Engineering of Software Recommender Systems Based on a Neural Network with Multithreading

Nataliia Komleva[1], Svitlana Zinovatna[1], Vira Liubchenko[1], Oleksandr Komlevoi[2]

[1] *Odesa Polytechnic National University, 1 Shevchenka av., Odesa, 65044, Ukraine*
[2] *Odesa National Medical University Valikhovsky Lane 2, Odesa, 65082, Ukraine*

Abstract

The article discusses the issue of creating application software using multithreading. A recommender system designed to provide recommendations to tourists regarding hotels has been developed. The system works using a neural network based on a multilayer perceptron. A hotel evaluation model described by objective and subjective features was built to create recommendations. The objective features include the characteristics of hotels, which can be presented as a logical value (presence or absence) or a quantitative assessment of the hotel's components (number of floors, area, etc.). Subjective features are estimations given by tourists on a specific scale. Among the subjective features, the most valuable are the textual reviews of tourists, which are analyzed with the determination of a particular emotional color; it can be classified as a positive, neutral, or adverse opinion. The recommender system was developed using the Python programming language and related libraries – spaCy, multiprocessing, etc. A preliminary data analysis was conducted to study text responses, including removing stop words, segmentation, reducing words to a single norm, and marking parts of speech. After that, the text responses were transformed into a valid feature space using the Word2vec algorithm and classified using machine learning methods. During the neural network training, the Word2vec algorithm tries to maximize the cosine similarity between the vectors of words in similar contexts and minimize the cosine distance between those words that are not located next to each other in the context. In the work, the neural network architecture was created using parallel computing, the training sample parallelization mechanism was used, and the activation function of the ReLU family was used. Data for training a convolutional neural network and a loss function describing how far the neural network model is from making ideal recommendation predictions for the given data are defined. The average absolute and root mean square errors, accuracy and completeness were used to check the quality of the recommendations. The results of the experiments showed that when analyzing ten HTML pages with descriptions of hotels with 500,000 epochs of neural network training when switching from sequential calculations to parallelization on two processors, the metrics for checking the quality of the received recommendations slightly deteriorated due to the costs of synchronizing gradients between processors, but when switching to 4 and 8 processors. When using eight processors, the precision metric showed the best results, which increased by 8.33%.

Keywords
Recommender system, objective features, subjective features, parallel computing, neural network

CEUR-WS.org/Vol-3501/s25.pdf

1. Introduction

Currently, a person is among a vast amount of information. And constantly, it is necessary to choose among a set of offers. While deciding, one can use friends' opinions, data from the Internet,

13th International Scientific and Practical Conference from Programming UkrPROGP'2022, October 11-12, 2022, Kyiv, Ukraine
EMAIL: nkomlevaya@gmail.com (A. 1); zinovatnaya.svetlana@op.edu.ua (A. 2); lvv@op.edu.ua (A. 3); shurik73.jan@gmail.com (A. 4)
ORCID: 0000-0001-9627-8530 (A. 1); 0000-0002-9190-6486 (A. 2); 0000-0002-4611-7832 (A. 3); 0000-0002-8297-089X (A. 4)
© 2022 Copyright for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

expert evaluations, and so on. It is essential to have an automated tool to help you make choices in any field (goods, movies, audio, books, news content, promotions, etc.).

Content-oriented filtering, collaborative filtering, or hybrid methods are usually used when creating a recommender system (RS) [1]. Matching the attributes of the user profile, which stores preferences and interests, with the content object (element) characteristics is the primary process in the case of RS based on content-oriented filtering [2]. Thus, RS deal with two entities, users, and items, where each user gives a rating (or preference value) to an item (or product) [3]. Collaborative filtering involves evaluating items through other people's opinions. Technology brings together the opinions of large, interconnected communities, supporting the filtering of significant amounts of data [4]. A hybrid approach between context-based and collaborative filtering includes various machine learning and clustering algorithms, eliminating each algorithm's shortcomings, and improving system performance since clustering, similarity, and classification lead to better recommendations and enhanced precision and accuracy [5].

