UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga
(Final project paper)

Napovedanje bodočega razvoja znanstvenih raziskav na podlagi znanstvenih publikacij

(Predicting the future development of scientific research based on scientific publications)

Ime in priimek: Hasan Mahmutagić
Študijski program: Računalništvo in informatika
Mentor: doc. dr. Branko Kavšek

Koper, september 2021

Ključna dokumentacijska informacija

Ime in PRIIMEK: Hasan MAHMUTAGIĆ

Naslov zaključne naloge: Napovedovanje bodočega razvoja znanstvenih raziskav na podlagi znanstvenih publikacij

Kraj: Koper

Leto: 2021

Število listov: 39 Število slik: 9 Število tabel: 3

Število referenc: 15

Mentor: doc. dr. Branko Kavšek

Ključne besede: računalništvo, strojno učenje, rudarjenje podatkov, analiza podatkov, podatkovna znanost

Izvleček:
V zaključni nalogi želimo predstaviti prihodnost in razvoj znanstvenih tem s področij "računalništvo", "strojno učenje", "podatkovno rudarjenje" in "analiza podatkov". Raziskava temelji na znanstvenih publikacijah iz zadnjih dveh stoletij, ki so združene v eno ogromno bazo podatkov Microsoft Academic Graph (MAG). V zaključni nalogi bomo analizirali to bazo podatkov (MAG) in z metodami strojnega učenja ugotovili, katere teme so trenutno najbolj priljubljene. Poleg tega bomo poskušali zgraditi napovedne modele, da bi natančneje določili prihajajoče teme in teme, ki bodo potencialno postale priljubljene v naslednjih petih, desetih ali več letih. Za izdelavo takšnih napovednih modelov bomo uporabili razpoložljive podatke iz že objavljenih raziskovalnih člankov (tema, avtor, založnik, univerza, število citatov, povzetek).

Key words documentation

Name and SURNAME: Hasan MAHMUTAGIĆ

Title of final project paper: Predicting the future development of scientific research based on scientific publications

Place: Koper

Year: 2021

Number of pages: 39 Number of figures: 9 Number of tables: 3

Number of references: 15

Mentor: Assist. Prof. Branko Kavšek, PhD

Keywords: computer science, machine learning, data mining, data analysis,data science

Abstract:
In the final project paper we want to present the future and development of scientific topics in the fields of “computer science”, “machine learning”, “data mining” and “data analysis”. The research is based on scientific publications from the last two centuries that are combined into one huge Microsoft Academic Graph (MAG) database. In the final project paper we will analyze this MAG database and use machine learning methods to identify which topics are currently the most popular. Moreover, we will try to build prediction models in order to pinpoint emerging topics and topics that will potentially become popular in the next five, ten or more years. We will use the available data from already published research papers (topic, author, publisher, university, number of citations, abstract, ...) in order to build such predictive models.

Acknowledgement

First of all, I would like to thank my mentor and professor Branko Kavesko for everything from the very beginning, the choice of topic, his guidelines, suggestions, planning of this final work.
I would like to thank my friends and special thanks to Mirza Krbezlija, Đorđe Klisura, Aleksandar Avdalović for their selfless help during their studies, for influencing my view of science and thus for my academic development.
And last but not least, I want to thank my family who have been my support and support all along. Especially my parents who supported me in all possible ways and enabled me to achieve all this.

List of Contents

List of Tables

List of Figures

List of Abbreviations

i.e.	that is
e.g.	for example
etc.	and other similar things
MAG	Microsoft Academic Graph
OAG.	Open Academic Graph
JSON	JavaScript Object Notation
CSV.	Comma-Separated values

1 Introduction

1.1 Problem description and goal

The past has revealed to me the structure of the future. (Pierre Teilhard De Chardin)
"Prediction" refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome, such as whether or not a customer will churn in 30 days. The algorithm will generate probable values for an unknown variable for each record in the new data, allowing the model builder to identify what that value will most likely be.
The word "prediction" can be misleading. In some cases, it really does mean that you are predicting a future outcome, such as when you're using machine learning to determine the next best action in a marketing campaign. Other times, though, the "prediction" has to do with, for example, whether or not a transaction that already occurred was fraudulent. In that case, the transaction already happened, but you're making an educated guess about whether or not it was legitimate, allowing you to take the appropriate action.
Machine learning model predictions allow businesses to make highly accurate guesses as to the likely outcomes of a question based on historical data, which can be about all kinds of things – customer churn likelihood, possible fraudulent activity, and more. These provide the business with insights that result in tangible business value. For example, if a model predicts a customer is likely to churn, the business can target them with specific communications and outreach that will prevent the loss of that customer.
Because of the pace at which the world is going, it is difficult to imagine what things will look like in the future. Part of that fast pace that is difficult to keep up with is scientific research. Therefore, it is almost impossible to predict in which direction all this will move and which areas will be popular. We said at the outset that the past can reveal the structure of the future. We just want to use the past and development to reach the goal. The quality and quantity of digitized data is huge, so we have the opportunity to develop modeling of all these scientific publications with great accuracy and efficiency.
The aim of this thesis is to predict which scientific topics will be popular in the fu-

ture, based on already existing scientific articles. We used Microsoft Academic Graph (MAG) as a base with scientific articles. For it, we have created a model of machine learning so that based on the given parameters it predicts topics that will be popular in the future. Using clustering methods, topics are divided into groups based on citations and data from the article.

The model that does such a thing was made and implemented by Adrian Mladenic Grobelnik, Dunja Mladenic, Marko Grobelnik on the topic of The Next Big Thing In Science (see [4]). They used the OAG (Open Academic Graph) database which they filtered and obtained 2.5 million articles which they used for their model. In the end, they got a model that has 74.3% accuracy in predicting which topics will be current in the future.

What we want is to expand the number of articles and areas in which we will train our models.

The rest of this paper is structured in the last way. Section 2 is Preliminaries where we will get acquainted with Machine learning, Python and its libraries for Machine learning, and finally about Algorithms its time complexities and space complexities. In Section 3, we will discuss the known developments, results, and scientific considerations to date. Section 4 describes the development models, description, and analysis algorithms we used, and finally presents the final results we obtained. Section 5 provides a conclusion.

