# Problems and Goals of Recommender Systems

***Bekkamov Fayzi, Islamova Dildora***
*Teacher of the department of Information security, Karshi branch of TUIT, Uzbekistan*

***Davlatova Navbahor***
*Teacher of the department of Information technology, Karshi branch of TUIT, Uzbekistan*

**ABSTRACT**

***Background.*** *On the Internet, where the number of choices is overwhelming, there is need to filter, prioritize and efficiently deliver relevant information in order to alleviate the problem of information overload, which has created a potential problem to many Internet users. Recommender systems solve this problem by searching through large volume of dynamically generated information to provide users with personalized content and services. This paper explores the different characteristics and potentials of different prediction techniques in recommendation systems in order to serve as a compass for research and practice in the field of recommendation systems.*

***Method:*** *In this article, we review the key advances in collaborative filtering recommender systems, focusing on the evolution from research concentrated purely on algorithms to research concentrated on the rich set of questions around the user experience with the recommender. We argue that evaluating the user experience of a recommender requires a broader set of measures than have been commonly used, and suggest additional measures that have proven effective.*

***Result:*** *Based on our analysis of the state of the field, we identify the most important open research problems, and outline key challenges slowing the advance of the state of the art, and in some cases limiting the relevance of research to real-world applications.*

***Conclusion.*** *Recommender systems are an advanced form of software applications, more specifically decision-support systems, that efficiently assist the users in finding items of their interest. Recommender systems have been applied to many domains from music to e-commerce, movies to software services delivery and tourism to news by exploiting available information to predict and provide recommendations to end user. The suggestions generated by recommender systems tend to narrow down the list of items which a user may overlook due to the huge variety of similar items or users' lack of experience in the particular domain of interest.*

**KEYWORDS:** *recommendation system, collaborative filtering, content-based recommendation systems, hybrid recommender systems.*

## Introduction

The massive increase of structured and unstructured data available on the internet introduced the concept of Big Data that is difficult to process using traditional data processing techniques. This abundance of uncategorized data on the internet makes it difficult to find useful information and creates the problem of information overload. In order to handle such problems two major internet technologies, information search retrieval, and recommendations, have been developed over the last decade. Recommendations are a growing paradigm which automatically presents and assist the user with what he/she is looking to search for from a huge amount of information. This paradigm is known as a recommender system. Recommender systems emerged as an independent research area during the 1990s, when researchers and practitioners started focusing on the problems of recommendations that explicitly rely on the ratings provided by the user as a way to capture user's preferences for different items. These user preferences further help to specify the initial ratings for items. These ratings can be further used to recommend items to different users [1, 6, 11].

Recommender systems are software applications that are used to recommend a product or service in order to optimize some user-centered goals in light of the internal uncertainty surrounding users and content. Traditionally, recommendation systems have been useful for helping users to search large information spaces such as product collections (movies, books, music CDs), documents (news articles, medical texts, Wikipedia articles), or users to find partners (dating services)), players / teams of online games, consumer markets. The recommended system can be defined as a decision-making approach for users in complex information environments. It is an advanced software application that helps users search for knowledge records according to their interests and preferences, which users express in the form of ratings. Recommendations use these ratings to predict what the user will like in the future [2, 7].

**Recommendation system as a problem**

According to Celma, the recommendation problem can be split into two subproblems: a prediction problem and a recommendation problem. The first one is about the estimation of the items' likeliness for a given user and the second problem is to recommender a list of N items, which is reduced to list the top-N items once the system can predict items into a totally ordered set [4].

Sarwar et al formalize the prediction problem as follows: let $U = \{u_1, u_2, \dots, u_m\}$ be the set of $m$ users, and let $I = \{i_1, i_2, \dots, i_n\}$ be the set of $n$ recommendable items. Each user $u_i$ has a list of items $I_{u_i}$, which represents the items that the user has expressed his or her opinion about through explicit or implicit feedback. Note that $I_{u_i} \subseteq I$, and that $I_{u_i}$ can be empty, $I_{u_i} = \theta$. The function $P_{u_a, i_j}$ is the predicted likeliness of item $i_j \notin I_{u_a}$ for the active user $u_a$. Recommendation is a list of $N$ items, $I_r \subset I$, that the user will like the most, the N items with the highest $P_{u_a, i_j}$ values. The recommended list should not contain items from the user's interests, $I_r \cap I_{u_a} = \theta$.

