Applying Natural Language Processing Models to Create Recommendations for Professional Skills Development

Katsiaryna Kosarava [1]

[1] *Yanka Kupala State University of Grodno, 22 Ozheshko str., Grodno, 230023, Belarus*

Abstract

The article examines the problem of applying intellectual methods and models for the automatic extraction of professional skills and competencies from the descriptions of academic disciplines in Russian. The accuracy of pre-trained multilingual models of distributed vector representation LASER and SBERT is investigated. The process of collecting and preprocessing a dataset for creating and training skills classification models is described. A neural network model of skills classification based on a bidirectional recurrent neural network with long-short-term memory is described and trained. It was trained on data prepared two different ways: 1) skills were separated on "soft skills" and "hard skills", the last one was not translated into Russian; 2) "soft skills" and "hard skills" were treated as one class and were translated into Russian. Learning curves on validation data are presented. They show high accuracy of trained models up to 99%. The test accuracy of the models was analyzed employing confusion matrices and the results show a fairly high performance of the model. an example of skills extraction from an academic discipline using trained models is given.

Keywords [1]

Professional skills, academic disciplines, natural language processing, word2vec, LASER, SBERT, neural network

1. Introduction

The sphere of education and training is one of the most conservative, and often training programs have not changed for decades. However, rapidly developing technologies and the development of new sectors of the modern economy require a specialist to master new skills that are not provided for in classical higher education. Thus, the main problem of education is the actualization of educational specialties and disciplines, by the needs of the modern labor market. Understanding what skills will be in demand shortly and what disciplines will ensure the development of these skills will allow students to form a plan for their professional growth. However, educational standards and curricula do not contain articulated skills. As in other industries, artificial intelligence has the potential to optimize the process of finding basic information in unstructured text and performing labor-intensive tasks for people [1, 2]. There are several ways in which artificial intelligence helps to match skills and their corresponding directions in work and learning. The article demonstrates the use of neural network models to search for professional skills in the descriptions of academic disciplines. This will help to optimize the choice of suitable disciplines by students, and also thanks to this approach, the picture of the relevance and demand for certain disciplines will be as plausible as possible.

The purpose of this article is to describe the application of distributed vector representation models to compare educational disciplines and skills that a student can master while studying them, as well as a comparative analysis of the accuracy of these models.

Proceedings of VI International Scientific and Practical Conference Distance Learning Technologies (DLT–2021), September 20-22, 2021, Yalta, Crimea
EMAIL: koluzaeva@gmail.com
ORCID: 0000-0001-7326-5307

2. Distributed vector representation models

In text mining, the first step is to convert the text to a form that is understandable for the machine learning model. The most famous approaches are "one-hot encoding", encoding each word with a unique number, word embeddings. All of these approaches are widely known and are described in detail in the literature [3].

2.1 Word Embeddings

The “Word embeddings” approach not only allows you to “encode text”, but also makes it possible to use an efficient dense representation in which similar words have a “similar” encoding. The result of inlining is a dense vector of floating-point values (the length of the vector is a parameter you specify, which does not depend on the size of the dictionary). Instead of specifying the values to be embedded manually, they are trainable parameters (weights obtained by the model during training, just like the model learns weights for a dense layer) [4]. When working with large datasets, it is common to see word embeddings that are 8-dimensional (for small datasets), up to 1024-dimensional. Higher-dimensional embedding can capture fine-grained relationships between words but requires more data to explore.

This method allows you to capture the context of a word in a document, semantic and syntactic similarity, relationships with other words, etc. Word embedding helps in achieving such goals as identifying similar words, semantic grouping, which groups things with the same characteristics together and dissimilarly far away; text classification: the text is matched with arrays of vectors, which are passed to the model for training and predicting; clustering of documents, etc. Some examples in which vectorized words can be used directly include applications for synonym creation, autocorrect, and predictive text input. Similar types of methods are used to perform fuzzy searches on Google and similar search tools with an almost infinite number of internal search capabilities that can be applied to directories and databases of organizations. In addition, since embeddings usually have good behavior, you can also perform arithmetic operations on vectors. This allows for unique operations where embeddings not only capture similarities between words but also encode higher-level concepts. The embeddings also allow logical analogies. For example, Rome is for Italy, like Beijing is for China - word embeddings can use such analogies and directly give believable answers.

