International Journal of Technology 11(6) 1213-1222 (2020)
Received October 2020 / Revised November 2020 / Accepted November 2020

International Journal of Technology

http://ijtech.eng.ui.ac.id

Developing and Testing an Algorithm to Identify Future Innovative Research Areas in Digitalization Conditions (using a Medical-sector example)

Irina Rudskaya[1], Ivan Ozhgikhin[1], Darya Kryzhko[1*]

[1]*Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Polytechnicheskaya, 29, 194064, Russia*

Abstract. Predicting more relevant areas of medical research, with a prediction period starting at five years, is currently done exclusively by experts, while the results of such forecasts are extremely ineffective and differ significantly, depending on their source. Modern development trends in world science require the creation of a universal forecasting tool that can be used as a basic resource—providing objective, system-independent information. This condition justifies the relevance of the present study. This study's goal was to develop and test an algorithm to identify future innovative research areas in digitalization conditions (using the medical sector as an example). During this research, a prognostic model was developed based on the hype cycle, which makes determining a list of possible areas for hardware development in the medical sector possible, presenting this list as a set of tokens that are decoded using mental analysis and automated through the Python programming language. The process of identifying future innovative research areas comprises the following stages: identifying the aggregator (hub) research results, parsing primary information, translating the analyzed information, forming a set of lexemes, forming an analytical dataframe, constructing regression models for the highlighted lexemes, forming and storing the resulting dataframe, and metathinking the highlighted lexemes. In total, 4,000 study names were analyzed, based on the ResearchGate platform, which made obtaining 28 significant lexemes based on the results of metathinking possible. Next, an associative map was created using the most promising research areas in medicine, namely: diagnosing viral infections, the spread of viral infections, coronaviruses, cardiovascular diseases, and lung diseases. The obtained algorithm for the automatic determination of promising research areas can be modified by choosing different sources of information.

Keywords: Development of medicine; Digitalization; Hardware; Hype cycle; Identification algorithm; Innovative research areas

1. Introduction

The research process in the field of medicine is one of the most complex research processes—both structurally and substantially—since it is characterized by the main participants' desire for system cooperation. However, the process of cooperation does not allow for a management of the forecast horizon, which involves identifying the most appropriate period of time for the implementation of the project. Accordingly, most specific research aims to solve current, not future, problems in medicine. At the same time, proposing innovative solutions for tasks that are not yet identified makes creating unique competitive advantages possible—the core of future markets.

*Corresponding author's email: darya.kryz@yandex.ru
doi: 10.14716/ijtech.v11i6.4438

One of the most common models describing the process of technological development is the Gartner hype cycle. In 1995, analysts from the Gartner company established that the development of a company that creates and introduces innovative technology to the market is characterized by the specific function of developing an information space—a fifth-degree polynomial that determines the five stages of developing the information background surrounding innovative solutions. In the information space of any stable trend described by the Gartner model, a field of research characterized by high commercial potential can be identified (O'Leary, 2008). This high commercial potential is defined by the prospect of a synergetic effect from the appearance not only of an innovative solution but also of its accompanying market and consumer infrastructure, which supplements and develops an opportunity to use this innovative solution (Kudryavtseva et al., 2017). Therefore, after identifying this research area during the hype cycle's earlier stages, a developer can create a potentially promising vector to develop their own research (Lyu et al., 2015). These specifics are most relevant to the development of medical equipment.

The model presented in this study clearly expresses functional specifics since each of its stages can be described by the certain function of a second- or third-degree polynomial (Dedehayir and Steinert, 2016). Consequently, if an information space is described by a system of polynomial functions and this system is itemized into its elementary components, determining the lexical categories characterized by one of the Gartner model's stages is possible.