2 Preliminaries

2.1 Machine learning

Machine learning is a data analysis method that automates the creation of analytical models. It is a branch of artificial intelligence based on the idea that systems can learn from data, recognize patterns, and make decisions with minimal human intervention.
The evolution of machine learning Due to new computer technologies, machine learning today is not like machine learning from the past. It arose from pattern recognition and the theory that computers can learn without programming to perform certain tasks; researchers interested in artificial intelligence wanted to see if computers could learn from data. The iterative aspect of machine learning is important because the models are exposed to new data and they can adapt themselves. They learn from previous calculations to produce reliable, repeatable decisions and results. It is a science that is not new - but one that has gained new momentum.
Although many machine learning algorithms have been around for a long time, the ability to automatically apply complex mathematical calculations to big data — over and over again, faster and faster — is a recent development. Here are some widely published examples of machine learning applications you may be familiar with:
-Self-driving Google car -Offers online recommendations like the ones we get from Amazon and Netflix
Why is machine learning important? The renewed interest in machine learning arises because of the same factors that have made data research and Bayesian analysis more popular than ever before. Things like the ever-increasing amount and variety of data available, computer processing that is cheaper and more powerful, and affordable data storage.
All of these things mean that it is possible to quickly and automatically produce models that can analyze larger, more complex data and provide faster, more accurate results - even on a very large scale. And by building precise models, an organization has a better chance of recognizing profitable opportunities - or avoiding unknown risks.
What is needed to create good machine learning systems? -Possibility of data preparation. -Algorithms -Automation and iterative processes -Scalability -Modeling
Who uses it? Most industries that work with large amounts of data have recognized the

value of machine learning technology. By gathering insights from this data - often in real time - organizations can work more efficiently or gain an advantage over competitors.
-Financial services -Government -Health care -Retail -Oil and gas -Transport
What are the popular methods of machine learning? The two most commonly adopted machine learning methods are supervised learning and unsupervised learning - but there are other machine learning methods. Here is an overview of the most popular types.
Supervised learning algorithms are practiced on labeled examples, such as an input where the desired output is known. For example, a piece of equipment might have data points labeled “F” (failed) or “R” (works). The learning algorithm receives a set of inputs along with the corresponding correct outputs, and the algorithm learns by comparing its actual output with the correct outputs to find errors. It then modifies the model accordingly. Through methods such as classification, regression, prediction, and gradient amplification, supervised learning uses patterns to predict tag values on additional unlabeled data. Supervised learning is typically used in applications where historical data predicts probable future events.
Unattended learning is used against data that does not have historical markings. The system is not told the "correct answer". The algorithm must understand what is being displayed. The goal is to research the data and find some structure in it. Unattended learning works well on transactional data. For example, it can identify customer segments with similar attributes that can then be treated similarly in marketing campaigns. Or it can find the main attributes that separate customer segments from each other. Popular techniques include self-organizing maps, mapping the nearest neighbor, k-means grouping, and decomposing an individual value. These algorithms are also used to segment text topics, recommend items, and recognize offline data.
Semi-supervised learning is used for the same applications as supervised learning. But it uses both tagged and unmarked data for training - usually a small amount of tagged data with a large amount of unmarked data (because unmarked data is cheaper and requires less effort to collect). This type of learning can be used with methods like classification, regression, and prediction. Supervised learning is useful when the costs associated with labeling are too high to allow for a fully marked training process. Early examples of this include recognizing a person’s face on a webcam.
Reinforced learning is often used for robotics, games, and navigation. By learning reinforcement, the algorithm uses trial and error to discover which actions bring the greatest rewards. This type of learning has three primary components: the agent (student or decision maker), the environment (everything the agent communicates with) and actions (what the agent can do). The goal is for the agent to select actions that maximize the expected reward over time. The agent will reach the goal much

faster by following a good policy. So the goal in learning reinforcements is to learn the best politics.
What are the differences between data mining, machine learning and deep learning? Although all these methods have the same goal - to extract data that can be used for decision making, they still have different approaches and abilities.
Data mining Data mining can be considered as a set of many different methods for gaining insight from data. This could include traditional statistical methods and machine learning. Data mining applies methods from many different areas to identify previously unknown patterns from data. These may include statistical algorithms, machine learning, text analytic, time series analysis, and other areas of analytic. Data mining also involves the study and practice of data storage and data manipulation.
Machine learning The main difference with machine learning is that, just like statistical models, the goal is to understand the structure of data - to adapt theoretical distributions to data that is well understood. So, with statistical models, there is a theory behind the model that is mathematically proven, but it requires that the data also meet certain strong assumptions. Machine learning has evolved based on the ability to use a computer to examine data for a structure, even if we don't have a theory of what that structure looks like. The test for the machine learning model is an error in checking new data, not a theoretical test that proves a null hypothesis. Because machine learning often uses an iterative approach to learning from data, learning can be easily automated. Passages are passed through the data until a robust pattern is found.
Deep learning In-depth learning combines advances in computing power and special types of neural networks to learn complex patterns in large amounts of data. Deep learning techniques are currently the most modern for recognizing objects in pictures and words in sounds. Researchers are now seeking to apply these successes in pattern recognition to more complex tasks such as automatic language translation, medical diagnoses, and a number of other important social and business issues. For more reading on these topics, refer to [7].
Machine learning has several different methods and some of them are:

- Regression
- Classification
- Clustering
- Dimensionality Reduction
- Ensemble Methods

- Neural Nets and Deep Learning
- Transfer Learning
- Reinforcement Learning
- Natural Language Processing
- Word Embeddings

See more about Machine learning methods in [15].

2.1.1 Classification

If you're looking to automate a classification task, your algorithm's job is to create a recipe that separates the data, like so:

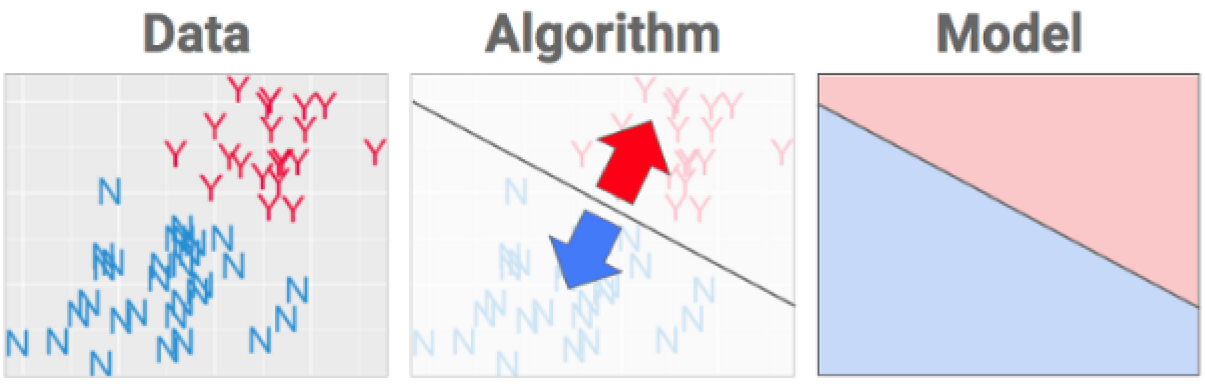


Figure 1: Classification example

What we have here is a dataset that is labeled with two classes (Y and N). In other words, the algorithm must put a fence in the data to best separate the Ys from the Ns. Our goal is to create a model that can be used to classify new examples correctly:

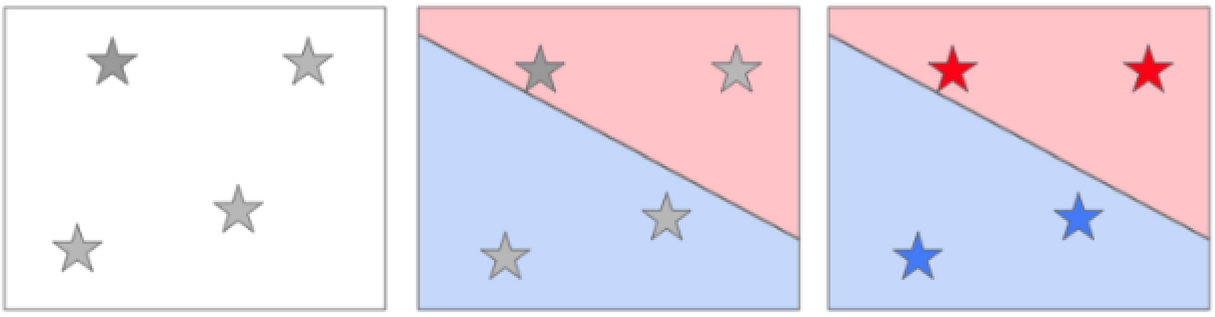


Figure 2: Classification example "where" the best place for that fence is.