Of particular interest is the development of software tools using artificial intelligence components in which specific design patterns are involved. In [6], the authors warn of the promising potentials of machine learning accompanying multifaceted challenges to traditional software development processes and practices.

«It can be more difficult to maintain strict module boundaries between machine learning components than for software engineering modules. Machine learning models can be "entangled" in complex ways that cause them to affect one another during training and tuning, even if the software teams building them intended for them to remain isolated from one another» [7].

Let's consider the development of recommendation systems as a system with artificial intelligence components. Recommender systems are designed to increase user satisfaction by providing helpful recommendations for various entities. Of course, in this case, the recommender system should have available data describing the user and the entities to which the recommendations are projected [8, 9].

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 [10].

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 [11].

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.

2. Purpose and tasks

The work aims to study 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:

- 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 recommender 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.

3. Analysis of literary sources and problem formulation

The work [12] 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.

In [13], it is shown that the most common methods are SentiWordnet and TF-IDF, and for machine learning Naïve Bayes and SVM. Choosing the right sentiment analysis method depends on the data itself. The authors consider applying these methods to solve the specific problem of analyzing sentiment in social networks.

The authors of [14] 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 [15], 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 [16] 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 [17];
- Word2Vec [18].

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 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 [19].

4. Formalization of the process of building a recommender 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:

$$H = (<Co_i \subseteq O, Cs_j \subseteq S>), \quad (1)$$

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.

Building a system for creating recommendations for complex objects includes several stages.

In the first stage, the evaluation object is studied, and the result of the stage is the determination of evaluation aspects.

The second stage is constructing the evaluation model; the result is directly the evaluation model.

In the third stage, a preliminary analysis of text data is performed; as a result, a set of prepared data in the form of text should be obtained.

In the fourth stage, the architecture of the neural network is determined using parallel calculations; the result of the stage is the architecture of the neural network, the loss function, and the method of optimizing the training procedure.

In the fifth stage, the text is transformed into a valid feature space, and the result of the stage is the formed feature vectors.

The following sixth stage is responsible for training the classifier; the result of the stage is the trained model of the classifier.

After that, at the seventh stage, the tonality of the text data is determined, and the result of the stage is the emotional coloring of the text data.

In the eighth stage, recommendations are created, resulting from the stage's implementation.

The last, ninth stage, is responsible for analyzing the received recommendations.

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 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 as 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.

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 follows:

$$Precision = \frac{TP}{TP + FP}, \tag{3}$$

$$Recall = \frac{TP}{TP + FN}. \tag{4}$$

5. Sentiment analysis using multithreading

Quite often, convolutional neural networks are used to analyze the tonality of texts [20]. Initially, convolutional neural networks were used for image recognition, but later they began 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 [21].

The idea of parallel computing is based on the fact that most tasks can be divided into 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. 1).

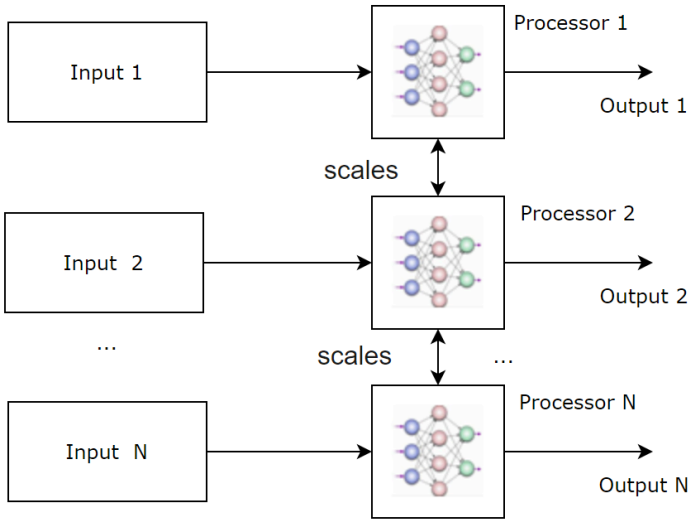

Figure 1: Parallelization of the training process

A difficult task in 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. 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 of 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 much 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 to quickly use large dictionaries (more than several hundred thousand words).

