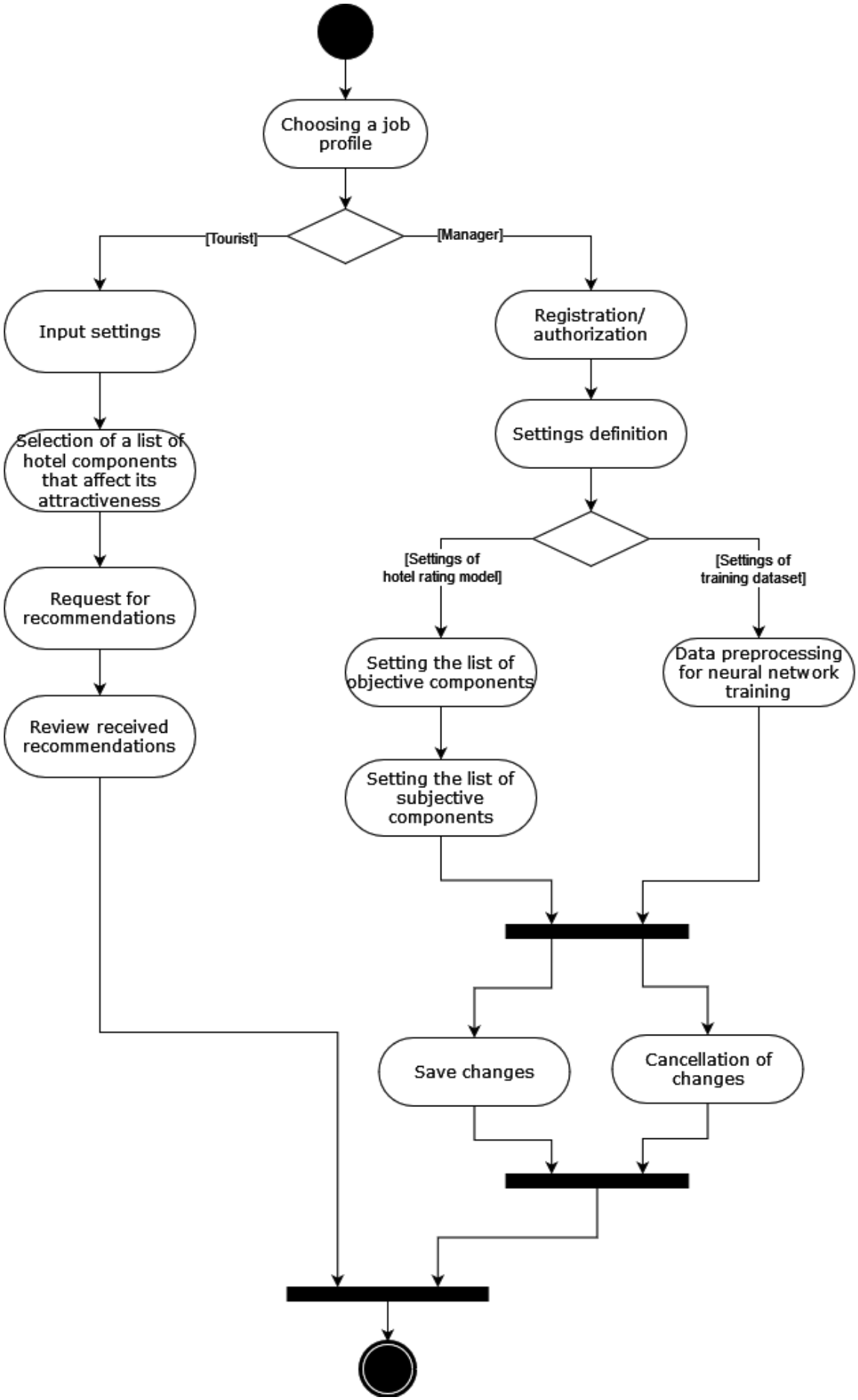

Figure 4. Activity diagram of the business requirements implementation subsystem