The algorithm defines "what" shape will get fitted to the data. The data is used to choose "where" the best place for that fence is.

2.1.2 Regression

Regression analysis consists of a set of machine learning methods that allow us to predict a continuous outcome variable (y) based on the value of one or multiple predictor variables (x).

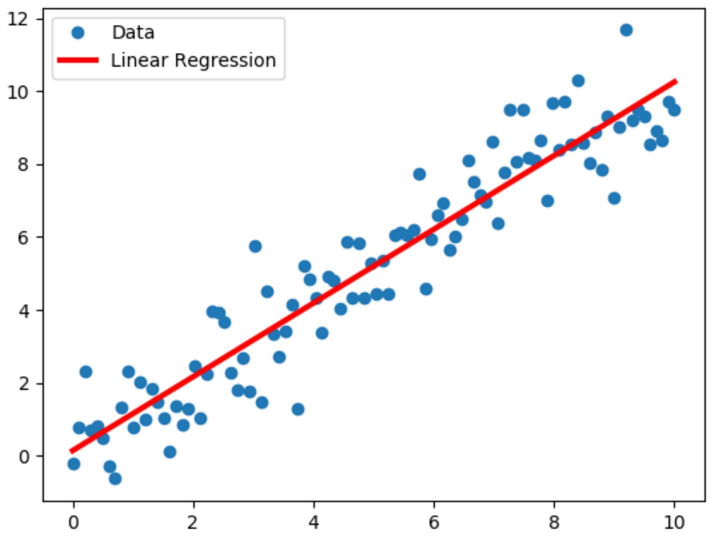


Figure 3: Linear regression example

Briefly, the goal of regression model is to build a mathematical equation that defines y as a function of the x variables. Next, this equation can be used to predict the outcome (y) on the basis of new values of the predictor variables (x). Linear regression is the most simple and popular technique for predicting a continuous variable. It assumes a linear relationship between the outcome and the predictor variables. The linear regression equation can be written as y = b0 + b*x + e, where:

- b0 is the intercept
- b is the regression weight or coefficient associated with the predictor variable x.
- e is the residual error

Technically, the linear regression coefficients are detetermined so that the error in predicting the outcome value is minimized. This method of computing the beta coefficients is called the Ordinary Least Squares method.
When you have multiple predictor variables, say x1 and x2, the regression equation can be written as y = b0 + b1*x1 + b2*x2 +e. In some situations, there might be an

interaction effect between some predictors, that is for example, increasing the value of a predictor variable x1 may increase the effectiveness of the predictor x2 in explaining the variation in the outcome variable. Note also that, linear regression models can incorporate both continuous and categorical predictor variables. When you build the linear regression model, you need to diagnostic whether linear model is suitable for your data.

2.1.3 Clustering

It is basically a type of unsupervised learning method. An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labelled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples. Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

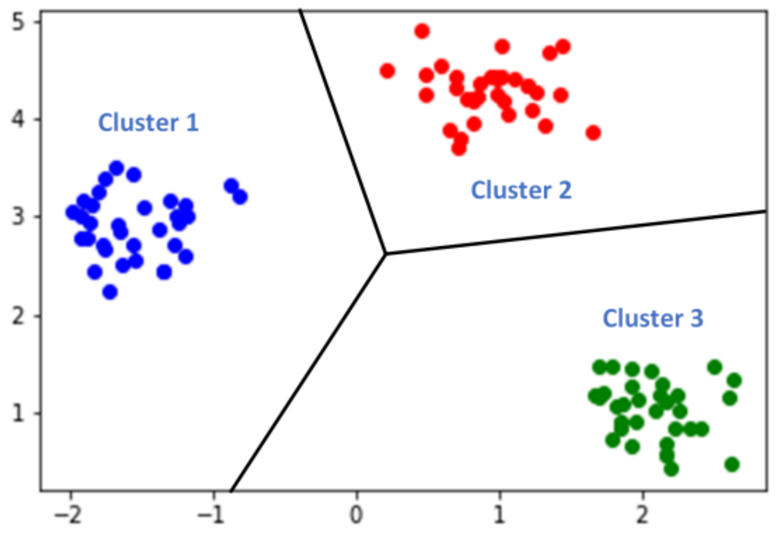


Figure 4: Clustering example

Clustering is very much important as it determines the intrinsic grouping among the unlabeled data present. There are no criteria for a good clustering. It depends on the user, what is the criteria they may use which satisfy their need. For instance, we could be interested in finding representatives for homogeneous groups (data reduction), in finding “natural clusters” and describe their unknown properties (“natural” data types), in finding useful and suitable groupings (“useful” data classes) or in finding unusual data objects (outlier detection). This algorithm must make some assumptions which constitute the similarity of points and each assumption make different and equally valid clusters. For more on clustering methods see [12].

2.2 Python for Machine Learning

Machine learning and artificial intelligence-based projects are obviously the future. We want better personalization, smarter referrals, and improved search functionality. Our applications can see, hear and respond - that's what artificial intelligence (AI) has brought, improving the user experience and creating value in many industries.
Artificial intelligence projects are different from traditional software projects. The differences lie in the pile of technology, the skills needed for a project based on artificial intelligence, and the necessity of in-depth research. To realize your aspirations for artificial intelligence, you should use a stable, flexible programming language that has the tools available. Python offers all of that, which is why we see a lot of Python AI projects today.
Implementing AI and ML algorithms can be tricky and time consuming. It is vital to have a well-structured and proven environment that will allow developers to come up with the best coding solutions.
To reduce development time, developers are turning to numerous Python frameworks and libraries. A software library is pre-written code that developers use to solve common program tasks. Python with its rich technology stack has an extensive set of libraries for artificial intelligence and machine learning. Here are some of them:

- Keras, TensorFlow, and Scikit-learn for machine learning
- NumPy for high-performance scientific computing and data analysis
- SciPy for advanced computing
- Pandas for general-purpose data analysis
- Seaborn for data visualization

Scikit-learn features various classification, regression, and clustering algorithms, including support vector machines, random forests, gradient boosting, k-means, and DBSCAN, and is designed to work with the Python numerical and scientific libraries NumPy and SciPy. With these solutions, you can develop your product faster. Your development team won't have to reinvent the wheel and can use an existing library to implement necessary features. You can see more about Python in Machine learning in the book [11] or on the website [14].

