**CHATBOT USING DEEP LEARNING**

**ABSTRACT:**

Dialogue Generation or Intelligent Conversational Agent development using Artificial Intelligence or Machine Learning technique is an interesting problem in the field of Natural Language Processing. In many research and development projects, they are using Artificial Intelligence, Machine Learning algorithms and Natural Language Processing techniques for developing conversation/dialogue agent. Their research and development is still under progress and under experimentation.

In the past, methods for constructing chatbot architectures have relied on hand-written rules and templates or simple statistical methods. With the rise of deep learning these models were quickly replaced by end-to-end trainable neural networks around 2015. More specifically, the recurrent encoder-decoder model dominates the task of conversational modelling.

**INTRODUCTION:**

A conversational agent (chatbot) is a piece of software that is able to communicate with humans using natural language. Modelling conversation is an important task in natural language processing and artificial intelligence (AI). Indeed, ever since the birth of AI, creating a good chatbot remains one of the field’s hardest challenges. While chatbots can be used for various tasks, in general they have to understand users’ utterances and provide responses that are relevant to the problem at hand.

Chabot’s are the systems which are designed to simulate conversation with human. One of the main important functions of Artificial Intelligence is to develop the system to find solutions to the problems. Chat bots are the next step in the internet evolution. In the early 21st century, with the expansion of globalization internet has become the most convenient way to share information. It provides the platform that allows users to participate, offer feedback, share ideas and receive information. Moreover, it is important to manage these resources wisely, relevant communications, timely response to the questions on the problems like data structure questions and question-answering (QA) from conversation.

Problem Statement:

Objectives:

To create a machine that will interact with client and give them response with their quires, it is important to manage these resources wisely, relevant communications, timely response to the questions on the problems like data structure questions and question-answering (QA) from conversation. The learning service is not an algorithmic program rather a trained model using JSON. With the implementation of problem solving chatbot, it will understand how to organize and retrieve data based on user’s data structure choice. There are short term and long-term memory storage with different priority based on variety of situation. The system can take inputs in written or voice format and respond the question from a knowledge base. In most cases a chat bot does not have problem solving capabilities. Our system can solve data structure problems using Deep learning mechanism. Deep neural network (DNN).

Results:

Methods:

Limitaons

Although, there are many chatbots currently available, majority of them are limited in

functionality, domain function, context and coherence. They often fail in long conversa-

tions and have reduced relevancy in dialogue generation. Most of these chatbots are devel-

oped for restricted domain. Majority of them are using simple rule based techniques. They

perform well in question answering sessions and in verystructured conversational modes.

But, fail to emulate real human conversation and lacks lexibility in functioning. Some of

the chatbots using machine learning algorithms often adhere to simple algorithms. They

lack in complexity and sophistication needed to produce good result speciically in open

domain conversation.

Some chat engines are available in market which are often used by businesses for devel-

oping automated customer support. They are also black box and business clients have

limited knowledge of their internal architectures. Hence, they produce results that can

become unreliable and fail to ill the need of customers. Following is an example of failed

chatbot replies.

Limitaons

Although, there are many chatbots currently available, majority of them are limited in

functionality, domain function, context and coherence. They often fail in long conversa-

tions and have reduced relevancy in dialogue generation. Most of these chatbots are devel-

oped for restricted domain. Majority of them are using simple rule based techniques. They

perform well in question answering sessions and in verystructured conversational modes.

But, fail to emulate real human conversation and lacks lexibility in functioning. Some of

the chatbots using machine learning algorithms often adhere to simple algorithms. They

lack in complexity and sophistication needed to produce good result speciically in open

domain conversation.

Some chat engines are available in market which are often used by businesses for devel-

oping automated customer support. They are also black box and business clients have

limited knowledge of their internal architectures. Hence, they produce results that can

become unreliable and fail to ill the need of customers. Following is an example of failed

chatbot replies.

Limitaons

Although, there are many chatbots currently available, majority of them are limited in

functionality, domain function, context and coherence. They often fail in long conversa-

tions and have reduced relevancy in dialogue generation. Most of these chatbots are devel-

oped for restricted domain. Majority of them are using simple rule based techniques. They

perform well in question answering sessions and in verystructured conversational modes.