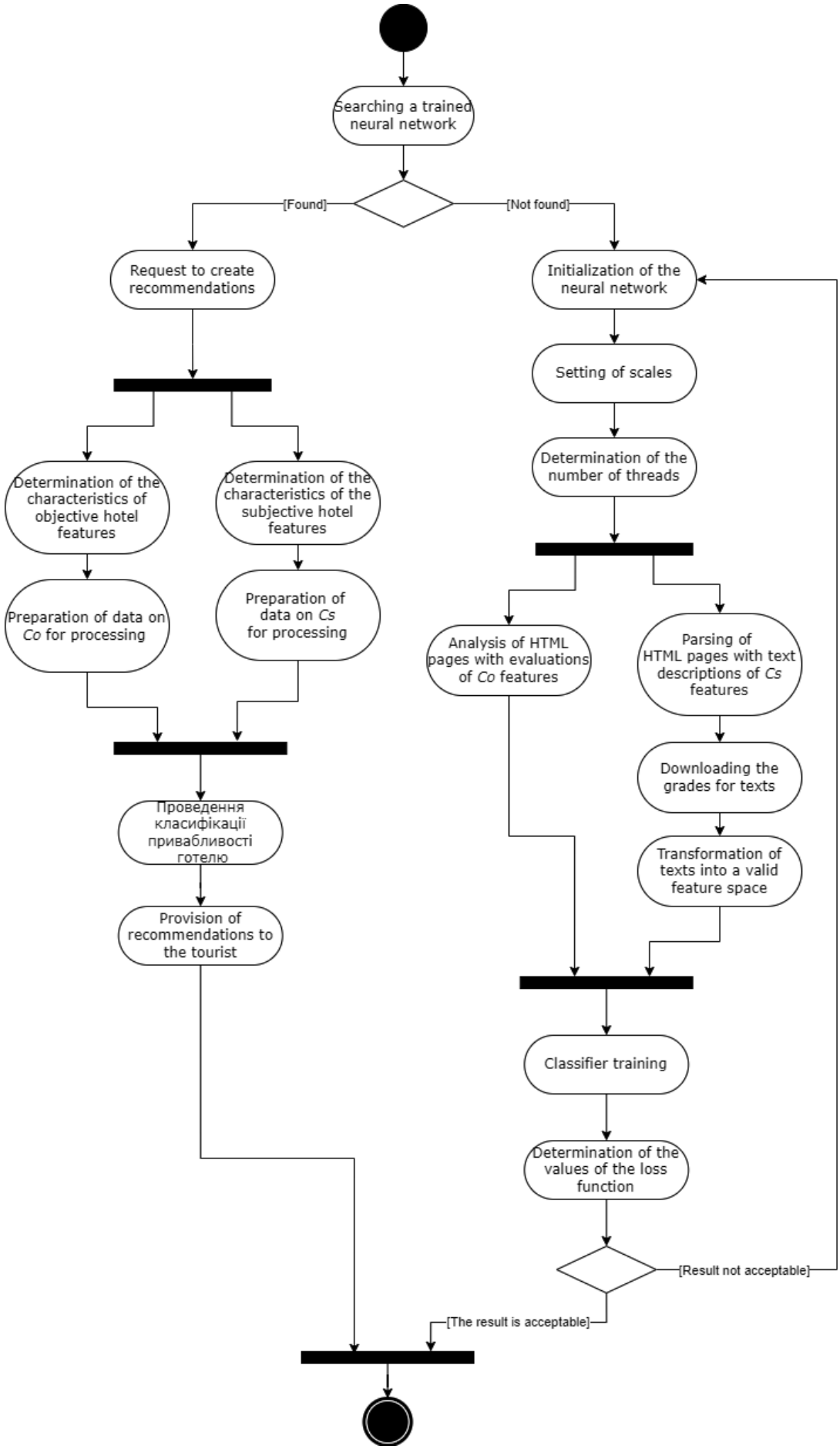

Figure 5. Activity diagram of the subsystem for creating relevant recommendations