2.3 Algorithms

The algorithm represents a series of steps that we need to follow in order to solve a particular task. It is a set of operations needed to solve a task that are performed in a specific order. In order for the algorithm to solve the problem, it is necessary to define it at the ”input” while at the ”output” there is a solution to the problem. It is important to note that the algorithm must have the correct procedure, which means that each segment of the set problem (at the input) must have a defined solution (at the output). If it is clearly set and there are no errors, the procedure is considered successful. Each algorithm consists of a beginning and an end, while in between there are elements that represent the basic parts that contain a defined path to solving the problem. The algorithm can be described in several ways: natural language, diagrams, pseudocode, programming language. Diagrams are perhaps the most popular way, because in a simple and picturesque way the problem is presented and solved. Depending on the complexity of the problem, three basic algorithm structures are used in the solution process:

- *Linear* algorithmic structure is one where the steps are performed one by one, with the proviso that they are not repeated, but each is executed only once.
- *Branched* algorithmic structure is one in which each step is executed once or not at all. It depends on the conditions that are set, so it can happen that a command is never executed. Several steps are set up to cover all possible conditions, and in this way we try to have an answer to every situation.
- *Cyclic* algorithmic structure represents the possibility of using certain steps several times. This happens when the values to be entered are not known. Cycles represent an opportunity to take a step back and allow other information to be entered.

Big $\mathcal{O}$ notation: Big $\mathcal{O}$ notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity. We give a formal definition: Let f be a real or complex valued function and g a real valued function. Let both functions be defined on some unbounded subset of the real positive numbers, and $g(x)$ be strictly positive for all large enough values of x. We write

$$f(x) = \mathcal{O}(g(x))$$

as $x \to \infty$.

If the absolute value of $f(x)$ is at most a positive constant multiple of $g(x)$ for all sufficiently large value of x. That is, $f(x) = \mathcal{O}(g(x))$ if there exists a positive real number M and a real number x_0 such that

$$|f(x)| \leq Mg(x)$$

for all $x \geq x_0$.

In many contexts, the assumption that we are interested in the growth rate as the variable x goes to infinity is left unstated, and one writes more simply that

$$f(x) = \mathcal{O}(g(x))$$

.

Big Omega notation: Another asymptotic notation is Ω, read "big Omega", defined as

$$f(x) = \Omega g(x)(x \to a)$$

where a is some real number, ∞, or - ∞, where f and g are real functions defined in a neighbourhood of a, and where g is positive in this neighbourhood.

Big Theta notation: Another asymptotic noation is Θ read "Big Theta", defined as:

$$f(n) = \Theta(g(n))$$

f is bounded both above and below by g asymptotically.

P and NP-hardness: We now say few word about hardness of an algorithm. *class P* or just P represents a general class of questions for which some algorithm can provide an answer in polynomial time. We do not always have an answer to some problems or there is no known way to find an answer quickly, but if one is provided with information showing what the answer is, it is possible to verify the answer quickly. The class of questions for which an answer can be verified in polynomial time is called NP, which stands for *non-deterministic polynomial time.* For more reading on these topics, refer to Open Data Structures An Introduction [10].

2.3.1 Time complexity

Time complexity is used to compute which tells us how much time it takes to execute our algorithm. It depends on the size of the data that the algorithm processes. It additionally helps us in defining the efficiency of the algorithm. Time complexity is an asymptotic function calculated from the magnitude of the input data and takes mathematical symbols of Landau: $\mathcal{O}$, Ω, Θ. Here, each symbol defines different time complexity. The $\mathcal{O}$ notation represents the upper limit of special time and has the worst possible outcome. On the other hand,the notation Ω represents the lower limit for the time required and describes the best possible outcome. The Θ notation limits the time required with an upper and lower limit (combines both Ω and $\mathcal{O}$). We have different types of time complexity and it depends on the expressions in the program. There are different types of time complexity, depending on the time spent by each algorithm till it reaches the end of its execution. Therefore, the type of time complexity depends on the instructions or statements in a program. Few examples are: constant time $\mathcal{O}(n)$=1, linear time $\mathcal{O}(n)$=n, logarithmic time $\mathcal{O}(n) = \log n$. If we add the time complexity of all the lines, we'll get the overall time complexity of the factorial function $T(n) = \mathcal{O}(n)$. The method of calculating the complexity in this way is called the iterative method since an iterative algorithm is parsed line by line and the complexity was added. Other than the iterative method, there are also other methods to calculate the time complexity. Such is the recursive method that calculates the time complexity for recurrent solutions and uses recursive trees or substitutions. Other methods for calculating the time complexity also exit, such as the master's theorem, and for more on them we refer to [3].

2.3.2 Space complexity

When we run an algorithm it requires a certain amount of memory space. That is why we say that the complexity of space represents the amount of memory we need for a program to run successfully. When running the program, it also needs the memory used to store the input data. As we said about temporal complexity, spatial complexity is used to help us evaluate algorithms and solutions. It is presented using the same Landau records we presented earlier in the case of temporal complexity. We also have several different types of spatial complexity, which depend on the memory consumed by each algorithm. If an algorithm operates on the given instance and includes at most a constant number of new variables or space slots, the algorithm has constant space complexity. An example of such an algorithm is the selection sort algorithm. Merge sort, on the other hand, is an algorithm with linear space complexity since it creates many arrays that copy the data from the original array. Observe that more space is needed as the size of the array increases. Similar to time complexity, space complexity is often expressed asymptotically in big $\mathcal{O}$ notation, such as $\mathcal{O}(n)$, $\mathcal{O}(n\log(n))$, $\mathcal{O}(n^\alpha)$, $\mathcal{O}(2^n)$ etc., where n is a characteristic of the input influencing space complexity. See [1] and [13] for more information about machine learning algorithms.

Calculating the Space Complexity: In order to calculate space complexity, we need to know the value of memory used by a different type of datatype variables, which depends on different operating systems, but the calculating method remains the same. In Table we present value of memory different type of datatype variables are using.

Table 1: Value of memory used by different type of datatype variables

Type	**Size**
bool, char, unsigned char, signed char, _int8	1 byte
_int16, short, unsigned short, wchar_t, _wchar_t	2 bytes
float, _int32, int, unsigned int, long, unsigned long	4 bytes
double, _int64, long double, long long	8 bytes

3 Known development results and scientific thoughts so far

Their work is a prediction of the development of scientific research based on scientific publications from the last two centuries. They applied machine learning methods to the Microsoft Academic Graph (MAG) database. The research thesis is based on the fact that scientific articles will be important in the future. They have defined the problem of machine learning so that the model will predict early indicators that suggest which scientific topics will become important in the future. The main contributions of this paper are the definition of the proposed problem, the presentation of the data and the identified topics that could be the next big thing in science (see [9]). They use the titles of scientific articles in their research, they determine when a scientific topic first appears, how often it appears and when it ceases to be used. To do this, they used today the largest open database of scientific articles known as Microsoft Academic Groups (MAG). The size of the database is 104 gigabytes and it includes 125 million scientific articles from 1800 to 2015. Out of 125 million articles, they singled out 2.5 million who could be potential candidates. They took into account data that have a range of 5 years or longer. That leaves them with 1 million such topics. The basic task is to use data from over 200 years of scientific discoveries from publications and to single out early signs for topics that could become popular. Using machine learning, they trained a statistical model to classify scientific topics into two categories: those that became important and those that did not. I will use the Perceptron MaxMarign algorithm for machine learning. Their statistical model MaxMarign Perceptron Algorithm gave the following test results: Precision: 74.3, Recall: 71.7, F1: 73.0.
This means that their model correctly identified 71.1% of the topics that became popular and 74.3% of the topics that it predicted would become popular (see for more detailed results [4]). They also want to point out that it would be interesting to repeat the same research with updated data. As well as to compare this algorithm and its results with another algorithm and its results. Another line of work in the future could be the application of the same algorithm to the Open Academic Graph (OAG) database or AMiner.