The set $I$ of possible items can be very large, which is also true for the user set $U$. In most recommender systems, the prediction function $P_{u_a, i_j}$ is usually represented by a rating, which is given by the user either explicitly or implicitly through some measures, by tracking if a song is skipped. They are represented as triples $(u, i, r)$ where $r$ is the rating value assigned by the user $u$ to a particular item $i$. The value is usually a real number (from 0 to 1), a value in a discrete range (from 1 to 5), or a binary variable (like/dislike) [3, 5].

There are several factors that affect the quality of recommendations:

**Table 1.** Factors that affect recommendations

| | |
|---|---|
| Novelty | A high rate of novel recommendations can make the quality of recommendations seem poor to the user. Recommending some familiar items increases the user's confidence in the recommender. |
| Serendipity | A recommender should help the user discover unexpected yet interesting items that they might not be able to discover otherwise. |
| Explainability | Giving explanations about recommended items can improve the user's trust in the recommender system. |

| | |
|---|---|
| Cold start problem | When a new user or item enters the system the lack of data prevents the system from giving useful recommendations. |
| Data sparsity and high dimensionality | High dimensionality of both users and items can result in low coverage of users' interactions with the items. |
| Coverage | Low coverage of the domain limits the space of possible items to be recommended. |
| Trust | Recommender systems that are trust-aware determine which users can be reliably used for recommendations, and which cannot. |
| Attacks | Recommender systems can be attacked, which reduces the quality of recommendations. An example of an attack is deliberate mistagging, which happens when a group of users tag an item using a false or malicious tag. |
| Temporal effects | Recommender systems can treat older items as less relevant than the new ones. The system has to decide which items from a user profile are taken into account when computing the recommendations. |

**Recommendation systems: state of the art**

Recommendation systems are the applications of software that can provide suggestions for users about particular lists of products such as movie, books or services in a personalized manner (in the case of e-commerce recommender systems). Recommendation systems have been very useful in assisting users in the scenarios of information overload that makes the exploration and selection of items from large information space a difficult task. Recommendation systems offer a personalized suggestive assistance in form of discovery as an effective way of improving revenue on many e-commerce and media streaming platforms such as Amazon, Netflix, Youtube and many others by increasing user satisfaction [9, 12].

Basically, recommender systems are built on the theories, algorithms, and technologies from different domains such as information retrieval (IR), artificial intelligence (AI), machine learning (ML), human-computer interaction (HCI), E-commerce marketing. Recommender systems remain an emerging and active research topic that has attracted more attention in the last decade.

A recommender system attempts to estimate / predict a rating function $R$ on the basis of an initial set of ratings. The rating function $R$ can be specified with the help of following Equation 1.

$R = User \times Item \rightarrow Rating$ (1)

Rating, in above formula, can be defined as an order set to capture preferences of the users, while users and items are the domains of users and items. A recommender system recommends the items that are highest rated, for the user once the rating function R is defined for the user and item space. In the following Figure 1, an approach following the traditional method of a two-dimensional recommender system is given [8].

The basic recommender system problem is to estimate or predict a utility function that can help to predict how a user will like an item. In the utility function, $U$ is represented as set of users $U \coloneqq \{users\}$, $I$ is represented as a set of recommendable items $I \coloneqq \{recommendableitems\}$ where $F \coloneqq utilityfunction$ then

$F = U \times I \rightarrow R$ (2)

Where $R \coloneqq \{Recommendeditems\}$ and for each user $u$ we want to choose the items $\tau$ that maximize $f$, as shown in the following Equation 3.

$u \in U\ \tau_u = argmax_f F(u, i)$ (3)

The traditional paradigm of recommender system has three main aspects: user, item, and rating. Rating, in this case, represents the feedback that a user gives for a specific item.

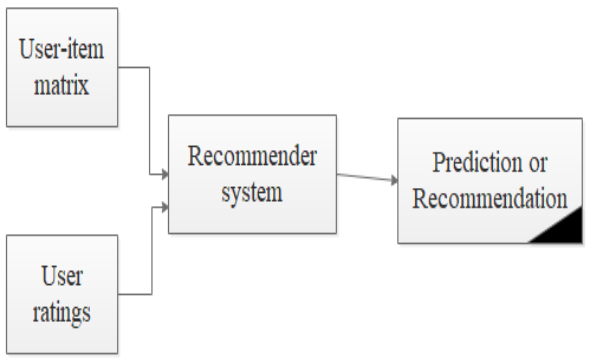

**Figure 1.** A Traditional recommendation approach

Ratings can be in different forms; implicit and explicit. Ratings can be defined as explicit when a user rates an item to share shopping experience. Ratings can be defined as implicit when a user buys an item and the information comes in the form of user logs. Rating can be in the form of a numeric value on a multi-point scale e.g. 1 to 5 or can be in a binary form such as yes/no format [10, 11].