6. Results and discussion

A hotel is selected as the object for which a recommendation is issued. The recommender system should help the tourist choose the most attractive hotel based on his request. A software system was created to implement the proposed model, including subsystems for implementing business requirements and making relevant recommendations for tourists regarding hotels (Fig. 2).

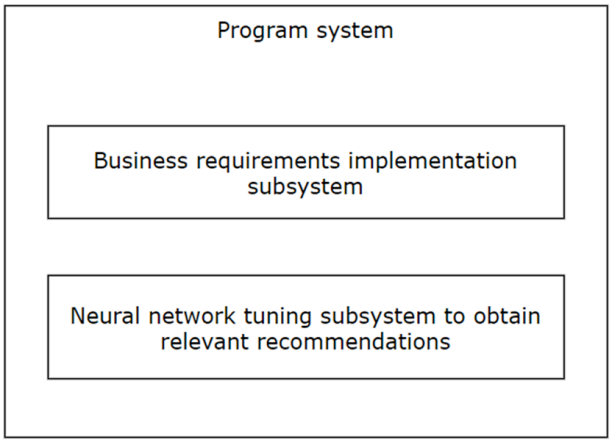

Figure 2: Generalized structure of the software system

The recommender system was designed based on the "Distinguish Business Logic from ML Model" pattern. The business logic depends on the results of the machine learning (ML) models, which may fail for various reasons. Hence, the overall business logic had been isolated from the ML models. Decoupling "traditional" business and ML components allowed the ML components to be monitored and adjusted to meet users' requirements and change inputs. Figure 3 shows the entire structure of the recommender system.

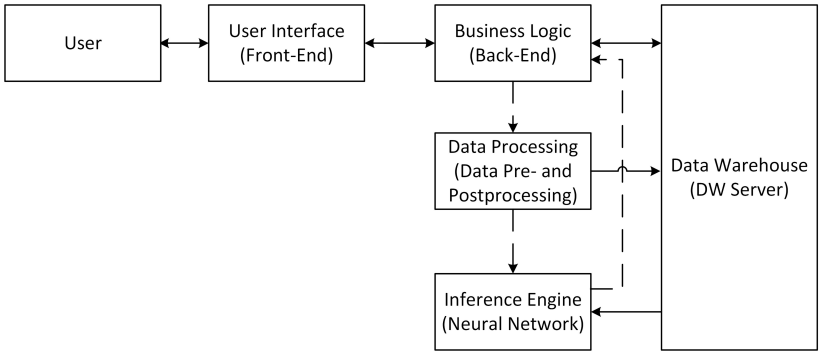

Figure 3: Logic representation of recommender system architecture

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).

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.