4 Development of a prediction model

4.1 Initial approach

In our research, we used the titles of scientific articles to determine when a scientific topic first appears, how often occurs over time and when discontinued. There are many databases of scientific articles in the world, but only some are open and available for research. Today, the largest open database of scientific articles is known as the "Microsoft Academic Graph." which was published for research use in 2016. The size of the database is 104 gigabytes, and includes a reference to 125 million scientific members from 1800 to 2015 from all fields of science. Each scientific article in the database is described by its titles, authors, their institution, the journal or conference at which it was published and year of issue. In mid-2017, OAG v1 was published which contains 166,192,182 papers from MAG and 154,771,162 papers from AMiner and generated 64,639,608 connecting (matching) relationships between the two charts. This time, in OAG v2, it is available with information about the author, place, and new publication that matches the match.

OAG V1 Overview
This data set is created by linking two large academic charts: Microsoft Academic Graph (MAG) and AMiner, and is used for research purposes only. This version includes 166,192,182 papers from MAG and 154,771,162 papers from AMiner. We generated 64,639,608 connecting (aligned) relationships between the two graphs. More connecting results will be published in the future, such as the authors. It can be used as a single large academic graph to study the citation network, paper content, and others, and it can also be used to study the integration of multiple academic graphs. The entire data set consists of three parts, which are described in the table below:

Data Set	**#Paper**	**#File**	**Total Data Size**
Linking relations (matching)	64,639,608	1	1.6GB 2017-06-22
MAG papers	166,192, 182	9	104GB 2017-06-09
AMiner papers	154,771,162	3	39GB 2017-03-22

For a set of MAG paper and AMiner papers, each article is written as a JSON object. Its data scheme is:

Field Name	Field Type	Description	Example
id	string	MAG or AMiner ID	53e9ab9eb7602d970354a97e
title	string	paper title	Data mining: concepts and techniques
authors.name	string	author name	Jiawei Han
author.org	string	author affiliation	department of computer science university of illinois
venue	string	paper venue	Inteligencia Artificial
year	int	published year	2000
keywords	list of string	keywords	[“data mining”, “social network”, “relational data”]
fos	list of string	fields of study	[“relational database”, “data model”]
n_citation	int	number of citation	29790
references	list of string	citing papers’ ID	[“53e99ef4b7602d97027c2346”]
page_stat	string	start of page	11
page_end	string	end of page	18
doc_type	string	paper type:book	journal, book title..
lang	string	detected language	en
publisher	string	publisher	Elsevier
volume	string	volume	10
issue	string	issue	29
issn	string	issn	0020-7136
isbn	string	isbn	1-55860-489-8
doi	string	doi	10.4114/ia.v10i29.873
pdf	string	pdf URL	//static.aminer.org/upload/pdf/53e.pdf
url	list	external links	[“http://dx.doi.org/10.4114/ia.v10i29.873”]
abstract	string	abstract	Our ability to generate...

For example:

```
{
"id": "53e9ab9eb7602d970354a97e",
"title": "Data mining: concepts and techniques",
"authors": [
{
"name": "jian pei",
"org": "department of computer science university of
   illinois at urbana champaign"
}
],
"year": 2000,
"keywords": [
"data mining",
"social network",
"relational data"
],
"fos": [
"relational database",
"data model",
"social network"
],
"n_citation": 29790,
"references": [
"53e99ef4b7602d97027c2346"
],
"doc_type": "book",
"lang": "en",
"publisher": "Elsevier",
"isbn": "1-55860-489-8",
"doi": "10.4114/ia.v10i29.873",
"pdf": "//static.aminer.org/upload/pdf/1254/370/239/53e9ab9
   eb7602d970354a97e.pdf",
"url": [
"http://dx.doi.org/10.4114/ia.v10i29.873",
"http://polar.lsi.uned.es/revista/index.php/ia/article/view
   /479"
],
"abstract": "Our ability to generate and collect data has
   been increasing rapidly. Not only are all of our
   business, scientific, and government transactions now
   computerized, but the widespread use of digital cameras,
    publication tools, and bar codes also generate data."
}
```

In the example, we see that keywords and foss contain the names of the areas covered by the article. We want to create a model in our work that will refer only to certain areas of computer science. The first step is to filter the data and single out articles that mention the following scientific fields: “computer science”, “machine learning”, “data mining”, “data analysis”, “data science”. After that, we will turn them into a CSV file for easier work with them in the rest of the research and model creation. And then creating and training models.

4.2 Main Algorithm

Our first goal is to filter the data and single out articles that mention the following scientific fields: “computer science”, “machine learning”, “data mining”, “data analysis”, “data science”. We did this with the following algorithm.

```
import json
keywords = ["computer science", "machine learning",
"data mining", "data analysis", "data science"]

foldernr = 6
od = 120
do = 139

folder = "mag_pepers_{}".format(foldernr)
for i in range(od, do+1):
    filename = "~/mag{}/mag_papers_{}/mag_papers_{}.txt".format(
    foldernr, foldernr, i)
    print(filename)
    f = open(filename, 'r')
    g = open("~/imena_papira.txt", "w")
    for line in f.readlines():
        article = json.loads(line)
        if article['n_citations'] >= 50:
            for keyword in keywords:
                if keyword in article['keywords']:
                    g.write(line)
                    break
```

Once we have finished filtering the data from we want to create a CSV file. To do this because we want to make it easier to manipulate the data we have, but also because it is easier to read and review the data itself.
The algorithm that will be used for this is as follows:

```
import csv
field_names = ['id', 'title', 'authors', 'venue.id',
               'venue.raw', 'year', 'keywords', 'n_citacion',
               'page_start', 'page_end', 'doc_type', 'lang',
               'publisher', 'volume', 'issue', 'issn', 'isbn',
               'doi', 'pdf', 'url', 'fos','references', 'abstract', '
                  venue', 'n_citation']


data = [#MAG DATA]

for dictionary in data:
    dd = dict()
    dd['author.name'] = dictionary['authors']
# with open("mag8_prebrano", "r") as mag:
with open('MAG8.1.csv', 'w') as csvfile:
    writer = csv.DictWriter(csvfile, fieldnames=field_names)
    writer.writeheader()
    writer.writerows(data)
```

Notice that all the data is in textual format. That implies that the data cannot be compared. However, python has a fix for that. Namely, the go around about that is to convert the words in the data to vectors, or rather vector values. The following algorithm does that for us.

```
from gensim.models import Word2Vec, KeyedVectors
import pandas as pd
import nltk
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

df = pd.read_csv('MAG81.csv')
df.head(10)

newsFos = df["fos"].values

for elm in df["fos"].values:
    print(" ".join(elm[2:-2].split("', '")))

newsFos = np.array([" ".join(elm[2:-2].split("', '")) for elm in df["fos"
    ].values])
newsFos

newsVec = [nltk.word_tokenize(fos) for fos in newsFos]
newsVec

model = Word2Vec(newsVec, vector_size=32, min_count=1)

model.wv.most_similar('computer')

vec = model.wv['Computer']+ model.wv['Science'] + model.wv['Machine'] +
                model.wv['Learning'] + model.wv['Data'] + ['Mining'] + ['
                    Analysis']
model.wv.most_similar([vec])
```