Fig. 6 shows spaCy's built-in loader for a pre-trained statistical language model to handle English text. Using the NLP (Natural Language Processing) method allows text tokenization.

```
In [2]: import spacy
        nlp = spacy.load("en_core_web_sm")

        text = """
        I booked this hotel because the price was reasonable and there was a pool.
        After reading a few negative reviews, I thought it would not matter
        to me as I am quite calm and we planned not to be in the hotel most
        of the time. However, the problems were not long in coming.
        I booked bed and breakfast, but the lady at the front desk said
        that breakfast was not included. Since she was not very helpful,
        I had to figure it out on my own. The indoor pool smelled and looked
        like it hadn't been cleaned since it opened! The air conditioner
        leaked when we left it on, and every time there was a puddle
        of water on the floor. Hot water pressure practically did not exist.
        I had to ask twice to be looked at. Needless to say, I was very upset.
        Wouldn't recommend staying there.
        """

        doc = nlp(text)
        token_list = [token for token in doc]

        print(token_list)

[
, I, booked, this, hotel, because, the, price, was, reasonable, and, there, was, a, pool, .,
, After, reading, a, few, negative, reviews, ,, I, thought, it, would, not, matter,
, to, me, as, I, am, quite, calm, and, we, planned, not, to, be, in, the, hotel, most,
, of, the, time, ., However, ,, the, problems, were, not, long, in, coming, .,
, I, booked, bed, and, breakfast, ,, but, the, lady, at, the, front, desk, said,
, that, breakfast, was, not, included, ., Since, she, was, not, very, helpful, ,,
, I, had, to, figure, it, out, on, my, own, ., The, indoor, pool, smelled, and, looked,
, like, it, had, n't, been, cleaned, since, it, opened, !, The, air, conditioner,
, leaked, when, we, left, it, on, ,, and, every, time, there, was, a, puddle,
, of, water, on, the, floor, ., Hot, water, pressure, practically, did, not, exist, .,
, I, had, to, ask, twice, to, be, looked, at, ., Needless, to, say, ,, I, was, very, upset, .,
, Would, n't, recommend, staying, there, .,
]
```

Figure 6. An example of tokenization of hotel feedback

When spaCy tools split the response into tokens using NLP, it receives a Doc object consisting of a set of Token class objects and other information. The `token.is_stop` construct is used to identify and remove stop words from tokenized responses. After that, spaCy tools reduce the tokens remaining in the feedback to their original form using a standard lemmatization procedure. The spaCy tools come with a default list of stop words. This list can be adjusted if necessary.

The next stage is the vectorization of text responses with the formation of feature vectors. The Word2Vec method is used for this. Note that the Word2vec algorithm has been known for a long time, so there is no need for the program implementation of a neural network for the vectorization of text words. The lemmatized tokens are transformed into unique numerical values; vectorization is performed and calculated by vector for each token. In the spaCy library, the vectors are dense, i.e., zero empty values are not created, which allows you to speed up the processing of a non-sparse array. For vectorization, the `nlp()` method is used, which calculates a vector using the vector attribute and determines the vector partition coefficients for testing. By default, 80% of the data is used for training, and 20% for testing and checking the quality of the classification result. Next, the classifier is trained. As mentioned above, a convolutional neural network is used for this. The parallelization of the training sample is provided by the appropriate APIs of the Python programming language multiprocessing package. After that, the marked text and its corresponding labels are loaded using the `load_training_data` method. The data is shuffled and split into two sets – a training set and a test set – and these sets are then returned as the result of the method. The application of the error backpropagation method is used to train a multilayer perceptron using an iterative gradient algorithm. It allows you to reduce the errors of the neural network and obtain satisfactory classification results.

Next, the datasets are loaded into the list, and the directory structure of the data files is created. A content tuple is then added to the list of hotel reviews and, in addition, a dictionary of tags (a requirement of the spaCy model format during training).

The trained model of the classifier is used to determine the tonality of the response, for this, the functions of the spaCy library are also used.

The classification results are evaluated according to the defined classification quality assessment measures. Classification accuracy is traditionally determined.

A program class diagram (Fig. 7) has been developed for the proposed system, which contains classes: `DataPreparation`, `MainClassifier`, `ReviewClassification`, `ClassificationNode`, `NetTrainer`, `View`, `HotelValueGather`, `HotelReviewsGather`, `WebPageParser`. The `ReviewClassifica-`

`tion` class is responsible for classifying hotels. It inherits from the `MainClassifier` class and uses the `NetTrainer` class to train the neural network. The `ClassificationNode` class is used to process neural network nodes. The `DataPreparation` class helps you configure input data for recommendations. The `View` class implements the user interface. The `HotelValueGather` and `HotelReviewsGather` classes analyze hotel feature ratings and summarize recommendations for them. The `WebPageParser` class is responsible for parsing HTML pages with hotel descriptions.