Finally, almost all other modern architectures now use some form of embedding layer and language model as the first step in performing subsequent NLP tasks. These downstream tasks include document classification, named entity recognition, question and answer systems, language generation, machine translation, and more.

2.2 Word2Vec

An example of models implementing the word embedding approach is the word2vec model. Word2vec is a well-known concept used to generate vector representations from words [5]. Word2vec is a three-layer neural network that processes text by “vectorizing” words. Its input is a text corpus, and its output is a set of vectors that represent words in that corpus, Figure 1. The purpose and usefulness of word2vec are to group vectors of identical words in a vector space, similar words are represented by vectors that are close in space. However, the word2vec model is not multilingual. For this reason, the results can be significantly distorted depending on the language used, as well as in the presence of untranslatable terminology (names of manufacturers, technologies, etc.).

Word2Vec uses 2 algorithms: Continuous Bag of Words (CBoW) and Skip-gram model. CBoW creates a sliding window around the current word to predict it from the used "context" of the surrounding words. Each word is presented as a feature vector containing information describing the most important characteristics of the object. After training, these vectors become word vectors. Vectors that represent the same words are close in different distance metrics and additionally include numerical ratios.

Skip-gram is the opposite of CBoW. Instead of predicting one word at a time, it uses one word to predict all the surrounding words ("context"). Skip-gram is much slower than CBoW, but is considered more accurate for rare words.

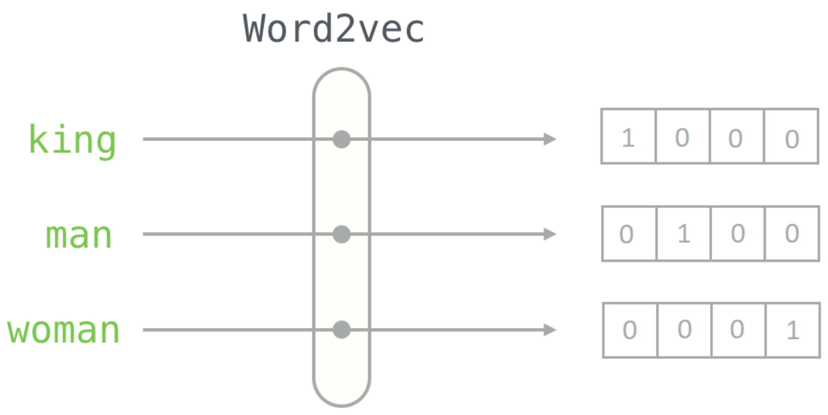


Figure 1: Word2Vec model

2.3 Large Pre-Trained Models for NLG

With the development of deep learning, various neural network architectures have come to be widely used to solve NLP problems. One of the main advantages of deep neural models is their ability to design features [6].

Pre-trained models (PTMs) can learn generic language representations in a large corpus, which is useful for subsequent NLP tasks and avoids learning a new model from scratch. First-generation PTMs are aimed at learning good embeddings for Skip-Gram [5] and GloVe [Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In EMNLP, 2014]. Second-generation PTMs are focused on learning contextual word embeddings such as ELMo [Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In NAACL-HLT, 2018], OpenAI GPT [Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. URL: https://s3-us-west-2.amazonaws. com/openai-assets/researchcovers/languageunsupervised/ languageunderstandingpaper.pdf], and BERT [Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In NAACL-HLT, 2019]. These models can be used to represent words in the context of subsequent tasks.

LASER and Sentence BERT (SBERT) models are alternatives to word2vec. The main advantages of both models are that they are initially pre-trained on large amounts of data, and also support many different languages besides English.

LASER (Language-Agnostic Sentence Representations) [7] is a library developed by Facebook for calculating and using multilingual word embeddings. This model is trained in 93 languages, written in 23 different alphabets. This includes all European languages, many Asian and Indian languages, Arabic, Persian, Hebrew, and various minority languages and dialects. The sentence encoder of this model also supports language switching, that is, the same sentences can contain words in several different languages.