2. Methods

First, we look at the algorithm's preparatory stage. The beginning of this stage involves choosing information sources. Since the subject of the current analysis is the information space of medical research, the basic resource could be an aggregator (hub) of research results. Several such aggregators exist on the internet—in particular, the international abstract databases Scopus and Web of Science, the electronic libraries ScienceDirect and eLibrary, and many other aggregators. However, the most suitable aggregator for the goal of automated research in the information space is the scientific social network ResearchGate, based on the properties of this resource described in the following paragraph.

Practically, any research work published anywhere can be indexed on ResearchGate. As a result of this feature, ResearchGate concentrates the research results presented in practically all of the aforementioned databases. ResearchGate does not require any additional peer review of the works placed on the platforms, and it allows free discussion in the comments section of any of these works. This feature also enables eliciting a reaction from the scientific and professional community to any research on the platform. ResearchGate does not require mandatory registration, and it allows searches without authentication. Therefore, automated algorithms to search for and collect information can interact with ResearchGate relatively stable compared to other platforms that aggregate research results. Of course, the site offers basic information protection and does not allow an open collection of information using the most basic automated algorithms. However, a simple evasion of this protection is examined below (Muscanell and Utz, 2017).

Thus, ResearchGate is the most effective source of information describing the medical research information space. When completing the preparatory stage of the proposed algorithm, clarifying information sources is necessary. ResearchGate includes a system of tags that each reflect a specific area of research. When researching the nature of medicine, the "Medicine" and "Disease" tags can be highlighted on ResearchGate. This set of tags reflects general information and includes the overwhelming majority of studies defining the current research area. Certainly, during the practical implementation of the proposed algorithm, this list can be added to and specified. However, during the theoretical development stage, this set of tags fully covers the need for resources to describe the relevant information space (Muscanell and Utz, 2017; O'Brien, 2019).

The research stage of the proposed algorithm is discussed next. This stage can be fully automated. This paper's framework proposes a management program in the Python programming language, which automates this stage (Hardeniya et al., 2016). The first step of implementing this algorithm is to collect the required primary information (Rahmatin et al., 2018). The name of a specific study contained in the database, in terms of the considered tags, acts as the primary information unit. For the appropriate tags, ResearchGate provides a maximum amount of information of 20 pages at 100 entries—which, in sum, gives 2,000 names of studies for each tag. The process of automating information collection online is called "parsing" (Martin et al., 1987). This process can be divided into a few stages:

- URL query sent. In this case, the basic queries are: "https://www.researchgate.net/topic/Disease/publications"; "https://www.researchgate.net/topic/Medicine/publications."
- Obtain and store the HTML code of the corresponding page.
- Decode the CSS markup on the stored pages.
- Search for the needed information using tags. Studies' names are stored in the tag "div" and the class "nova-e-text nova-e-text--size-l nova-e-text--family-sans-serif nova-e-text--spacing-none nova-e-text--color-inherit nova-v-publication-item__title."
- Store the necessary information (in this case, the studies' names) in an ordinal list.
- Delete the accompanying information (markup elements, references to resources, and so on) from the list elements.
- Continue to the next page and repeat the sequence of actions performed.

Each of the given tasks can be solved using a specific tool. Since a basic protection from automated information collection is installed on ResearchGate, modeling human behavior is necessary when sending a URL query. For this goal, a tool from the library "Selenium"—a webdriver—can be used. At the same time, following the principle of delay randomization is necessary; according to this principle, the human behavior online (following links, sending queries, et cetera) cannot be conditionally constant in time (Huang and Chang, 2016). To create delays, we propose using the "Sleep" tool from the "Time" library. For randomization, we suggest using the "Uniform" tool from the "Random" library. We determined empirically that the necessary range of delay randomization for an analyzed resource is 5.95–9.15 seconds. To decode the received HTML code and subsequently "clean" the results, we suggest using the "bs4" and "re" libraries. The described algorithm for parsing ResearchGate can be presented in the form of eight consecutive stages, presented in Table 1.