But, fail to emulate real human conversation and lacks lexibility in functioning. Some of

the chatbots using machine learning algorithms often adhere to simple algorithms. They

lack in complexity and sophistication needed to produce good result speciically in open

domain conversation.

Some chat engines are available in market which are often used by businesses for devel-

oping automated customer support. They are also black box and business clients have

limited knowledge of their internal architectures. Hence, they produce results that can

become unreliable and fail to ill the need of customers. Following is an example of failed

chatbot replies.

**Limitations:**

Although, there are many chatbots currently available, majority of them are limited in functionality, domain function, context and coherence. They often fail in long conversations and have reduced relevancy in dialogue generation. Most of these chatbots are developed for restricted domain. Majority of them are using simple rule based techniques. They perform well in question answering sessions and in very structured conversational modes. But, fail to emulate real human conversation and lacks flexibility in functioning. Some of the chatbots using machine learning algorithms often adhere to simple algorithms. They lack in complexity and sophistication needed to produce good result specifically in open domain conversation. Some chat engines are available in markets which are often used by businesses for developing automated customer support. They are also black box and business clients have limited knowledge of their internal architectures. Hence, they produce results that can become unreliable and fail to bill the need of customers. Following is an example of failed chatbot replies.

Intents.json – The data file which has predefined patterns and responses.

train\_chatbot.py – In this Python file, we wrote a script to build the model and train our chatbot.

Words.pkl – This is a pickle file in which we store the words Python object that contains a list of our vocabulary.

Classes.pkl – The classes pickle file contains the list of categories.

Chatbot\_model.h5 – This is the trained model that contains information about the model and has weights of the neurons.

Chatgui.py – This is the Python script in which we implemented GUI for our chatbot. Users can easily interact with the bot.

**Algorithms & Literature survey**

Linear Regression

What is Regression Analysis?

Regression analysis is a form of predictive modelling technique which investigates the relationship between a dependent (target) and independent variable (s) (predictor). This technique is used for forecasting, time series modelling and finding the causal effect relationship between the variables. For example, relationship between rash driving and number of road accidents by a driver is best studied through regression.

Regression analysis is an important tool for modelling and analyzing data. Here, we fit a curve / line to the data points, in such a manner that the differences between the distance of data points from the curve or line is minimized. The topic will be explained in detail in coming sections.

Why do we use Regression Analysis?

As mentioned above, Regression analysis estimates the relationship between two or more variables. Let’s understand this with an easy example:

Let’s say, you want to estimate growth in sales of a company based on current economic conditions. You have the recent company data which indicates that the growth in sales is around two and a half times the growth in the economy. Using this insight, we can predict future sales of the company based on current & past information.

There are multiple benefits of using Regression analysis. They are as follows:

It indicates the significant relationships between dependent variable and independent variable.

It indicates the strength of impact of multiple independent variables on dependent variable.

Regression analysis also allows us to compare the effects of variables measured on different scales, such as the effect of price changes and the number of promotional activities. These benefits help Market Researchers / Data Analysts / Data Scientists to eliminate and evaluate the best set of variables to be used for building predictive models.

Linear Regression

It is one of the most widely known modeling technique. Linear regression is usually among the first few topics which people pick while learning predictive modeling. In this technique, the dependent variable is continuous, independent variable(s) can be continuous or discrete, and nature of regression line is linear.

Linear Regression establishes a relationship between dependent variable (Y) and one or more independent variables (X) using a best fit straight line (also known as regression line).

It is represented by an equation Y=a+b\*X + e, where a is intercept, b is slope of the line and e is error term. This equation can be used to predict the value of target variable based on given predictor variable(s).

Linear regression is perhaps one of the most well known and well understood algorithms in statistics and machine learning.

In this post you will discover the linear regression algorithm, how it works and how you can best use it in on your machine learning projects. In this post you will learn:

Why linear regression belongs to both statistics and machine learning.

The many names by which linear regression is known.

The representation and learning algorithms used to create a linear regression model.

How to best prepare your data when modeling using linear regression.

You do not need to know any statistics or linear algebra to understand linear regression. This is a gentle high-level introduction to the technique to give you enough background to be able to use it effectively on your own problems.