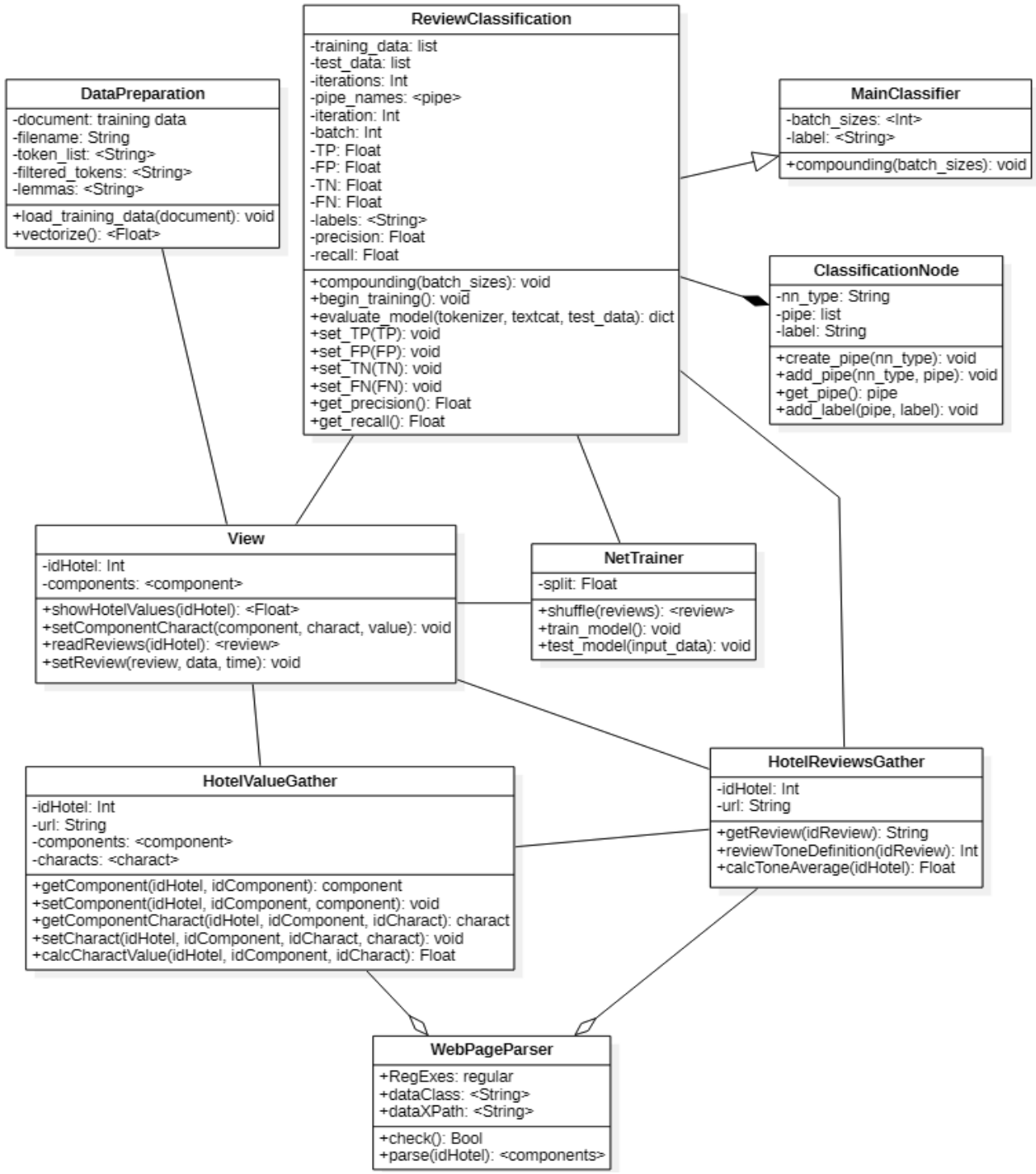

Figure 7. The software classes diagram

During the research, different scenarios were used, which differed in the number of epochs of neural network training: 10,000 (scenario 1), 100,000 (scenario 2), and 500,000 (scenario 3) epochs.

The results of neural network training and testing obtained using 1, 2, 4, and 8 processors are shown in Table 1. As can be seen, there is no significant improvement in results between training the neural network on one and two processors because the gain from organizing parallel computations on two processors is spent on synchronizing the gradients between processors before each update of the neurons of the neural network.