When working with other tags, the appropriate URL in Line 3.1 is replaced. Next, per the algorithm proposed above, the primary information is distributed in time. Since each study, per the ResearchGate architecture, includes a publication date in the "Month-Year" format, this stage can also be automated by integrating the following code into the algorithm presented in Table 1 between stages 7 and 8: "dates = usefull_inf.findAll('li', {'class':'nova-e-list__item nova-v-publication-item__meta-data-item'}) For date in dates: date_text = date.text, try_list.append(date_text)."

The next stage of the proposed algorithm gives the primary information a uniform form. In this case, the information is structurally uniform. However, we should mention that ResearchGate is an international resource and, therefore, many studies are presented in

different languages on the platform. We propose English to act as the basic language for the proposed algorithm. For automated translation, we propose using the "Translator" tool from the "Googletrans" library. This tool allows users to automatically send queries to the translator from Google and decode the received answers. Table 2 presents the algorithm for an automated translation of the analyzed information.
The next step of the proposed algorithm involves highlighting lexemes. This linguistic procedure can be divided into several stages, specifically (Nakayama, 1996; Hannebauer, et al., 2018):

- Divide the basic elements (in this case, the studies' names) into tokens—elementary lexemes.

Table 1 Algorithm for parsing primary information for the "Disease" tag on ResearchGate, using the Python programming language

Step	Code
1. Import the necessary libraries.	1. from bs4 import BeautifulSoup 2. import re 3. from selenium import webdriver 4. from selenium.webdriver.support.ui import WebdriverWait 5. import time 6. import random asrd
2. Open the browser.	1. chromedriver = 'way to webdriver' 2. options = webdriver.ChromeOprions() 3. options.add_argument('headless') 4. browser = webdriver.Chrome(executable_path=chromedriver, chrome_options=options)
3. Send a URL request and store the given page's HTML code.	1.browser.get('https://www.researchgate.net/topic/Disease/publications') 2. time.sleep(rd.uniform(5.95, 9.15)) 3. response = browser.page_sourse
4. Create a register and lists for data storage.	1. x=1 2. main_list = [] 3. try_list = []
5. Create a cycle for processing information.	1. while x <=20 ***The next code will appear inside the cycle***
6. Search and store studies' names.	1. useful_int = BeautifulSoup(response) 2. titles = useful_inf.findAll('div', {'class':nova-e-text nova-e-text—size-Inova-e-text—family-sans-serif nova-e-text—spacing-none nova-e-text—color-inherit nova-v-publication-item_title'})
7. Delete the metadata from studies' names.	1. for title in titles: 1) title_text=re.sub(r'\<[^>*\>', ", str(title)) 2) main_list.append(title_text)
8. Go to a new page.	1. x+=1 2. if x<=11: 1) button = browser.find_element_by_xpath('//*[@id="lite-page"]/main/section[3]/div/div[2]/div/div[101]/div/div/ div[1]/div/ div[12]/a') 2) button.click() 3) time.sleep(rd.uniform(5.95, 9.15)) 4) response=browser.page_sourse 3. else: 1) button = browser.find_element_by_xpath('//*[@id="lite-page"]/main/section[3]/div/div[2]/div/div[101]/div/div/ div[1]/div/ div[13]/a') 2) button.click() 3) time.sleep(rd.uniform(5.95, 9.15)) 4) response=browser.page_sourse

- Lemmatize the highlighted tokens. This procedure involves presenting the tokens in a single lexical form in order to exclude semantic duplicates. The final list of tokens can already be characterized as a list of lexemes.
- Exclude lexemes with low semantic content—numbers, punctuation marks, prepositions, et cetera.
- Create a single set of lexemes that is suitable for analysis.

The stages described can be automated using tools from the "NLTK" and "re" libraries. Likewise, note that the final stage of this algorithm requires the formation of a single set of lexemes. This set is a continuous text containing the appropriate lexemes.