4.3 Analysis and results

4.3.1 Algorithm analysis

Algorithm 1: Data filtering algorithm
Line 1 of code we define keywords, ie the areas we want to highlight. In our case they are: "computer science", "machine learning", "data mining", "data analysis", "data science".
Line 4-6, select the folder and the number of articles. In our case, these are folder number 6 and all the articles that are in it, ie from mag_papers_120 to mag_papers_139.
Line 8 we call the data folder we defined earlier.
Line 9-11 we go through the whole file (mag_papers_120 to mag_papers_139), then save the whole path to filename.
Line 13 of the code we open that file. While in line 14 we open the file in which we will write.
Line15-19 we go through all the lines mag_papers_C to check, then we go through all the keywords from the list and then we look if it is in one of the articles we write it in another file that we keep.We also look at whether the article has more or less than 50 readings. In this way we make two clusters, ie popular and those less popular.
The time complexity of the algorithms is as follows:

$$\sum_{i=1}^{m}(1+1+1+\sum_{j=1}^{n_i}(\sum_{k=1}^{5}1)) =$$
$$\sum_{i=1}^{m}(3+\sum_{j=1}^{n_i}5)=\sum_{m}^{i=1}(3+5n_i) =$$
$$\sum_{i=1}^{m}3+5\sum_{i=1}^{m}n_i=\mathcal{O}(n)$$

Where we denote by m and n_i the following:

- m-number of MAG
- n_i - number of articles in MAG

Since the highest order of n in the equation 4n 12 is n, so the space complexity is $\mathcal{O}(n)$ or linear.
Algorithm 2: JSON data to CSV We first defined field_name - this is the first row that divides all the data contained in the article into columns (id ',' title ',' authors ',' venue.id ',' venue.raw ',' year ',' keywords ',' n_citacion ',' page_start ',' page_end ',' doc_type ',' lang ',' publisher ',' volume ',' issue ',' issn ',' isbn ',' doi ',' pdf ', 'url', 'fos', 'references', 'abstract', 'venue', 'n_citation').
Line 9 we see an example of JSON data (Due to size we took only one example)
Line 11-13 we separate the authors and define the author.name for each article (because we need an author.name and we have an authors-list)

Line 15-18 open the csv file in which we will store the data. At the beginning we write the first row (id, title, authors, venue.id, ...), after that we write the other data and sort the columns properly. This algorithm has time complexity $\mathcal{O}(n)$(n-number of all articles) because it goes iteratively through each file, reads it line by line, and writes each to a CSV file, so the space complexity is $\mathcal{O}(n)$ or linear.

Algorithm 3: Word2Vec

In this algorithm, we use a language modeling technique used to transform words into vectors of real numbers. Represents words or expressions in a vector space with several dimensions. Word2Vec consists of a model for generating word insertion. This model is a shallow two-layer neural network that has one input layer, one hidden layer, and one output layer.

n-number of all words read This algorithm has time complexity $\mathcal{O}(n)$ because it goes iteratively through selected columns, reads and assigns vector values. The space complexity of this algorithm is $\mathcal{O}(n)$

Line 1-6 import required libraries.

Line 8 read CSV data file.

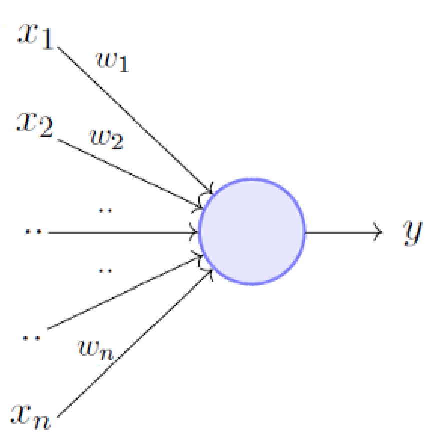
Line 9 print the first 10 for data verification.

Line 11-17 review all words and print them.

Line 19-28 model creation, token assignment, and model printing.

Shih-Yuan Yu et. al in [16] provide more details on the word2vec method and their as well as examples of their application.

Algorithm 4: Perceptron



$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i \geq \theta$$

$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i < \theta$$

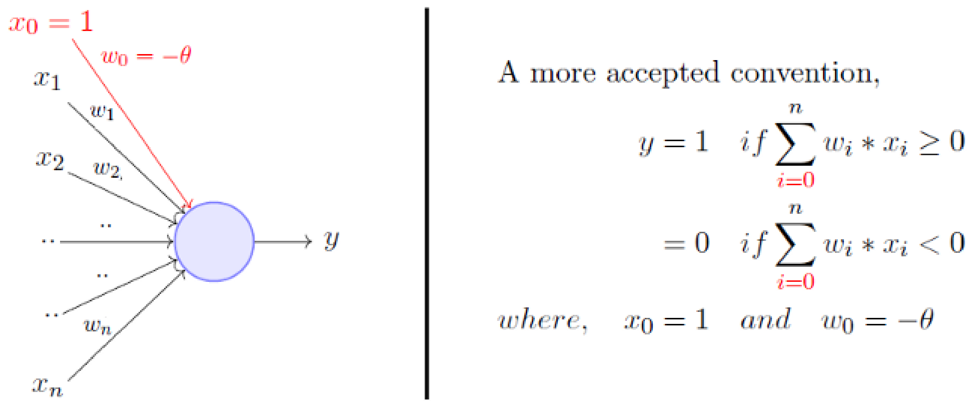
Rewriting the above,

$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i - \theta \geq 0$$

$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i - \theta < 0$$

Figure 5: The Perceptron model looks like a model of neurons

The perceptron model is a computational model of neurons. It takes an input, aggregates it, and returns 1 only if the sum is greater than some threshold that otherwise returns 0. By overwriting the threshold as shown above and making it a constant input with variable weight, we would end with something like this:



A more accepted convention,

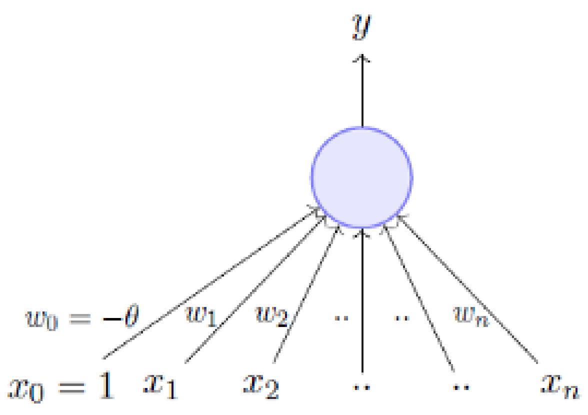
$$y = 1 \quad if \sum_{i=0}^{n} w_i * x_i \geq 0$$

$$= 0 \quad if \sum_{i=0}^{n} w_i * x_i < 0$$

$$where, \quad x_0 = 1 \quad and \quad w_0 = -\theta$$

Figure 6: Perceptron constant input with variable weight

One perceptron can only be used to implement linearly separable functions. Both real and logical inputs are required and they are accompanied by a set of weights, along with a bias.
Defining the problem:



- $x1$ = authors.name
- $x2$ = title
- $x3$ = publisher
- $x4$ = n_citacion(scaled to 0 to 1)
- xn = author.org

Figure 7: Perceptron model for article evaluation

We will use a perceptron for as we have said an estimate of which scientific topics will be popular based on the data above in the mentioned inputs. See more about the Perceptron prediction algorithm in [5]. The data have positive and negative examples.
Perceptron Learning Algorithm:
Our goal is to find a v vector that can classify positive and negative investments in our data. More informations about Perceptron Learning can be found in the book [8].