BERT[8] and RoBERTa [9] have established new leading-edge performance in sentence pair regression problems such as semantic textual similarity. However, this requires both proposals to be submitted to the network, which is computationally expensive: it takes about 50 million computations (~ 65 hours) with BERT to find the most similar pair in a collection of 10,000 sentences. The BERT construct makes it unsuitable for semantic similarity searches as well as unsupervised tasks such as clustering. The Sentence-BERT (SBERT)[10] model replaced BERT in the listed tasks, a modification of the pre-trained BERT network that uses Siamese and triplet networks to obtain semantically meaningful sentence attachments that can be compared using cosine similarity. This reduces the effort to find the closest match from 65 hours with BERT/RoBERTa to about 5 seconds with SBERT while maintaining BERT accuracy. In addition, SBERT can be used for certain new tasks that, until now, did not apply to BERT. These tasks include large-scale semantic similarity comparison, clustering, and information retrieval using semantic search. SBERT has set new leading performance for a variety of sentence classification and sentence pair regression problems.

3. Data Collection and Preprocessing

The first step in solving this problem is data collection. Since the goal is to search for academic disciplines for the development of professional skills and competencies of students, then the data set is information about all possible skills that students or employees in the field of information technology may have. To obtain this information, we used the capabilities of the application interface of the EMSI (Economic Modeling, LLC) project, an analytical company that uses data to identify patterns and trends in the labor market [11]. EMSI provides a full range of IT skills in English, including "Hard Skill" (28094 items) and "Soft Skill" (327 items) on the date of the experiment. "Hard Skill" contains the names of technologies (for example, the names of programming languages or technical terms: Java, web development, system administration), "Soft Skill" are analogous to personal qualities (coordination, time management, attention to detail, etc.), Table 1.

Table 1
Example of IT skills from EMSI

ID	Name	Type name
KS126XS6CQCFGC3NG79X	.NET Assemblies	Hard Skill
KS1200B62W5ZF38RJ7TD	.NET Framework	Hard Skill
KS1238D6SC8NVDVCSMRB	Amazon DynamoDB	Hard Skill
KS120NM6L2KDLC2Z9DND	Artificial Intelligence Systems	Hard Skill
ES7A07CF87C516875B23	Accountability	Soft Skill
ESCBF9FD5DB49E82C7C9	Basic Reading	Soft Skill
KS95IACBUORYQ0U7M90W	Calmness Under Pressure	Soft Skill
KS79O5AFPWQ4IEY9HCH5	Challenge Driven	Soft Skill

Before training the models, it is necessary to clean and preprocess the collected data: convert all letters to lowercase, remove numbers, extra spaces, special characters, punctuation marks, tokenize sentences and bring words to their initial form.

4. Application of Pre-Trained Models in the Problem of Finding Skills in the Descriptions of Educational Courses

4.1 Methodology

Let us describe the methodology for extracting professional skills from academic disciplines in Russian using multilingual pre-trained models. The technique includes the following steps:

1. Extracting all noun phrases (as the basis for describing skills) of a given length from the description of the academic discipline. To implement this, the praise machine library [12] was used, which allows extracting a list of noun phrases from a sentence. In addition, the pymorphy2 library [13] was used to implement support for the Russian language, and the gensim library was used for text preprocessing (tokenization, removal of stop words, etc.) [14].
2. Vectorization of phrases extracted from the academic discipline using a pre-trained model. The result of this step is a set of vectors of a given dimension (1024 for LASER and 512 for SBERT) corresponding to the phrases extracted in step 1.

3. Vectorization of the list of professional skills (EMSI) using a pre-trained model. The result of this step is a set of vectors of a given dimension corresponding to hard and soft skills.

4. Comparison of the vectors obtained in items 2 and 3 by the cosine measure of proximity - a measure of similarity between two non-zero vectors calculated as the cosine of the angle between them. The cosine measure of proximity the measure changes from 0 to 1. In NLP the closer this measure is to 1, the more similar the words (phrases, documents) are.

In this paper, we tested two pre-trained models LASER and SBERT. Since these models support many different languages, no translation of the list of skills into Russian is required.

4.2 Experiment

Let us give an example of searching for skills in the text using the LASER model. As input, the following text was passed:

"Features of the profession; setting up a computer and programs; work with data and lists; functions; Git; algorithms; Linux; frameworks; libraries; work with models and files; testing; teamwork; 3 projects. Teachers: Lev Lunev is a FullStack developer at the German company Innoscripta. Ivan Kuznetsov is an active Backend developer. After graduation, you will be able to: program in Python; create web programs and sites, applications; use frameworks and libraries; work with HTML, CSS, JavaScript; create complex web interfaces; work with databases; optimize unsuccessful applications; be part of a real project team; work with the Git version control system". This text was obtained from the description of the course "Fullstack developer in Python" (from SF Education and translated in English from Russian).