Table 2 Algorithm for automated translation of the analyzed information

Step	Code
1. Import the necessary libraries.	1. *from googletrans import Translator* 2. translator = Translator()
2. Create a list to store new studies' names. Create a cycle for an exhaustive search of studies' names.	1. *trans_main_list = []* 2. *for sentens in main_list::* ***The next code will appear within the cycle***
3. Identify the original language of the studies' names.	1. *time.sleep(rd.uniform(0.08,0.35))* 2. *language = translator.detect(sentens)*
4. Create a translation function for the names presented in languages other than English.	1. *if language!='en':* 1) *new_sentens=translator.translate(sentens)* 2) *trans_main_list.append(new_sentens.text)* 2. else: 1) *trans_main_list.append(sentens)*

The next stage of the proposed algorithm involves calculating the mentioned lexemes' frequency and creating an analytical dataframe according to which a mentioned lexeme's frequency corresponds to an appropriate period. For these periods, using a month is advisable since the ResearchGate architecture does not identify a study's exact publication date in the research results. To calculate the frequency and form a dataframe, we suggest using the "pandas" and "sklearn.feature_extraction.text" libraries. We must mention that, in the created dataframe, months act as an ordinal variable and are subsequently converted into a sequence from 1 to n, where n is the number of months. Table 4 presents the algorithm for forming an analytical dataframe.

Table 3 Algorithm for forming a set of lexemes

Step	Code
1. Import the necessary libraries.	1. *import* re 2. *from nltk import pos_tag* 3. *from nltk.tokenize import word_rokenize* 4. *from nltk.stem.snowball import SnowballStemmer* 5. from nltk.stem import WordNetLemmatizer 6. lemmatizer = WordNetLemmatizer
2. Form lists to store the intermediary results.	1. *blok_of_tokens = []* 2. *blok_of_sentens = []* 3. *blok_of_sentens_2 = []*
3. Create a cycle for processing studies' names.	1. *for sentens in trans_main_list:* ***The next code will appear within the cycle until the seventh stage (inclusive)***
4. Separate studies' names into tokens.	1. *tokens = word_tokenize(sentens)*
5. Lemmatize the highlighted tokens.	1. *tokens_2 = [lemmatizer.lemmatize(word, pos='v')for word in tokens]*
6. Exclude the lexemes with low semantic content.	1. *tokens_3 = [re.sub(r'\Wl\dl\s', ", word) for word in tokens_2]*
7. Create an intermediary list of lexemes.	1. tokens_4 = [] 2. for w in tokens_3: 1) w!=": 1) *tokens_4.append(w)* 3. *blok_of_sentens.append(tokens_4)*
8. Create a single set of lexemes.	1. for sentens in blok_of_sentens: 1) *full_sentense = ' '.join(sentens)* 2) *blok_of_sentens_2.append(full_sentense)*

During the next stage of the proposed algorithm, a regression model is formed for each of the lexemes. To construct this regression model, we suggest using the least squares method. However, the use of this method is possible exclusively for building a linear regression model—but, for the present study, an approximation of the data with an exponential function is necessary. If the experimental dependency is nonlinear, it can be made linear by replacing the variables (Rudskaya and Rodionov, 2017). This approach is called linearization of functional dependencies. The linearization of an exponential functional dependency is presented in Figure 1.