Algorithm 1: Perceptron

```
P ← inputs with label 1;
N ← inputs with label 0;
Initialize w randomly;
while !convergence do
    Pick random x ∈ P ∪N;
    if x ∈ P and w.x ¡ 0 then
        w = w + x;
    if x ∈ N and w.x ≥ 0 then
        w = w - x;
```

We will run w some random vector. Then we repeat all the examples in the data, P $\in N$ and the positive and negative examples. And if x belongs to N, the dot product must be less than 0. So if you look at the if conditions in the while loop:

```
while !convergence do
    Pick random x ∈ P ∪N;
    if x ∈ P and w.x ¡ 0 then
        w = w + x;
    if x ∈ N and w.x ≥ 0 then
        w = w - x;
```

Case 1: When **x** belongs to P and its dot product $\mathbf{w.x} < 0$
Case 2: When **x** belongs to N and its dot product $\mathbf{w.x} \geq 0$
Only for these cases do we update the randomly initialized **w**. Otherwise, we do not touch w at all because case 1 and case 2 violate only the perceptron rule. The angle between **w** and **x** should be less than 90 because the cosine of the angle is proportional to the point product. Thus, whatever vector **w** be, as long as it makes an angle less than 90 degrees with positive examples of data vectors ($x \in P$) and an angle greater than 90 degrees with negative examples of data vectors ($x \in N$) In [2] you can see more about Perceptron and vectors. In the following view we can see how it should look in ideal shape:

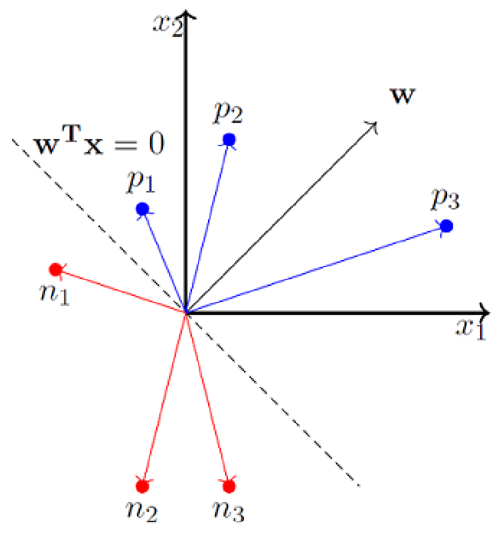


Figure 8: Example ideal shape

More details about the Perceptron algorithm can be found in the book [6].

4.3.2 Result

We split the topics into a training (70%) and test set (30%), where the training set is used to train the model and the testing set is used to test the model. The Perceptron algorithm gave the following test results: Accuracy: 67.1% Recall: 64.3% F1: 65.7% for tilt difference 1. This means that the model correctly recognizes us 66.2% of the topics that have become popular and 67.1% of topics are predicted to become popular and have become polar. The increase in inclination caused a decrease in accuracy. This is most likely a consequence of the difficulty of the collision problem as the number of positive workouts.

Table 2: Table of results

Slope Diff	**Precision**	**Recall**	**F1**	**Accuracy**
1	67.143	64.379	65.744	52.541
2	50.859	49.358	46.091	59.477
3	37.817	44.286	43.223	52.409

The following figure shows the performance of the model for gradually stricter criteria. We can see how the results are different and how the slope increases the performance decreases compared to the performance on the training set. Taking into account the

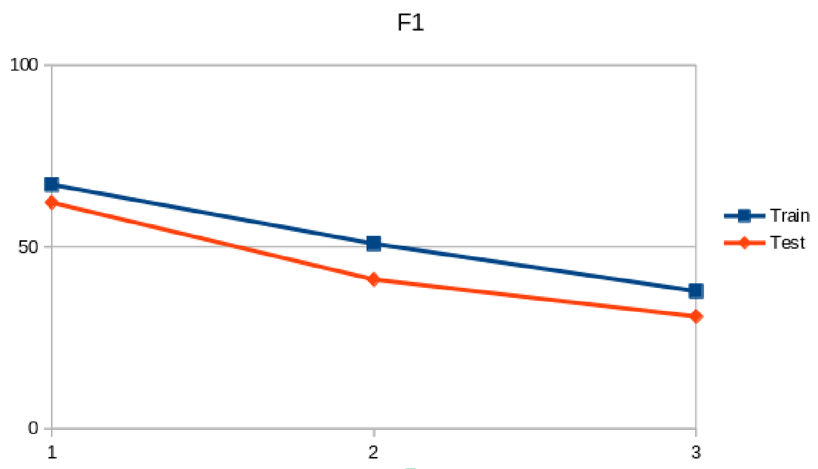


Figure 9: Model performance for test and train data

results of our model we can notice that scientific topics get more attention depending on the university, the author, the journal that publishes it.

5 Conclusion

In this final paper, we studied the method of creating a model with the help of Machine Learning to predict the popularity of scientific topics in the field of "computing", "machine learning", "data mining", "data analysis", "data science". We used 166,192,182 articles from the Microsoft Academic Graph database for analysis. We presented the entire procedure from data preparation to the creation of the final prediction model. We used Python for all our algorithms because of the pre-installed libraries.

In order to be able to process our data and perform the experiment, we implemented an algorithm that separates the articles with the areas we want, then we let the data through an algorithm that converts our data into a CSV file. We then separated articles that had more than 50 citations and those that had less than 50 citations. After we prepared the data for them we created an algorithm that assigns them vector values. And finally, with the help of the Perceptron algorithm, we created a model that was tested on a 70/30 train/test division. The results show satisfactory performance F1 achieves 66%. The model predicts 64.3% of scientific topics that have become important in the history of science.

In future research, it would be interesting to compare different algorithms and methods on the same database and then compare the results.