The search result is a list of skills, Table~2.

Table 2
Skills extracted from "Fullstack developer in Python" course description

Hard Skills	Soft Skills
Linux, Javascript, Python, Microsoft Windows XP	Team Leadership, Team Management, Team Building, Program Management, Mobile Devices, Web Browsers, Computer Literacy, Computer Terminals, Evaluating Staff, Calendaring Software, Teamwork, Contract Reviews, Application Development, Checklists, Web Conferencing, Personal Computers

We see that the most found skills are "soft" and what is much worse, the model could not find some of the "Hard skills" such as HTML and CSS.

4.3 Models Testing

To assess the accuracy of skill search using pre-trained models, test data were prepared. To do this, a list of noun phrases was extracted from the descriptions of the curriculum of the three disciplines, and a set of skills was manually selected as an "expected" result, a total of 259 skills. Such a large number is explained by the fact that several extracted phrases can correspond to the same skill; manual cleaning and filtering of such phrases was not carried out. Further, using LASER and SBERT models, the same test disciplines studying programs were analyzed and skills were extracted. The ratio of the number of "found" skills to the number of "expected" skills was taken as the test accuracy of the models, Table~3. The LASER model showed an accuracy of 42.54%, and the SBERT model showed an accuracy of 35.07%. To improve the search accuracy, the list of skills was also translated into Russian, the search accuracy was 51.53% for LASER and 36.27% for SBERT. From the results we can conclude that both models find less than half of the "expected" skills, this is since the phrases extracted from the curriculum can contain the names of skills in combination with other words, which reduces the degree of similarity when comparing vectors in step 4 of the described algorithm. It is also obvious that the accuracy of the models increases when the list of skills is coerced into the same language as the curriculum.

Table 3
Comparative analysis of search accuracy

Models	LASER, accuracy %	SBERT, accuracy %	# of found skills, LASER	# of found skills, SBERT
Skills without translation	42.54	35.07	110	91
Skills with translation	51.53	36.27	134	94

5. Building and Training a Model for Finding Skills in Course Descriptions

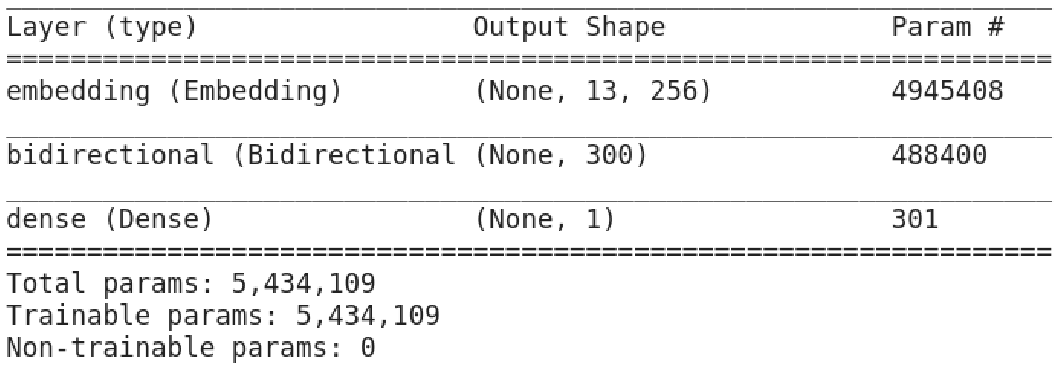
To search for skills in the descriptions of educational courses, a binary classification model was trained. A train data was containing 28423 “skill phrases” and 6976 “non-skill phrases”. To form a list of phrases that are not related to professional skills, the educational standard of the specialty "Computer Security" was taken, since the curricula are drawn up based on the educational standard and contain similar vocabulary. Noun phrases were extracted from this document using the phrase machine library [12]. After that, the extracted phrases were preprocessed, as described above, and vectorized using the LASER model, which showed the best result in the previous experiment, Table 3. Next, the phrases with which vectors had high cosine similarity (more than 0.9) with vectors of EMSI skills were removed; the remaining phrases were also “cleaned up” manually. After preparing the data we received a dataset of 28,423 skill phrases and 6976 non-skill phrases.

The constructed model is a neural network with the first Embedding layer with dimension 512, which transforms positive integers (indices) into dense vectors of fixed size as described in section 2.1.