But when switching to 4 and 8 processors, all metrics show a gain for checking the quality of received recommendations. Thus, with 500,000 learning epochs of the neural network, calculations on 8 processors compared to sequential calculations allowed to increase the value of MAE by 0.22%, RMSE by 9.65%, Precision by 8.33%, and Recall by 2.83%.

Table 1. Dependencies between parallelization and values of execution time, MAE (3), RMSE (4), Precision (5), and Recall (6)

Scenarios	Execution time (seconds) on processors			
	1	2	4	8
Scenario 1	4,92	5.34	3.91	2.28
Scenario 2	49,26	53.38	39.07	25.01
Scenario 3	243,15	260.94	190.12	129.04
Scenarios	MAE,%			
	1	2	4	8
Scenario 1	0,873	0,875	0,880	0,883
Scenario 2	0,886	0,888	0,889	0,892
Scenario 3	0,892	0,890	0,891	0,894
Scenarios	RMSE,%			
	1	2	4	8
Scenario 1	0,804	0,816	0,846	0,903
Scenario 2	0,825	0,832	0,852	0,914
Scenario 3	0,839	0,841	0,869	0,920
Scenarios	Precision,%			
	1	2	4	8
Scenario 1	0,716	0,718	0,727	0,792
Scenario 2	0,752	0,749	0,756	0,836
Scenario 3	0,804	0,811	0,823	0,871
Scenarios	Recall,%			
	1	2	4	8
Scenario 1	0,704	0,703	0,726	0,738
Scenario 2	0,723	0,725	0,732	0,749
Scenario 3	0,741	0,744	0,751	0,762

Conclusions and prospects for further research

The paper presents the results of studying the peculiarities of using multi-threading when building a recommender system using a neural network based on a multilayer perceptron. Modern publications in the work domain were analyzed, and best practices were determined. Further, these best practices were applied in developing a hotel recommendation software system. The software implementation was performed using multithreading, which made it possible to conduct an experimental study of the speed of its operation depending on the parallelization dimension. Quantitative measurements showed that parallelization reduces the execution time, which corresponds to theoretical assumptions. Also, improved performance leads to improved model training, which is confirmed by the quantitative values of the model performance metrics.

Although the quantitative indicators summarized in Table 1 are obtained on representative samples, the experiment is set for one recommender system. It can be assumed that the behavior of the meters will not change depending on the subject area for which the recommendation system was developed. Therefore, the following research step will be the study of the impact on the recommender system's quality of using different neural network structures.

References

1. Yadav N. et al. (2020) Diversity in Recommendation System: Cluster Based Approach. *Hybrid Intelligent Systems*. 1179, pp. 113–122. Available from: https://doi.org/10.1007/978-3-030-49336-3_12. [Accessed 28/07/2022].
2. Bag, S., Abhijeet, G. & Manoj, K.T. (2019) An integrated recommender system for improved accuracy and aggregate diversity. *Computers Industrial Engineering*. 130, pp. 187–197. Available from: https://doi.org/https://doi.org/10.1016/j.cie.2019.02.028. [Accessed 28/07/2022].
3. Balshetwar, S.V., Tuganayat, R.M. & Regulwar, G. (2019) Frame Tone and Sentiment Analysis. *International Journal of Computer Sciences and Engineering*. 7, pp. 24–40. Available from: https://doi.org/10.26438/ijcse/v7i6.2440 [Accessed 28/07/2022].
4. Wankhade, M., Rao, A.C.S. & Kulkarni, C. (2022) A survey on sentiment analysis methods, applications, and challenges. *Artif Intell Rev*. Available from: https://doi.org/10.1007/s10462-022-10144-1 [Accessed 28/07/2022].
5. Raviya, K. & Vennila, M. (2022) An Approach for Recommender System Based on Multilevel Sentiment Analysis Using Hybrid Deep Learning Models. *8th International Conference on Smart Structures and Systems (ICSSS).* pp. 01–06. Available from: https://doi.org/10.1109/ICSSS54381.2022.9782172 [Accessed 28/07/2022].
6. Banker, S. (2016) A Brief Review of Sentiment Analysis Methods. *International Journal of Information Sciences and Techniques*. 6(1/2), pp. 89–95. Available from: https://doi.org/10.5121/ijist.2016.6210 [Accessed 28/07/2022].
7. Preethi, G. et al. (2017) Application of Deep Learning to Sentiment Analysis for recommender system on cloud. *International Conference on Computer, Information and Telecommunication Systems (CITS)*. pp. 93–97. Available from: https://doi.org/10.1109/CITS.2017.8035341. [Accessed 28/07/2022].
8. Mishra, R. K., Urolagin, S. & Jothi, J. A. A. (2020) Sentiment Analysis for POI Recommender Systems. *Seventh International Conference on Information Technology Trends (ITT)*. pp. 174–179. Available from: https://doi.org/10.1109/ITT51279.2020.9320885 [Accessed 28/07/2022]
9. Qader, W., M. Ameen, M., & Ahmed, B. (2019) An Overview of Bag of Words; Importance, Implementation, Applications, and Challenges. pp. 200–204. Available from: https://doi.org/10.1109/IEC47844.2019.8950616 [Accessed 28/07/2022].
10. Li, S. & Gong, B. (2021) Word embedding and text classification based on deep learning methods. *MATEC Web of Conferences PY*. 336. Available from: https://doi.org/10.1051/matecconf/202133606022 [Accessed 28/07/2022].
11. Hu, W. & Huang, F. (2020) Review of Deep Learning Parallelization and Its Application in Spatial Data Mining. *2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*. pp. 110–115. Available from: https://doi.org/10.1109/ICBAIE49996.2020.00029 [Accessed 28/07/2022].
12. Jacovi, A, Sar Shalom, O. & Goldberg, Y. (2018) Understanding Convolutional Neural Networks for Text Classification. *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP.* pp. 56–65. Available from: https://doi.org/10.18653/v1/W18-5408 [Accessed 28/07/2022].
13. Leung H. & Haykin S. (1991) The complex backpropagation algorithm. *IEEE Transactions on Signal Processing*, 39(9), pp. 2101–2104. Available from: https://doi.org/ 10.1109/78.134446 [Accessed 28/07/2022].