One of the most studied cases in the domain of recommender systems is movie recommendation for Netflix in which the recommendation task is to suggest the movies to the users from a large list of movies. The suggestions or recommendations are made based on the preferences that users provide in form of initial ratings of the movies that they have watched. However, there are many other examples in which the typical conditions of movie recommendation do not apply such as book recommendations where the contents of the books are used for recommendation process.

## Collaborative filtering

Collaborative filtering helps people make decisions based on the opinions of others who share similar interests. Collaborative filtering recommends based on the historical interaction of user-element relationships, either explicitly - for example, from previous user ratings (a user-based method), or using implicit feedback (a subject-based method) - for example, from browsing history. With a custom approach, users are provided with recommendations for products that are popular with similar users. With a substantive approach, the user receives product recommendations similar to those he had in the past [13, 15].

## Content-based recommendation systems

A content-based recommender system is usually based on comparing individual elements and useful information about users. Different types of information can be taken into account, such as images, videos or texts (user reviews, texts of books or music, and others). Content-based technique is a domain-dependent algorithm and it emphasizes more on the analysis of the attributes of items in order to generate predictions. When documents such as web pages, publications and news are to be recommended, content-based filtering technique is the most successful. Systems implementing content-based recommendations analyze a set of documents and descriptions of items previously assessed by the user and create a model or profile of user interests based on the properties of objects assessed by the user. A profile is a structured form of user interests that is used to recommend new interesting things to users. The recommendation process mainly consists of adapting user profile attributes and content object attributes. The result is a relevance check that shows the user's level of interest in the object. It is very important for the efficiency of the information access process that the profile accurately reflects the user's preferences. For example, they can be used to filter search results when the system decides whether a user is interested in a particular website. Otherwise, this page will not be displayed [13, 14, 15].

**Problems of traditional recommender models**

However, the above models have their limitations in terms of data sparseness, cold start issues, and balance quality recommendations in terms of various evaluation metrics.

To achieve higher performance and overcome the shortcomings of traditional recommendations, hybrid recommendations have been proposed that combine the best features of two or more recommendation techniques into a single hybrid technique. The most common approach in existing hybrid systems methods is to combine the collaborative filtering recommendation method with other recommendation methods to avoid problems with cold start, sparse data, or data scalability. These systems use both user rating information and available user and product information and data to generate better recommendations. Both implicit and explicit information (obtained when creating a user profile) are taken into account [9, 12, 13].

The main problem for recommender systems is the so-called cold start problem. In order for the system to adapt to the user, he must know what the user wants and what is important to him. This is necessary for content-based filtering to make decisions about items similar to those that users have liked in the past.

What if we still don't know anything about users who have just started using the system? The developers of these systems tend to solve the problem either by asking users to rate the items first, or by offering them a demographic questionnaire from which certain stereotypes can be deduced (for example, older people listen more to classical music). Therefore, we require users to explicitly complete a user profile.

Both methods require user effort. It's also not easy to decide which items users should rate. Also, stereotypes can be relatively bad and offensive (for example, some people prefer popular music and don't want to be seen as older). We will gradually learn about a new user's taste, for example by evaluating our recommended items, or by using a more implicit method of checking whether they spend time on those items or not. We make recommendations to the new user so that a group of existing users can be satisfied, including the new user (or more accurately, the person we now consider the new user). The weight given to a new user will be low at first, because we don't know much yet, and will gradually increase.

The data sparse problem is caused by the lack of transaction and feedback data, which limits the usability and success of collaborative filtering and other methods. This problem can be minimized, for example, by direct or indirect similarity between users and by computing a similarity matrix from the relative distance between users' ratings. Recently, however, new recommender systems have been created that try to minimize this problem as much as possible. To do this, they use machine learning and complement the disadvantages of collaborative filtering with a low density of user data.