The second layer of the model is a Bidirectional Long-Short Memory (LSTM) with 150 neurons. LSTM is a type of recurrent network with a gated structure to learn long-term dependencies of sequence-based tasks [15]. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell [16]. The Bidirectional LSTM processes sequence data in both forward and backward directions with two separate hidden layers and its architecture based on bidirectional RNN [17].

The last layer of the described model is a fully connected (Dense) layer for classification with one neuron and sigmoidal activation function [18]. The dense layer is the regular deeply connected neural network layer. It is the most common and frequently used layer [19]. Neurons of this layer are connected with all neurons of neighboring layers. Inside each neuron, a weighted sum of signals from the neurons of the previous layer is calculated. Then the activation function is applied and this value is the output of the neuron.

Figure 2 shows the structure of the network and the number of trained parameters. To train the neural network, the Adam optimizer [20] was used, which minimizes the binary error of cross-entropy with a learning rate of 0.01.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding (Embedding)        (None, 13, 256)           4945408   
_________________________________________________________________
bidirectional (Bidirectional (None, 300)               488400    
_________________________________________________________________
dense (Dense)                (None, 1)                 301       
=================================================================
Total params: 5,434,109
Trainable params: 5,434,109
Non-trainable params: 0
```

Figure 2: Model summary

Two models of the same architecture, described above, were trained on the data prepared in two ways:

1) the skills were divided into 2 subclasses – soft skills and hard skills. Hard skills have not been translated into Russian, since this list contains mainly the names of technologies, soft skills have been translated into Russian;
2) all skills were combined into one class, and both “soft” and “hard” skills were translated into Russian.

Models were trained on 70% of the data and validate on 30% of data. The accuracy of the classification model based on validation data for the first model reached 99.25% in five epochs, for the second model - 96.16% already in the second epoch, then the model started to overfit, the learning curves are shown in Figures 3-4. When testing models on new data and comparing “found” skills with a previously extracted list of “expected” skills, the accuracy of the models was 97.51% and 97.38%, respectively.

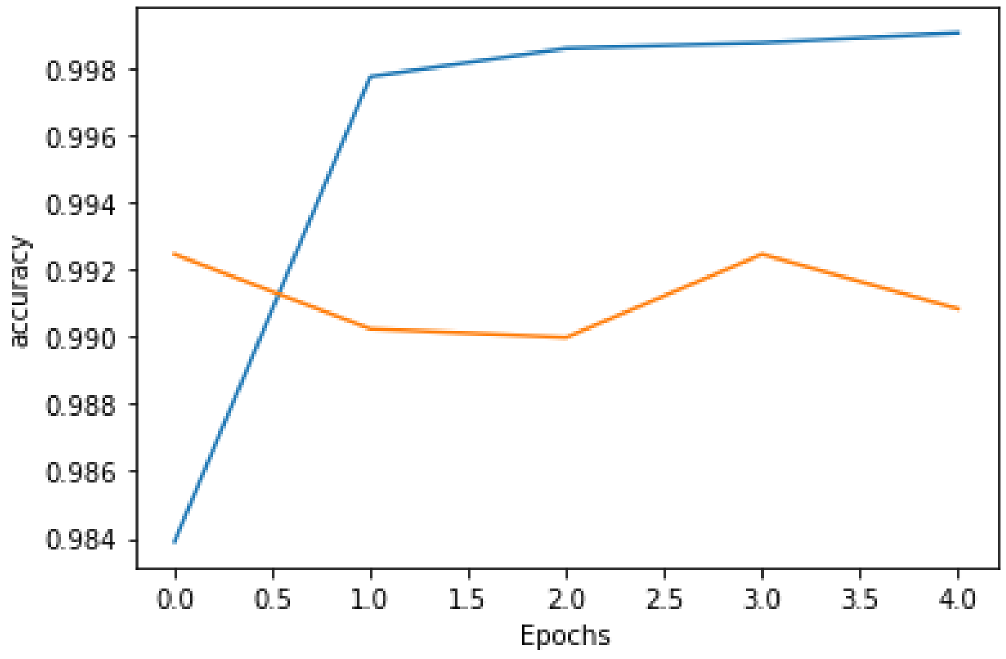


Figure 3: Learning curve for the first model

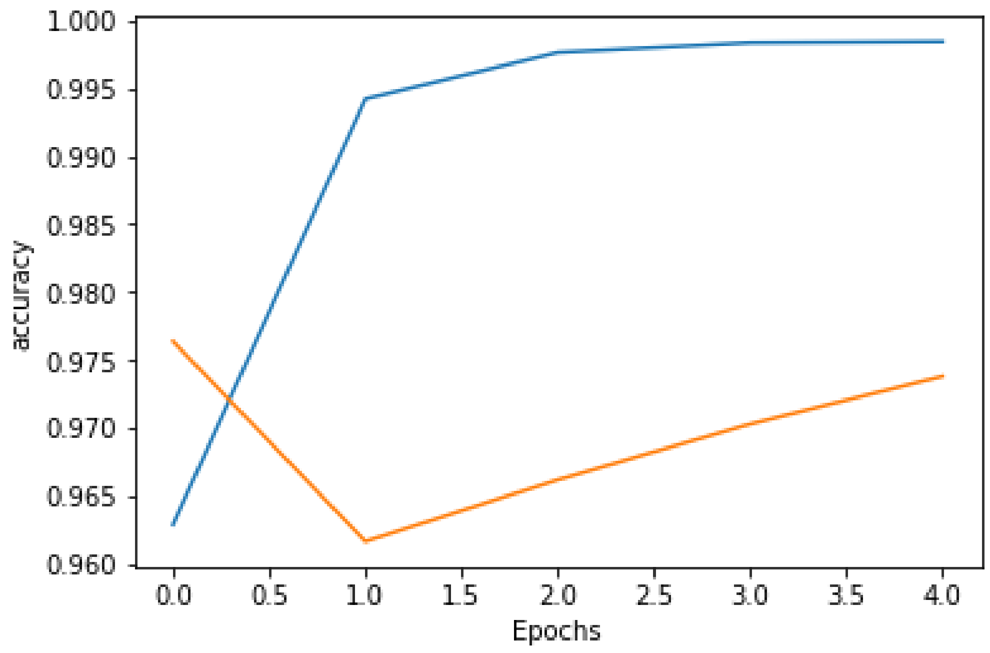


Figure 4: Learning curve for the srcond model