Machine learning, more specifically the field of predictive modeling is primarily concerned with minimizing the error of a model or making the most accurate predictions possible, at the expense of explainability. In applied machine learning we will borrow, reuse and steal algorithms from many different fields, including statistics and use them towards these ends.

As such, linear regression was developed in the field of statistics and is studied as a model for understanding the relationship between input and output numerical variables, but has been borrowed by machine learning. It is both a statistical algorithm and a machine learning algorithm.

Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).

When there is a single input variable (x), the method is referred to as simple linear regression. When there are multiple input variables, literature from statistics often refers to the method as multiple linear regression.

Different techniques can be used to prepare or train the linear regression equation from data, the most common of which is called Ordinary Least Squares. It is common to therefore refer to a model prepared this way as Ordinary Least Squares Linear Regression or just Least Squares Regression.

Now that we know some names used to describe linear regression, let’s take a closer look at the representation used.

Linear Regression Model Representation

Linear regression is an attractive model because the representation is so simple.

The representation is a linear equation that combines a specific set of input values (x) the solution to which is the predicted output for that set of input values (y). As such, both the input values (x) and the output value are numeric.

The linear equation assigns one scale factor to each input value or column, called a coefficient and represented by the capital Greek letter Beta (B). One additional coefficient is also added, giving the line an additional degree of freedom (e.g. moving up and down on a two-dimensional plot) and is often called the intercept or the bias coefficient.

For example, in a simple regression problem (a single x and a single y), the form of the model would be:

y = B0 + B1\*x

In higher dimensions when we have more than one input (x), the line is called a plane or a hyper-plane. The representation therefore is the form of the equation and the specific values used for the coefficients (e.g. B0 and B1 in the above example).

It is common to talk about the complexity of a regression model like linear regression. This refers to the number of coefficients used in the model.

When a coefficient becomes zero, it effectively removes the influence of the input variable on the model and therefore from the prediction made from the model (0 \* x = 0). This becomes relevant if you look at regularization methods that change the learning algorithm to reduce the complexity of regression models by putting pressure on the absolute size of the coefficients, driving some to zero.

**Linear Regression Learning the Model**

Learning a linear regression model means estimating the values of the coefficients used in the representation with the data that we have available.

In this section we will take a brief look at four techniques to prepare a linear regression model. This is not enough information to implement them from scratch, but enough to get a flavor of the computation and trade-offs involved.

There are many more techniques because the model is so well studied. Take note of Ordinary Least Squares because it is the most common method used in general. Also take note of Gradient Descent as it is the most common technique taught in machine learning classes.

1. Simple Linear Regression

With simple linear regression when we have a single input, we can use statistics to estimate the coefficients.

This requires that you calculate statistical properties from the data such as means, standard deviations, correlations and covariance. All of the data must be available to traverse and calculate statistics.

This is fun as an exercise in excel, but not really useful in practice.

2. Ordinary Least Squares

When we have more than one input we can use Ordinary Least Squares to estimate the values of the coefficients.

The Ordinary Least Squares procedure seeks to minimize the sum of the squared residuals. This means that given a regression line through the data we calculate the distance from each data point to the regression line, square it, and sum all of the squared errors together. This is the quantity that ordinary least squares seeks to minimize.

This approach treats the data as a matrix and uses linear algebra operations to estimate the optimal values for the coefficients. It means that all of the data must be available and you must have enough memory to fit the data and perform matrix operations.

It is unusual to implement the Ordinary Least Squares procedure yourself unless as an exercise in linear algebra. It is more likely that you will call a procedure in a linear algebra library. This procedure is very fast to calculate.

3. Gradient Descent

When there are one or more inputs you can use a process of optimizing the values of the coefficients by iteratively minimizing the error of the model on your training data.

This operation is called Gradient Descent and works by starting with random values for each coefficient. The sum of the squared errors are calculated for each pair of input and output values. A learning rate is used as a scale factor and the coefficients are updated in the direction towards minimizing the error. The process is repeated until a minimum sum squared error is achieved or no further improvement is possible.

When using this method, you must select a learning rate (alpha) parameter that determines the size of the improvement step to take on each iteration of the procedure.