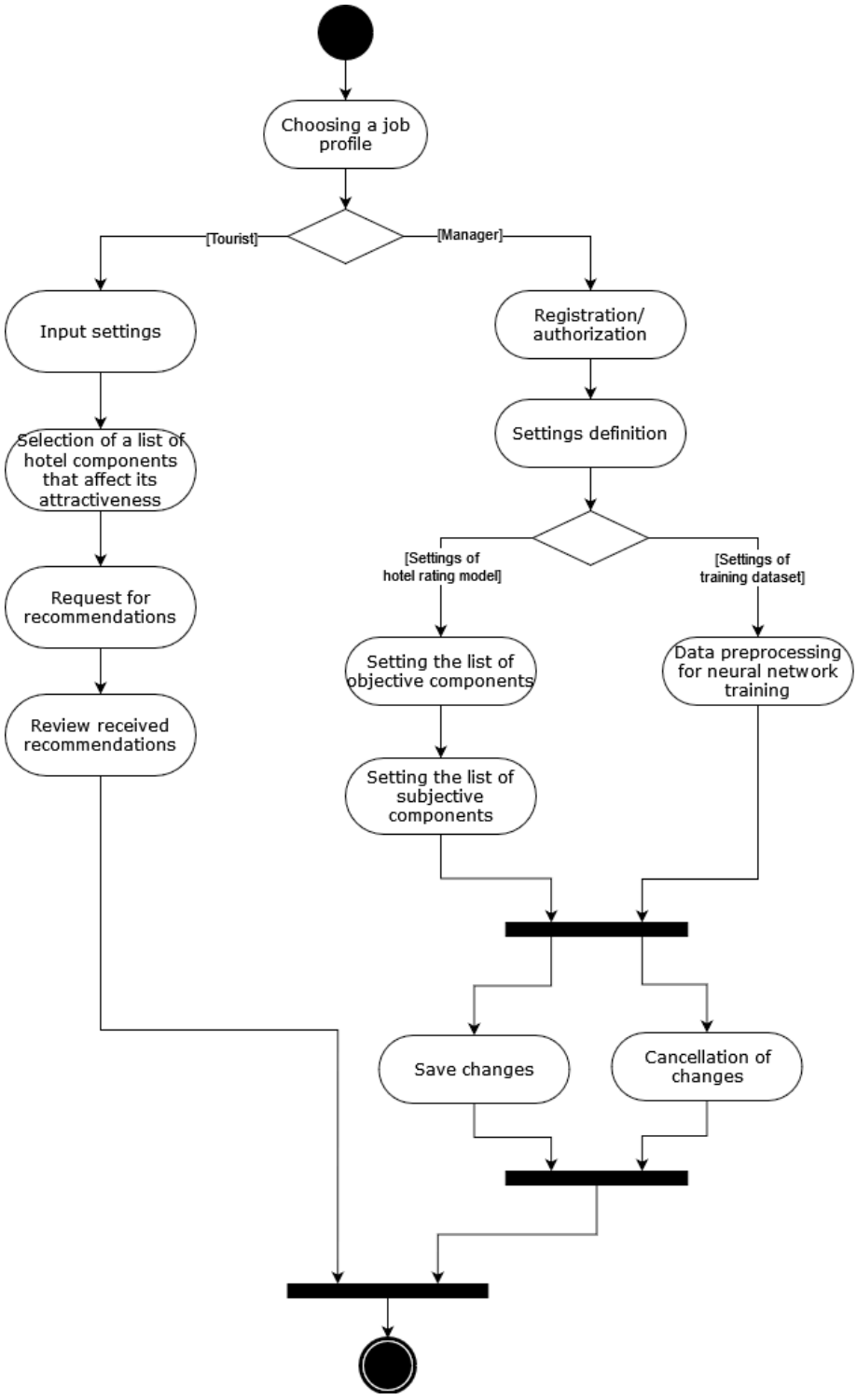

Figure 4: Activity diagram of the business requirement implementation

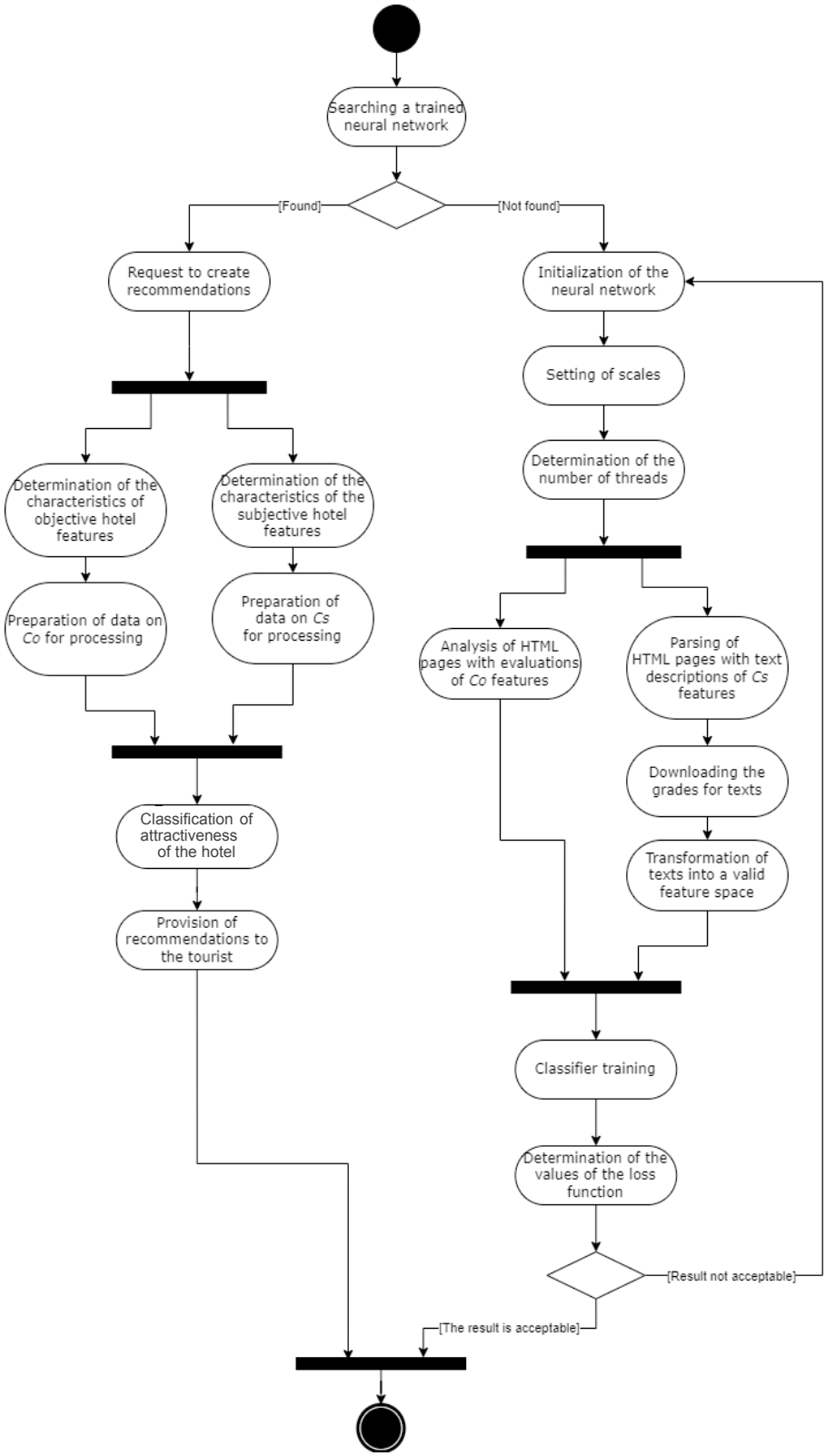

Figure 5: Activity diagram for creating relevant recommendations

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 is analyzed with the determination of a specific 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.

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 identifies and removes 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 appropriate APIs of the Python programming language multiprocessing package provides the parallelization of the training sample. 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 error backpropagation method trains 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 `ReviewClassification` 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.