Therefore, recommender systems are used to evaluate user preferences for items that users have not yet seen. Based on the obtained results, we separate recommendation systems into ranking prediction, top n item prediction, and classification.

The rating prediction system seeks to fill in as many missing elements as possible in a matrix containing the ratings that the user has assigned to individual elements in the past. The result of the top-n system is an evaluated list of elements of length n. The classification system focuses on classifying candidate elements into the correct categories for recommendations.

## Natural language processing

Natural language processing [6] focuses on the interaction between human language and computers. The field of informatics, artificial intelligence and computational linguistics intersects here. Natural language processing is a way for computers to analyze, understand, and derive meaning from human language in an intelligent and useful way. Using Natural language processing, developers can organize and structure knowledge to perform tasks such as automatic summarization, translation, named entity recognition, dependency extraction, sentimental analysis, speech recognition, and topic recognition. Developing Natural language processing applications is challenging because computers usually require people to communicate with them through a programming language. Programming languages are accurate, unambiguous and highly structured. However, human speech is not always accurate, is often ambiguous, and language structure can depend on many complex variables, including slang, regional dialects, and social context [5, 9].

Unlike pixels in image processing, in Natural language processing tasks, sentences or documents are represented as matrices by input [11]. Each line of the matrix corresponds to one token, typically a word, but it can also be a character. Thus, each line is a vector representing a word. These vectors are mostly low-dimensional representations, but they can also be one-time vectors that index a word into a dictionary. For a sentence of ten words using 100 dimensions, we would have an input matrix of 10 x 100.

In computer vision, filters move through small portions of the matrix, but in natural language processing, we usually use filters that move across entire lines (words). The width of the filters is therefore usually the same as the width of the input matrix. The height or size of the area may vary, but sliding windows processing 2 to 5 words at a time are typical. An example of the use of a convolutional neural network for language processing is shown in Figure 3.3.

The most suitable problems for convolutional neural networks appear to be classification tasks, semantic analysis, spam detectors or topic categorization. Convolution and pooling operations lose information about the local word order, so sequential tagging, such as in “PoS tagging” or “Entity extraction”, is a bit more difficult due to the pure convolutional neural networks architecture [10].

## Word representation in natural language processing

In natural language processing, words are usually converted to vectors with certain properties. Most so-called embedding algorithms are able to convert lexical units, mostly words, into a vector space in which the morphological, syntactic and some semantic properties of these words preserve linear dependencies.

Word embedding is a class of approaches for representing words and documents that use dense vector representation. This is an improvement on the traditional bagof-words code scheme, where large sparse vectors are used to represent each word in the vector to represent the entire vocabulary. These expressions were rare, because vocabulary is usually huge and the word or document represents a large vector, which consists mostly of zero values [14].

Instead, in embedding algorithms, words are represented by dense vectors, where the vector represents the projection of the word into a continuous vector space. The position of a word in vector space is learned by the algorithm from the text and is based on the words that surround the word as it is used. The position of a word in a learned vector space is referred to as its embedding.

Two popular examples of methods of learning word anchors from the text present are Word2Vec and GloVe.

Word2Vec provides an efficient implementation of the bag-of-words and skip-gram architecture for calculating the vector representation of words. The skip-gram model contains a corpus of words $w$ and their contexts $c$. We consider the conditional probability $p(c|w)$ and with respect to the corpus text we try to set the parameters $\varphi$ probability $p(c|w;\varphi)$ to achieve the maximum probability of the corpus (4):

$$\underset{\varphi}{argmax} \prod_{w\in Text} \left|\prod_{c\in C(w)} p(c|w;\varphi)\right| (4)$$

Where $C(w)$ is the set of contexts of the word$w$.

One approach for parameterizing the skip-gram model and modeling the conditional probability using a softmax is given in Equation (7).

$$p(c|w;\varphi) = \frac{e^{v_c\cdot v_w}}{\sum_{c'\in C} e^{v_{c'}\cdot v_w}} (5)$$

where $v_c$ and $v_w \in R^d$ is a vector representing $c$ and $w$, respectively, and $C$ is the set of all available contexts. The parameters $\varphi$ are $v_{c_i}$, $v_{w_i}$ for $w \in V$, $c \in C$, $i \in 1,\dots,d$ (total $|C| \times |V| \times d$ parameters). We choose the parameters so as to reach the maximum value for (4).