Table 4 Algorithm for forming an analytical dataframe

Step	Code
1. Import the necessary libraries.	1. *import numpy as np* 2. *import pandas as pd* 3. *from datetime import datetime* 4. *from sklearn.feature_extraction.text import Count Vectorizer* 5. *count = CountVectorizer()*
2. Create a list of lexemes.	1. *anal_data=np.array(blok_of_sentens_2)*
3. Calculate the frequency of the mentioned lexemes and form lists.	1. *dag_of_words=count.fit_transform(anal.data)* 2. *anal.array = dag_of_words.rorray()* 3. *coll_names=count.get_feature_names()*
4. Create a dataframe.	1.*data=pd.DataFrame(anal_array,columns=coll_names))*
5. Transform indices of dataframe lines into dates.	1.*data.index=pd.to_datetime(try_list[0:len(trans_main_list)])*
6. Create a multi-index and group the data according to publication months.	1. *data ['Year']=data.index.year* 2. *data['Mon']=data.index.month* 3. *dates=data[['Year', 'Mon']]* 4.*data.index=pd.MultiIndex.from_tuples(dates.values.tolist(), names=dates.columns)* 5. *year_day_mean=data.groupby(level=[0,1].sum()* 6. *year_day_mean.drop('Year', 'Mon'],inplace=True,axis=1)*

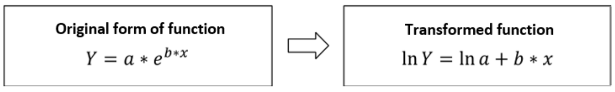

Figure 1 Linearization of exponential functional dependency

Per rule of linearization of functional dependencies a transformation of the frequency into its natural logarithm is necessary. We suggest eliminating zero values by a frequency offset of one. The formed dataframe can be transformed using a built-in function in the "NumPy" library, specifically: "analitic_data = np.log((year_day_mean+1))."

Based on the formed dataframe, corresponding regression models can be constructed for each of the highlighted lexemes. For this goal, the library "statsmodels.api" is used. Since the only analyzed parameter of the resulting models is the coefficient of determination, immediately considering for each model's coefficient and storing it in the appropriate list is advisable. Table 5 shows the algorithm for constructing regression models for the highlighted lexemes.

Completing the second stage of the proposed algorithm requires the formation of a final dataframe containing a structured list of tokens and their corresponding coefficients of determination. To solve this task, we propose using the "csv" library, which allows the formed dataframe to be stored in a file of the appropriate format.

The dataframe obtained from the results of the proposed algorithm contains the necessary and sufficient amount of information for the goals of further analysis. The final, analytical stage of the algorithm includes sequential mental processing of the obtained result (Muslim and Riansa, 2017). First, the primary selection of tokens takes place according to their respective coefficients of determination. Per the Cheddock scale, the minimum acceptable value of a coefficient of determination is 0.5. Consequently, at this stage, all lexemes whose models have coefficients of determination less than this boundary value are excluded from the set.

The following stage of analysis involves metathinking the highlighted lexemes. In this case, each of the lexemes is conceptualized in the context of research in developing medical equipment. During this stage, the lexemes can be divided into two basic groups:

1. *Conditionally conceptualized in the context of research in developing medical equipment:* This group includes lexemes that are directly and indirectly related to studies in developing medical equipment.
2. *Conditionally non-conceptualized in the context of research in developing medical equipment:* This group includes lexemes that are removed from or completely contextually unrelated to studies in developing medical equipment, as well as general lexemes that can relate to virtually any research area.

Table 5 Algorithm for constructing regression models for the highlighted lexemes

Step	Code
1. Import the necessary libraries.	1. *import statsmodels.api as sm*
2. Create independent variables.	1. *X = list(range(1, len(analitic_data)+1))*
3. Create dependent variables.	1.. *Y_list = analitic_data.columns*
4. Make a list to store the models' coefficients of determination.	1. *models_list = []*
5. Create a cycle for an exhaustive search of lexemes.	1. *for tok in Y_list* ***The next code will appear within the cycle***
6. Select dependent variables.	1. y= *analitic_data[tok]*
7. Create a constant.	1. *X = sm.add_constant(X)*
8. Build a regression model.	1. *odel = sm.OLS(y, X).fit())*
9. Calculate and store the corresponding model's coefficient of determination.	1. *models_list.append(model.rsquared)*

Table 6 presents an algorithm for forming and storing the resulting dataframe.