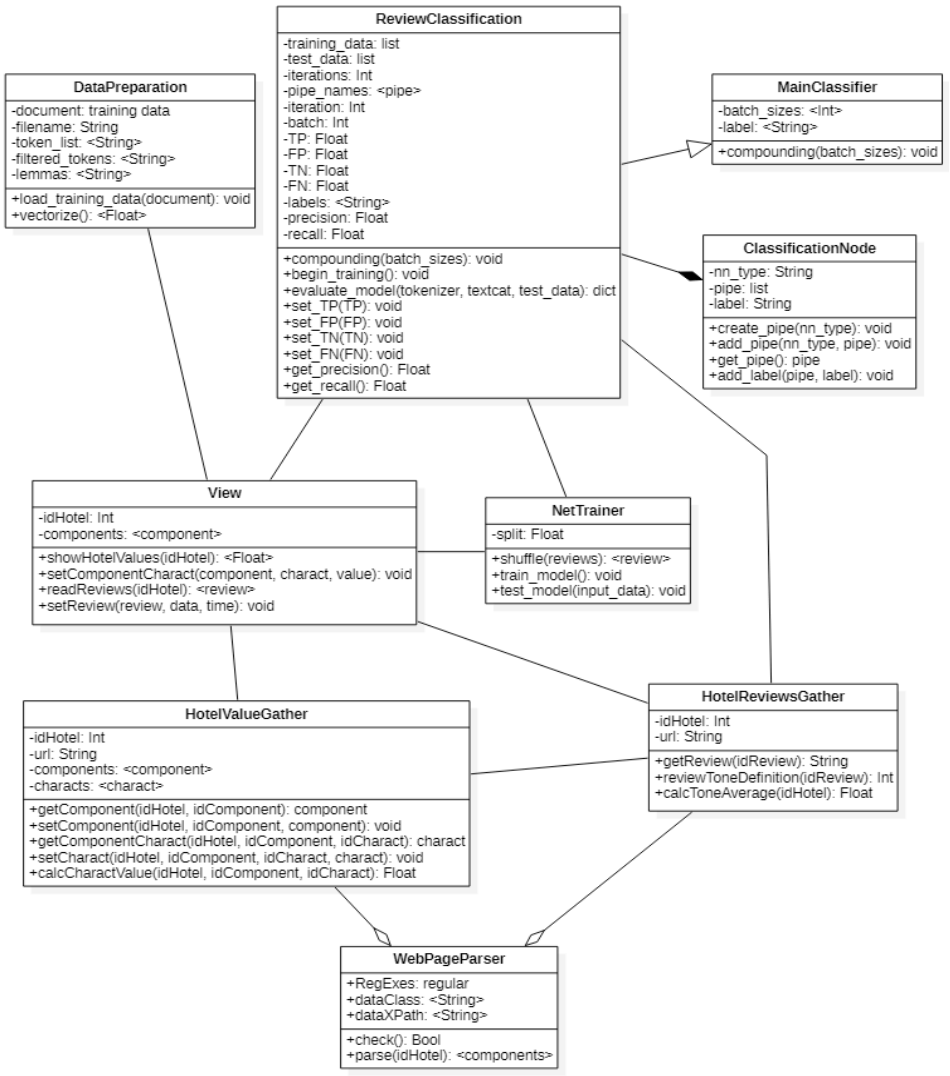

Figure 7: The software classes diagram

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 Precision's value by 8.33% and Recall by 2.83%.

Table 1

Dependencies between parallelization and values of execution time, Precision (3), and Recall (4)

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	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

7. 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 from 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 recommender system was developed. Therefore, the following research step will study the impact on the recommender system's quality of using different neural network structures.

8. References

[1] M. Jalali, N. Mustapha, Md. N. Sulaiman, A.Mamat, WebPUM: A Web-based recommendation system to predict user future movement. Expert Systems with Applications. 37 (9) (2010) 6201-6212. doi:10.1016/j.eswa.2010.02.105.