Here are 3 filter sizes, each of which has 2 filters. Each filter performs a convolution on a sentence matrix and generates an activation map of different lengths. Then we apply 1-max pooling to each of the maps. This will select the largest number from the map. The result is a vector from all six maps, where all features are combined to form a vector for the penultimate layer. The last softmax layer accepts this vector as its input and uses it to classify the sentence. We assume a binary classification here, so we display two possible output states.

GloVe is a second algorithm for obtaining a vector representation of words with learning without a teacher. The training is performed using summary global statistics on the common occurrence between words from the corpus and the resulting representations show interesting linear substructures of the word vector space.

**Linear substructures**

Similarity metrics used to evaluate nearest neighbors (methods working with two word vectors that effectively measure the linguistic or semantic similarity of corresponding words) produce a single scalar value that quantifies the connection of two words. This simplicity can be problematic because the two words almost always have more complex relationships than can be covered by a single number. For example, the word man can be considered similar to woman in that both words describe human beings. On the other hand, the two words are often considered to be the opposite because they emphasize the primary axis along which people differ from each other [12].

In order to qualitatively capture the nuances needed to distinguish a man from a woman, it is necessary for the model to associate more than one number for word pairs. A simple candidate to extend the set of discriminant numbers is the vector difference between two vector vectors. GloVe is designed to capture such vector differences as best as possible by comparing two words.

The basic concept that distinguishes a man from a woman, ie gender, can be equally specified by various other pairs of words, such as king and queen or brother and sister. To express this observation mathematically, we can expect that the vector differences between man and woman, king and queen, brother and sister, could be almost the same. This fact can be seen in the pictures. The GloVe model [28] is trained on non-zero inputs of the global coherence matrix.

The embedding layer transforms positive integers, or indices, into dense vectors of fixed size. This is in contrast to one-hot encoding, where if we have, for example, a dictionary of 1000 words, we represent the word with a vector of length 1000, which contains a large number of zeros. This condition is not computationally advantageous for large datasets. When training a neural network with this layer, the vector that is associated with the individual inputs of the layer is updated. An important stage when using the Embedding layer is to encode the input words using indexes. An example representation of the input sentence can be seen in Figure 3.4. Subsequently, an embedding matrix is created. We decide how many latent factors to assign to each index, which indicates how long the given vector will be. Usually lengths such as 32 or 50 are chosen. This means that compared to a dimensional vector in one-hot coding, we keep the size of the embedding matrix more reasonable [9, 12].

Since the built-in vectors also update during deep neural network training, we can investigate which words are similar in multidimensional space. Visualization is possible using dimensional reduction techniques, such as the t-SNE technique. Capturing such relationships in common language is relatively complex, so word embedding is very important when processing natural language.

Bag of Words is an algorithm that counts the number of times a word appears in a document. Individual word counts are used to compare documents and measure their similarities. This technique is used in classification, searching, or in creating statistical models. A number of the most common words, or a predetermined dictionary of words to be counted, are used to create the input of the deep neural network.

TF-IDF is a methodology for evaluating relevance in text search, which is based on Bag of Words, but takes into account the frequency of words in all documents. Hence the name Time Frequency and Inverse Document frequency in all documents. This technique takes into account the importance of the word throughout the corpus of documents, making it one of the most widely used techniques in word processing-based recommendation systems, the first component is defined as:

$$tf_i, j = \frac{n_i, j}{\sum_k n_k, j} \quad (6)$$

where $n_i, j$, is the number of occurrences of the word $t_i$ in the document $d_j$ and the denominator contains the sum of all the number of words in the document $d_j$.

The IDF section takes into account the importance of the word. The most common words are the least important. These are, for example, the English members of "a" or "the". We perform the calculation according to the formula:

$$idf_i = log \frac{|D|}{|\{j : t_i \in d_j\}} \quad (7)$$

where $i$ is the analyzed word, $|D\ rvert$ is the number of documents and the denominator contains the number of documents in which the processed word $i$ is contained.

The resulting value of TF-IDF [1] is then obtained by multiplying the two parts with each other, $TF \cdot IDF$.

The use of recurrent neural networks in language processing can be more intuitive because they resemble how we process language: by reading the text from left to right . However, this does not mean that convolutional neural network does not work in this area. The big advantage of convolutional neural network is their high speed. Convolution is a central part of computer graphics and is implemented at the hardware level of the graphics process.