Table 6 Algorithm for forming and storing the resulting dataframe

Step	Code
1. Import the necessary libraries.	1. *import csv*
2. Create a dataframe.	1. *frame = zip(analitic_data.columns, models_list)*
3. Create a file.	1. with open ('Lex.csv', "w") asf: ***The next code will appear within this command***
4. Create a command to write data.	1. *writer = csv.writer(f)*
5. Write column names.	1. *writer.writerow(('lexeme', 'coefficients of determination'))*
6. Write dataframe data in a file.	1. fir row in frame: 1. *writer.writerow(row)*

The set of lexemes obtained from the results of this refinement is the basis for identifying the most promising areas of research in the realm of medical hardware development. To form a set of the most significant lexemes and research theses, building an associative map that joins all the highlighted lexemes into research areas is necessary. These comprehensive lexemes are research theses upon which a set of the most promising future research areas can be based.

The methodology developed here by the present authors is based on Python libraries' tools and adapted to the ResearchGate architecture. The proposed algorithm enables an expansion of the information-parsing mechanism due to the modulating elements of human behavior, translating the collected information into the selected base language (English), and presenting the obtained data in a single form that is convenient for further analysis. In addition, we emphasize the importance of supplementing any analysis of the collected tokens with the process of dividing the tokens into two basic groups: the first group of data comprises lexemes that directly and indirectly relate to a study, while the second group comprises lexemes that relate to virtually any research area. This process enables an increase in the degree of universality of an application of the proposed algorithm.

Thus, the proposed algorithm uses objective information as its research basis, which allows subjectivity to be avoided in its analytical results. Since this part of the algorithm is completely automated, it can be used inclusively by subject-matter specialists without special knowledge in statistics and data analysis. The proposed algorithm can be tested.

3. Results and Discussion

The proposed algorithm was applied to the sections of ResearchGate proposed above. As a result, 4,000 studies' names were processed, enabling the algorithm to obtain 5,733 pairs of "lexeme–coefficient of determination" pairs during the present research. In the first phase of the proposed algorithm's analytical stage, this sample was reduced to 278 pairs, according to the coefficients of determination. Based on metathinking results, 28 significant lexemes were highlighted, which are presented in Figure 2.

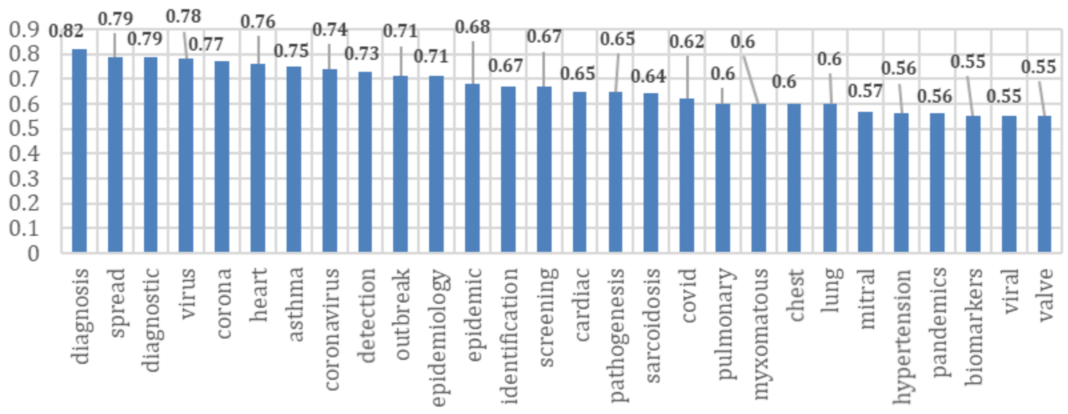

Figure 2 Basis for identifying the most promising research areas in medical hardware development

As can be seen on the Figure 2, the coefficients of determination for the corresponding lexemes exceed 0.5, satisfying the previously formed limits. An in-depth thematic analysis of the highlighted lexemes enables the formation of the following global research areas: the diagnosis of viral infections, the spread of viral infections, coronaviruses, cardiovascular diseases, and lung diseases. The first three highlighted research areas can be consolidated into a single area—viral diseases. Figure 3 maps the thematic connection among the lexemes highlighted in this research area.