Received 03.08.2022

Про авторів:

Комлева Наталія Олегівна,
кандидат технічних наук,
доцент кафедри Інженерії програмного забезпечення
Кількість публікацій у вітчизняних виданнях: 38.
Кількість публікацій у закордонних виданнях: 4.
http://orcid.org/0000-0001-9627-8530.

Зіноватна Світлана Леонідівна,
кандидат технічних наук,
доцент кафедри Інженерії програмного забезпечення
Кількість публікацій у вітчизняних виданнях: 28.
Кількість публікацій у закордонних виданнях: 2.
http://orcid.org/0000-0002-9190-6486.

Любченко Віра Вікторівна,
доктор технічних наук,
професор кафедри Інженерії програмного забезпечення
Кількість публікацій у вітчизняних виданнях: 52.
Кількість публікацій у закордонних виданнях: 5.
http://orcid.org/0000-0002-4611-7832.

Комлевой Олександр Миколайович,
кандидат біологічних наук,
доцент кафедри клінічної імунології, генетики і медичної біології
Кількість публікацій у вітчизняних виданнях: 21.
Кількість публікацій у закордонних виданнях: 3.
http://orcid.org/0000-0002-8297-089X.

Місце роботи авторів:

Національний університет «Одеська політехніка»,
65044, Одеса, проспект Шевченка, 1.
Тел.: (38)(048)705-85-66
E-mail: nkomlevaya@gmail.com
E-mail: zinovatnaya.svetlana@op.edu.ua
E-mail: lvv@op.edu.ua

Одеський національний медичний університет,
65082, Одеса, пер. Валіховський, 2.
Тел.: (38)(048) 728-54-74
E-mail: shurik73.jan@gmail.com.

Прізвища та ім'я авторів і назва доповіді англійською мовою:
Komleva N. O., Zinovatna S. L., Liubchenko V. V., Komlevoi O. M.
Features of building recommendation systems based
on neural network technology using multithreading

Прізвища та ім'я авторів і назва доповіді українською мовою:
Комлева Н. О., Зіноватна С. Л., Любченко В. В., Комлевой О. М.
Особливості побудови рекомендаційних систем на основі
технології нейронних мереж з використанням багатопотоковості

Контакти для редактора:
Зіноватна Світлана Леонідівна, доцент кафедри Інженерії
програмного забезпечення Національного університету «Одеська політехніка»,
e-mail: zinovatnaya.svetlana@op.edu.ua, phone: 067-939-00-94