6 Povzetek naloge v slovenskem jeziku

Z naraščajočo hitrostjo znanstvenega razvoja postaja vse težje spremljati trenutne znanstvene raziskovalne teme, kaj šele napovedati obetavne usmeritve za prihodnje raziskave. Vse večja količina in boljša kakovost digitaliziranih znanstvenih publikacij dandanes omogoča modeliranje razvoja znanstvenih publikacij skozi čas z vse večjo natančnostjo in učinkovitostjo. V naši raziskavi poskušamo ugotoviti kako deluje preprost algoritem perceptrona (umetne nevronske mreže) na veliki količini podatkov o znanstvenih publikacijah.
Naša raziskovalna hipoteza je, da so znanstvene teme, ki bodo pomembne v prihodnosti, že prisotne v današnjih znanstvenih člankih. Za njihovo identifikacijo smo v veliki meri uporabili metode strojnega učenja, zbirko publikacij, in sicer Microsoft Academic Graph. Opredelili smo problem strojnega učenja, tako da model napoveduje zgodnje kazalnike, ki nakazujejo, katere znanstvene teme v današnji literaturi bodo najverjetneje postale pomembne v prihodnosti.
V svoji raziskavi smo se naslanjali predvsem na naslove znanstvenih člankov, iz katerih smo skušali ugotoviti, kdaj se neka znanstvena tema prvič pojavi, kako pogosto se kasneje pojavlja skozi čas in kdaj se neha uporabljati. Obstaja veliko zbirk znanstvenih člankov, vendar so "odprte" in prosto-dostopne le nekatere, ki so tako na voljo širšemu krogu raziskovalcev. Danes največja prosto-dostopna baza podatkov o znanstvenih člankih je poznana kot "Microsoftov akademski graf" (MAG), ki je bila za raziskovalno uporabo objavljena leta 2016. Velikost te baze podatkov je 104 gigabajte in vključuje sklicevanja na 125 milijonov znanstvenih člankov od leta 1800 do 2015 z vseh področij znanosti. Vsak znanstveni članek v bazi podatkov opisuje njegov naslov, avtorji, njihove institucije, revijo ali konferenco, kjer je bil objavljen in leto izdaje.
V zaključni nalogi smo uporabili bazo znanstvenih publikacij "Odprti akademski graf" ali krajše OAG, ki vključuje nekaj čez 160 milijonov znanstvenih publikacij iz baze MAG ter dodatno še nekaj čez 150 milijonov znanstvenih publikacij iz baze AMiner, ki je še ena izmed prosto-dostopnih baz znanstvenih publikacij. OAG oz. OAG v2 (2. različica) vsebuje tako prek 310 milijonov znanstvenih publikacij in je posodobljena tako, da vsebuje tudi znanstvene publikacije po letu 2015.

Ker nas je zanimal razvoj znanstvenih tem le na področju računalništva in informatike ter podpodročjih, ki so povezana z podatkovno znanostjo, strojnim učenjem in podatkovnim rudarjenjem, smo iz celotne baze prek 310 milijonov prispevkov izluščili ustrezen vzorec le-teh. Vzorčenje smo naredili tako, da smo obdržali le znanstvene članke, ki so v naslovu ali povzetku (kjer je le-ta bil na voljo) vsebovali eno od sledečih besed ali besednih zvez: "computer science", "machine learning", "data mining", "data analysis" in "data science".
Tako pridobljen vzorec je vseboval 8932742 znanstvenih prispevkov. Iz prvotne oblike (OAG v2) smo podatke o znanstvenih prispevkih najprej spremenili v CSV obliko (besede ločene z vejico) ter nato s pomočjo Word2Vec postopka iz vsak znanstveni prispevek pretvorili v vektor pojavitve besed, ki jih je le-ta vseboval. Končen seznam vektorjev pojavitve besed smo potem uporabili kot vhod v algoritem strojnega učenja. Kot algoritem strojnega učenja smo se odločili za "preprosti" perceptron - umetna nevronska mreža s povezavami le naprej (feed-forward artificial neural network). Po pregledu literature ter nekaj uvodnega testiranja, smo se odločili za sledečo topologijo perceptrona: 2 skritih nivojev, 5 nevronov v vsakem nivoju. Napovedni model smo naučili na "prvih" 70% podatkov in ga preizkusili na "zadnjih" 30% podatkov. Rezultati preizkusa so prikazani v spodnji tabeli.

Table 3: Tabela rezultatov

Slope Diff	**Precision**	**Recall**	**F1**	**Accuracy**
1	67.143	64.379	65.744	52.541
2	50.859	49.358	46.091	59.477
3	37.817	44.286	43.223	52.409

Rezultati kažejo 66% F-mero točnosti napovedi. Model je bil zmožen pravilno napovedati 64,3% znanstvenih tem, ki so kasneje postale pomembne v znanosti. V nadaljnjem delu bi bilo zanimivo našo raziskavo razširiti z uporabo še drugih algoritmov strojnega učenja ter napraviti primerjavo napovednih točnosti. Prav tako bi bilo zanimivo identificirati katere so tiste "vroče" znanstvene teme, ki bodo postale pomembne v bližnji prihodnosti.

7 Bibliography

[1] Taiwo Oladipupo Ayodele. Types of machine learning algorithms. *New advances in machine learning*, 3:19–48, 2010. *(Cited on page 12.)*

[2] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 conference on empirical methods in natural language processing (EMNLP 2002)*, pages 1–8, 2002. *(Cited on page 23.)*

[3] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009. *(Cited on page 12.)*

[4] Adrian Mladenic Grobelnik, Dunja Mladenic, and Marko Grobelnik. The next big thing in science. In *Proceedings of the 22nd International Multiconference Information Society–IS*, pages 9–12, 2019. *(Cited on pages 2 and 14.)*

[5] Corina Gurău, Dushyant Rao, Chi Hay Tong, and Ingmar Posner. Learn from experience: Probabilistic prediction of perception performance to avoid failure. *The International Journal of Robotics Research*, 37(9):981–995, 2018. *(Cited on page 22.)*

[6] Corina Gurău, Dushyant Rao, Chi Hay Tong, and Ingmar Posner. Learn from experience: Probabilistic prediction of perception performance to avoid failure. *The International Journal of Robotics Research*, 37(9):981–995, 2018. *(Cited on page 24.)*

[7] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015. *(Cited on page 5.)*

[8] Shivaram Kalyanakrishnan. The perceptron learning algorithm and its convergence. 2017. *(Cited on page 22.)*

[9] Yaoyong Li, Hugo Zaragoza, Ralf Herbrich, John Shawe-Taylor, and Jaz S Kandola. The perceptron algorithm with uneven margins. In *Proceedings of the nineteenth international conference on machine learning*, pages 379–386, 2002. *(Cited on page 14.)*

[10] Pat Morin. *Open Data Structures: An Introduction*, volume 2. Athabasca University Press, 2013. *(Cited on page 11.)*

[11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011. *(Cited on page 9.)*

[12] Kristina P Sinaga and Miin-Shen Yang. Unsupervised k-means clustering algorithm. *IEEE Access*, 8:80716–80727, 2020. *(Cited on page 9.)*

[13] Space complexity of algorithms. `https://www.studytonight.com/data-structures/space-complexity-of-algorithms`, 2020. *(Cited on page 12.)*

[14] Python - time complexity. `https://wiki.python.org/moin/TimeComplexity`. Accessed: 2020-08-07. *(Cited on page 9.)*

[15] Sholom M Weiss and Nitin Indurkhya. Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research*, 3:383–403, 1995. *(Cited on page 6.)*

[16] Shih-Yuan Yu, Sujit Rokka Chhetri, Arquimedes Canedo, Palash Goyal, and Mohammad Abdullah Al Faruque. Pykg2vec: A python library for knowledge graph embedding. *J. Mach. Learn. Res.*, 22:16–1, 2021. *(Cited on page 21.)*