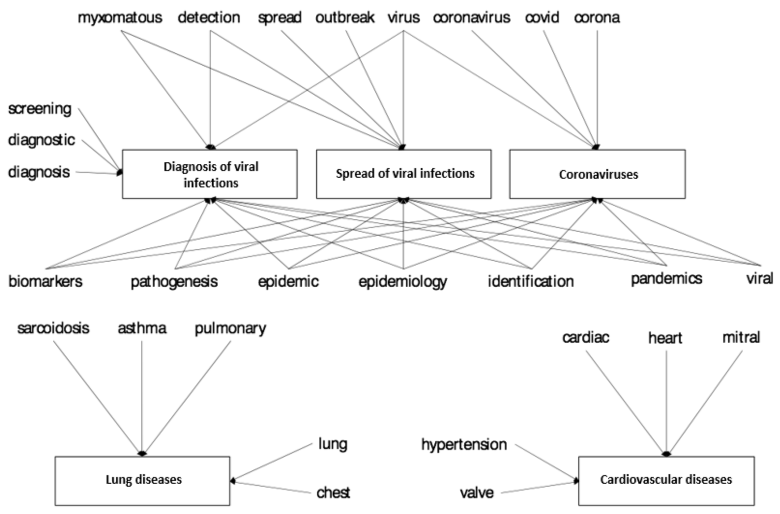

Figure 3 Map of research areas' thematic connections

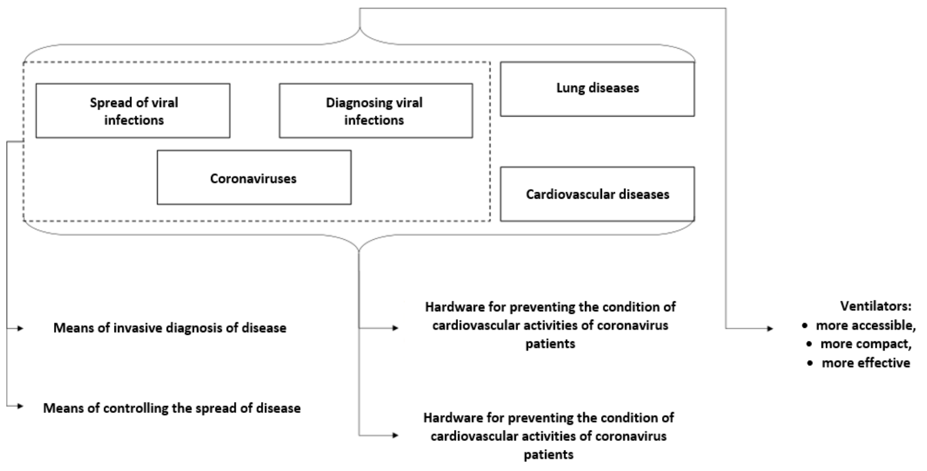
Spread of viral infections
Diagnosing viral infections
Coronaviruses
Lung diseases
Cardiovascular diseases
Means of invasive diagnosis of disease
Means of controlling the spread of disease
Hardware for preventing the condition of cardiovascular activities of coronavirus patients
Hardware for preventing the condition of cardiovascular activities of coronavirus patients
Ventilators:
more accessible,
more compact,
more effective

Figure 4 Set of the most promising research areas in medical hardware development

Thus, the developed algorithm and its instrumentation and software offer application significance, and they can be used by research organizations and medical equipment developers to determine the vector of development in the medical hardware development research process.