For the test data, confusion matrices A^1 (1) and A^2 (2) were built, which show how the models classify objects of different classes. A confusion matrix summarizes the classification performance of a classifier concerning some test data. It is a two-dimensional matrix, indexed in one dimension by the true class of an object and in the other by the class that the classifier assigns [21]. The matrix element a_{ij} shows the number of objects of class i that the model has classified as class j. For the first model, it can be seen that the third class, which corresponds to "soft" skills, is classified randomly: since this class is the smallest, the model is aimed at minimizing errors in more numerous classes. The second model is more likely to make mistakes in the "not skill" class, which is understandable, since the data were not balanced and the model seeks to minimize the error in a more numerous class, but in general, the error in the "not skill" class is not that big - a little more than 4%.

$$A^1 = \begin{pmatrix} 2321,20,4 \\ 175,9036,22 \\ 28,41,34 \end{pmatrix}. \quad (1)$$

$$A^2 = \begin{pmatrix} 2245,100 \\ 206,9130 \end{pmatrix}. \quad (2)$$

As an example, Figure 5 shows a "word cloud" built on the skills extracted from the description of the discipline "Machine Learning and Data Analysis" (in Russian) using a trained model and then translated into English. Skills extracted from the description of the academic discipline are presented on the "word cloud". The larger the word size, the more often this skill is mentioned in the description.

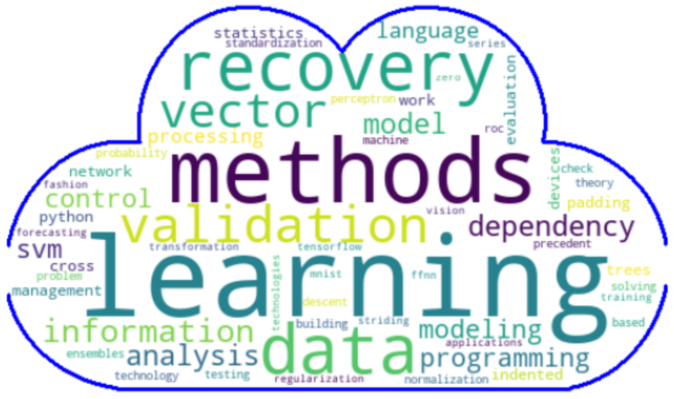


Figure 5: Extracted skills of the discipline “Machine Learning and Data Analysis”

6. Conclusions

The article describes the application of distributed vector representation models for the automatic extraction of professional skills and competencies from the descriptions of academic disciplines. The process of collecting and processing a dataset for training such models is described. A comparative analysis of the accuracy of the investigated models is carried out to identify the most accurate model. The process of building and training of neural network model for skills classification based on a bidirectional LSTM is described. The results are analyzed using confusion matrices. Further improvement of the accuracy of the developed neural network model is associated with replacing the embedding layer with pre-built vectors, for example, using the pre-trained LASER model, both with increasing the dataset for training and balancing it.

7. References

[1] E.V. Kosareva, Investigation of natural language processing methods and their application in job online-search, in: Scientific Collection «InterConf», (36): with the Proceedings of the 7th International Scientific and Practical Conference «Challenges in Science of Nowadays», November 26-28, 2020, Washington USA, pp. 21-27.