[2] P. Lops, M. de Gemmis, G. Semeraro, Content-based Recommender Systems: State of the Art and Trends. Recommender Systems Handbook. (2011) 73-105. doi:10.1007/978-0-387-85820-3_3.
[3] D. Roy, M. Dutta, A systematic review and research perspective on recommender systems. Journal of Big Data. 9, 59 (2022) 291-324. doi:10.1186/s40537-022-00592-5.
[4] J. B. Schafer, D. Frankowski, J. Herlocker, S. Sen, Collaborative Filtering Recommender Systems, in: P.Brusilovsky, A.Kobsa, W. Nejdl (Eds.), The Adaptive Web. Lecture Notes in Computer Science, volume 4321, Springer-Berlin, Heidelberg, 2007. doi: 10.1007/978-3-540-72079-9_9.
[5] G. Geetha, M. Safa, C. Fancy, D. Saranya, A Hybrid Approach using Collaborative filtering and Content based Filtering for Recommender System. Journal of Physics: Conf. Series 1000 (2018). doi:10.1088/1742-6596/1000/1/012101.
[6] S. Schelter, F. Biessmann, T. Januschowski, D. Salinas, S. Seufert, G. Szarvas, On Challenges in Machine Learning Model Management. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering (2018).
[7] S. Amershi et al., Software engineering for machine learning: A case study, in: Proc. 41st Int. Conf. Softw. Eng., Softw. Eng. Pract. (ICSESEIP), 2019, pp. 291–300, doi:10.1109/ICSE-SEIP.2019.00042.
[8] N. Yadav et al., Diversity in Recommendation System: Cluster Based Approach. Hybrid Intelligent Systems. 1179 (2020) 113–122. doi;10.1007/978-3-030-49336-3_12.
[9] S. Bag, G. Abhijeet, K. T. Manoj, An integrated recommender system for improved accuracy and aggregate diversity. Computers Industrial Engineering. 130 (2019) 187–197. doi:10.1016/j.cie.2019.02.028.
[10] S. V. Balshetwar, R. M. Tuganayat, G. Regulwar, Frame Tone and Sentiment Analysis. International Journal of Computer Sciences and Engineering. 7 (2019) 24–40. doi:10.26438/ijcse/v7i6.2440.
[11] M. Wankhade, A. C. S. Rao, C. Kulkarni, A survey on sentiment analysis methods, applications, and challenges. Artif Intell Rev. (2022). doi:10.1007/s10462-022-10144-1.
[12] K. Raviya, M. Vennila, An Approach for Recommender System Based on Multilevel Sentiment Analysis Using Hybrid Deep Learning Models. 8th International Conference on Smart Structures and Systems (ICSSS). (2022) 01–06. doi:10.1109/ICSSS54381.2022.9782172
[13] Z. Drus, H. Khalid, Sentiment Analysis in Social Media and Its Application: Systematic Literature Review Procedia Computer Science. 161 (C) (2019) 707–714 doi:10.1016/j.procs.2019.11.174.
[14] S. Banker, A Brief Review of Sentiment Analysis Methods. International Journal of Information Sciences and Techniques. 6 (1/2) (2016) 89–95. doi:10.5121/ijist.2016.6210.
[15] G. Preethi et al. Application of Deep Learning to Sentiment Analysis for recommender system on cloud. International Conference on Computer, Information and Telecommunication Systems (CITS). (2017) 93–97. doi:10.1109/CITS.2017.8035341.
[16] R. K. Mishra, S. Urolagin, J. A. A. Jothi, Sentiment Analysis for POI Recommender Systems. Seventh International Conference on Information Technology Trends (ITT). (2020) 174–179. doi:10.1109/ITT51279.2020.9320885.
[17] W. Qader, M. Ameen, B. Ahmed, An Overview of Bag of Words; Importance, Implementation, Applications, and Challenges. 2019 International Engineering Conference (IEC). (2019) 200–204. doi:10.1109/IEC47844.2019.8950616.
[18] S. Li, B. Gong, Word embedding and text classification based on deep learning methods. MATEC Web of Conferences PY. 336 (2021). doi:10.1051/matecconf/202133606022.
[19] W. Hu, F. Huang, 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). (2020) 110–115. doi:10.1109/ICBAIE49996.2020.00029.
[20] A. Jacovi, O. Sar Shalom, Y. Goldberg, Understanding Convolutional Neural Networks for Text Classification. Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. (2018) 56–65. doi:10.18653/v1/W18-5408.
[21] H. Leung, S. Haykin, The complex backpropagation algorithm. IEEE Transactions on Signal Processing, 39(9) (1991) 2101–2104. doi:10.1109/78.134446.