4. Conclusions

In conclusion, we can add that today, the trend of developing personalized medicine has made producers' ability to design and launch (often in small batches) medical technology that can complete clearly defined tasks, when applied to a specific medical method, more and more important. Meanwhile, the process of determining the most promising research areas in developing medicine can be algorithmized and automated. This study proposed a forecasting model to form a set of the most promising research areas in medical hardware development. This model was based on fractionation of the current research context according to the hype cycle, which differs from alternative models by minimizing expert evaluation components and forming a vector of current studies, based on objective information. Based on this model, an algorithm was developed for identifying the most promising research areas in medical hardware development, presented as a set of tokens that were decoded using mental analysis and automated using the Python programming language. This forecasting model's results established that the most promising research lies in the following areas: diagnosing viral infections, the spread of viral infections, coronaviruses, cardiovascular diseases, and lung diseases.

Acknowledgements

The reported study was funded by the Russian Foundation for Basic Research (RFBR), according to research project № 18-310-20012.

References

Dedehayir, O., Steinert, M., 2016. The Hype Cycle Model: A Review and Future Directions. *Technological Forecasting and Social Change*, Volume 108, pp. 28–41

Hannebauer, C., Hesenius, M., Gruhn, V., 2018. Does Syntax Highlighting Help Programming Novices*? Empirical Software Engineering,* Volume 23(5), pp. 2795–2828

Hardeniya, N., Perkins, J., Chopra D., Joshi, N., Mathur I., 2016. Natural Language Processing: Python and NLTK. Packt Publishing Ltd.

Huang, M.-H., Chang, C.-P., 2016. A Comparative Study on Three Citation Windows for Detecting Research Fronts. *Scientometrics*, Volume 109(3), pp. 1835–1853

Kudryavtseva, T.J., Ivanova, E.A., Kozlova, E.A., Skhvediani, A.E., 2017. Pricing and Assessment of Competitiveness of Innovative Medical Devices in the Context of Commercialization Strategy. *Academy of Strategic Management Journal*, Volume 16(Special Issue 1), pp. 110–122

Lyu, P.-H., Yao, Q., Mao, J., Zhang, S. J., 2015. Emerging Medical Informatics Research Trends Detection Based on MeSH Terms. *Informatics for Health and Social Care*, Volume 40(3), pp. 210–228

Martin, W.A., Church, K.W., Patil, R.S., 1987. Preliminary Analysis of a Breadth-First Parsing Algorithm: Theoretical and Experimental Results. *Natural Language Parsing Systems*. Springer, pp. 267–328

Muscanell, N., Utz, S., 2017. *Social Networking for Scientists: An Analysis on How and Why Academics Use ResearchGate*. Online Information Review. Emerald Publishing Limited

Muslim, E., Riansa, I., 2017. Analytic Hierarchy Process (AHP) Pairwise Matrix with One Missing Value. *International Journal of Technology*, Volume 8(7), pp. 1356–1360

Nakayama, T., 1996. Method and Apparatus for Highlighting and Categorizing Documents Using Coded Word Tokens. Google Patents

O'Brien, K., 2019. ResearchGate. *Journal of the Medical Library Association: JMLA*, Volume 107(2), p. 284

O'Leary, D.E., 2008. Gartner's Hype Cycle and Information System Research Issues. *International Journal of Accounting Information Systems*, Volume 9(4), pp. 240–252

Rahmatin, N., Santoso, I., Indriani, C., Rahayu, S., Widyaningtyas, S., 2018. Integration of the Fuzzy Failure Mode and Effect Analysis (Fuzzy FMEA) and the Analytical Network Process (ANP) in Marketing Risk Analysis and Mitigation. *International Journal of Technology*, Volume 9(4), pp. 809–818

Rudskaya, I., Rodionov, D., 2017. Econometric Modeling as a Tool for Evaluating the Performance of Regional Innovation Systems (with Regions of the Russian Federation as The Example). *Academy of Strategic Management Journal*, Volume 16(2).