## Conclusion

The recommender system solves the problem of information overload and opens up new possibilities for obtaining personalized information on the Internet and for users' access to products and services. This paper describes content-oriented and collaborative filtering, and also compares their strengths and weaknesses with hybrid filtering to improve the efficiency of the system. In this paper, defined the basic recommendation problem and explained how traditional recommender systems work. Reviewed different algorithms and techniques used for recommender systems and discussed the motivation about context-aware recommender systems.

**References**

1. F. Abel, Q. Gao, G.-J. Houben, and K. Tao. Analyzing user modeling on twitter for personalized news recommendations. In Proc. 19th International Conference on User Modeling, Adaption, and Personalization, UMAP'11, pages 1–12, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-22361-7. URL http://dl.acm.org/citation.cfm?id=2021855.2021857.
2. Lu, J.; Wu, D.; Mao, M.; aj.: Recommender System Application Development s: A Survey 2019
3. Sarwar, B.; Karypis, G.; Konstan, J.; aj.: Item - based collaborative filtering recommendation algorithms. 2018
4. Resnick, P.; Iacovou, N.; Suchak, M.; aj.: GroupLens: an open architecture for collaborative filtering of netnews. 2018
5. Shambour, Q.; Lu, J.: hybrid trust - enhanced collaborative filtering recommendation approach for personalized government - to - business e - services, International Journal of Intelligent Systems. 2017
6. Madadipouya, K.; Chelliah, S.: A Literature Review on Recommender Systems Algorithms, Techniques and Evaluations. BRAIN: Broad Research in Artificial Intelligence and Neuroscience, ročník 8, č. 2, July 2017.
7. P., L.; de Gemmis M.; G., S.: Content-based Recommender Systems: State of the Art and Trends. In: Ricci F., Rokach L., Shapira B., Kantor P. (eds) Recommender Systems Handbook. Springer, Boston, MA, 2011, ISBN 978-0-387-85819-7.
8. Lisa Wenige, Johannes Ruhland, Retrieval by recommendation: using LOD technologies to improve digital library search, © Springer Verlag GmbH Germany 2017
9. Mingdan Si, Qingshan Li, Shilling attacks against collaborative recommender systems: a review, Artificial Intelligence Review (2020) 53:291–319
10. Hui Li, Yan Gu, Saroj Koul, A Review of Digital Library Book Recommendation Models, 2015
11. Joseph A. Konstan, John Riedl, Recommender systems: from algorithms to user experience, © Springer Science+Business Media B.V. 2012
12. Emmanouil Vozalis, Konstantinos G. Margaritis, Analysis of Recommended Systems Algorithms, 2014
13. Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. Computer, 42(8):30–37, Aug. 2009.
14. Q. Lu, T. Chen, W. Zhang, D. Yang, and Y. Yu. Serendipitous personalized ranking for top-n recommendation. In Proc. The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Volume 01, WI-IAT '12, pages 258–265, Washington, DC, USA, 2012. IEEE Computer Society.Rustamov A., Bekkamov F., Recommender systems: an overview, Scientific reports of Bukhara state university, 2021/3(85).
15. Буранова, М. А. (2020). ИННОВАЦИИ-ЗАЛОГ РАЗВИТИЯ И КОНКУРЕНТОСПОСОБНОСТИ ПРОМЫШЛЕННОСТИ СТРАНЫ. *Интернаука*, (13-2), 9-11.

16. Хашимова, Н. А., & Буранова, М. А. (2020). РАЗВИТИЕ ИНТЕЛЛЕКТУАЛЬНОГО ПОТЕНЦИАЛА ЗАЛОГ УСПЕШНОЙ ПОЛИТИКИ РУЗ. *Интернаука*, (13-2), 28-29.
17. Буранова, М. А., & Сайфутдинова, Н. Ф. (2020). РАЗВИТИЕ ПРОМЫШЛЕННОСТИ-ОСНОВА КОНКУРЕНТОСПОСОБНОСТИ СТРАНЫ. *Интернаука*, (13-2), 12-14.
18. Буранова, М. А. (2019). Перспективы развития электроэнергетической отрасли в условиях модернизации экономики Узбекистана. *Российский внешнеэкономический вестник*, (7), 60-63.
19. Буранова, М. А. (2019). Модернизация–ключ к развитию энергетики. *Экономика и финансы (Узбекистан)*, (5).