[2] K.V. Kosarava, N. V. Davydik, Topic modeling application for intellectual analysis of reviews in Russian, in: Selected Papers of the V International Scientific and Practical Conference "Distance Learning Technologies" (DLT 2020), Yalta, Crimea, September 22-25, 2020, CEUR-WS.org, online http://ceur-ws.org/Vol-2914/paper13.pdf.

[3] J. Camacho-Collados, M.T. Pilehvar, From word to sense embeddings: a survey on vector representations of meaning, Journal of Artificial Intelligence Research 63 (2018) 743 788. doi:10.1613/jair.1.11259.

[4] TensorFlow.org, Word embeddings, 2012. URL: https://www.tensorflow.org/text/guide/word_embeddings.

[5] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS'13), 2013 (2), pp. 3111 3119.

[6] X. Qiu, T. Sun, Y. Xu et al. Pre-trained models for natural language processing: A survey, Sci. China Technol. Sci. 63 (2020) 1872–1897. doi:10.1007/s11431-020-1647-3.

[7] Open-sourcing enhanced LASER library, Zero-shot transfer across 93 languages. URL: https://engineering.fb.com/2019/01/22/ai-research/laser-multilingual-sentence-embeddings/.

[8] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language

Technologies, Volume 1, Stroudsburg, PA, USA, 2019, pp. 4171–4186. doi:10.18653/v1/N19-1423.

[9] Y. Liu, et al., RoBERTa: A Robustly Optimized BERT Pretraining Approach, ArXiv abs/1907.11692 (2019).

[10] N. Reimers, I. Gurevych, Sentence-BERT: Sentence Embeddings using Siamese BERT – Networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, 2019. - p. 3982 3992.

[11] Emsi skills. URL: https://skills.emsidata.com/.

[12] Quickly extract multi-word phrases from a corpus. URL: https://github.com/slanglab/phrasemachine.

[13] Morphological analyzer pymorphy2 URL: https://pymorphy2.readthedocs.io/en/0.2/user/index.html.

[14] Gensim – Beginner's Guide. URL: https://webdevblog.ru/gensim-rukovodstvo-dlya-nachinajushhih/.

[15] S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural computation, volume 9 8(1997) 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[16] G. Van Houdt, C. Mosquera and G. Nápoles, A review on the long short-term memory model, Artif Intell Rev 53(2020) 5929–5955. DOI: 10.1007/s10462-020-09838-1.

[17] M. Schuster and K. K. Paliwal, Bidirectional recurrent neural networks, IEEE Transactions on Signal Processing, volume 45 (1997) 2673–2681. DOI: 10.1109/78.650093.

[18] Siddharth Sharma, Simone Sharma, and A. Athaiya, Activation functions in neural networks, International Journal of Engineering Applied Sciences and Technology, 4 (2020) 310-316. DOI: 10.33564/IJEAST.2020.v04i12.054.

[19] Keras – Dense Layer. URL: *https://www.tutorialspoint.com/keras/keras_dense_layer.htm*.

[20] D.P. Kingma and J. Ba. Adam, A Method for Stochastic Optimization, In: Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 2015. URL: *http://arxiv.org/abs/1412.6980*.

[21] K.M. Ting, Confusion Matrix, In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA, 2011. DOI: 10.1007/978-0-387-30164-8_157.