Gradient descent is often taught using a linear regression model because it is relatively straightforward to understand. In practice, it is useful when you have a very large dataset either in the number of rows or the number of columns that may not fit into memory.

4. Regularization

There are extensions of the training of the linear model called regularization methods. These seek to both minimize the sum of the squared error of the model on the training data (using ordinary least squares) but also to reduce the complexity of the model (like the number or absolute size of the sum of all coefficients in the model).

Making Predictions with Linear Regression

Given the representation is a linear equation, making predictions is as simple as solving the equation for a specific set of inputs.

Let’s make this concrete with an example. Imagine we are predicting weight (y) from height (x). Our linear regression model representation for this problem would be:

y = B0 + B1 \* x1

or

weight =B0 +B1 \* height

Where B0 is the bias coefficient and B1 is the coefficient for the height column. We use a learning technique to find a good set of coefficient values. Once found, we can plug in different height values to predict the weight.

Preparing Data For Linear Regression

Linear regression is been studied at great length, and there is a lot of literature on how your data must be structured to make best use of the model.

As such, there is a lot of sophistication when talking about these requirements and expectations which can be intimidating. In practice, you can uses these rules more as rules of thumb when using Ordinary Least Squares Regression, the most common implementation of linear regression.

Try different preparations of your data using these heuristics and see what works best for your problem.

**Linear Assumption:** Linear regression assumes that the relationship between your input and output is linear. It does not support anything else. This may be obvious, but it is good to remember when you have a lot of attributes. You may need to transform data to make the relationship linear (e.g. log transform for an exponential relationship).

**Remove Noise:** Linear regression assumes that your input and output variables are not noisy. Consider using data cleaning operations that let you better expose and clarify the signal in your data. This is most important for the output variable and you want to remove outliers in the output variable (y) if possible.

**Remove Collinearity:** Linear regression will over-fit your data when you have highly correlated input variables. Consider calculating pairwise correlations for your input data and removing the most correlated.

**Gaussian Distributions:** Linear regression will make more reliable predictions if your input and output variables have a Gaussian distribution. You may get some benefit using transforms (e.g. log or BoxCox) on you variables to make their distribution more Gaussian looking.

**Rescale Inputs:** Linear regression will often make more reliable predictions if you rescale input variables using standardization or normalization.

**Logistic Regression:**

Logistic Regression

Researchers are often interested in setting up a model to analyze the relationship between predictors (i.e., independent variables) and it's corresponding response (i.e., dependent variable). Linear regression is commonly used when the response variable is continuous. One assumption of linear models is that the residual errors follow a normal distribution. This assumption fails when the response variable is categorical, so an ordinary linear model is not appropriate. This newsletter presents a regression model for response variable that is dichotomous–having two categories. Examples are common: whether a plant lives or dies, whether a survey respondent agrees or disagrees with a statement, or whether an at-risk child graduates or drops out from high school.

In ordinary linear regression, the response variable (Y) is a linear function of the coefficients (B0, B1, etc.) that correspond to the predictor variables (X1, X2, etc.,). A typical model would look like:

Y = B0 + B1\*X1 + B2\*X2 + B3\*X3 + … + E

For a dichotomous response variable, we could set up a similar linear model to predict individual category memberships if numerical values are used to represent the two categories. Arbitrary values of 1 and 0 are chosen for mathematical convenience. Using the first example, we would assign Y = 1 if a plant lives and Y = 0 if a plant dies.

This linear model does not work well for a few reasons. First, the response values, 0 and 1, are arbitrary, so modeling the actual values of Y is not exactly of interest. Second, it is the probability that each individual in the population responds with 0 or 1 that we are interested in modeling. For example, we may find that plants with a high level of a fungal infection (X1) fall into the category “the plant lives” (Y) less often than those plants with low level of infection. Thus, as the level of infection rises, the probability of plant living decreases.

Thus, we might consider modeling P, the probability, as the response variable. Again, there are problems. Although the general decrease in probability is accompanied by a general increase in infection level, we know that P, like all probabilities, can only fall within the boundaries of 0 and 1. Consequently, it is better to assume that the relationship between X1 and P is sigmoidal (S-shaped), rather than a straight line.

It is possible, however, to find a linear relationship between X1 and function of P. Although a number of functions work, one of the most useful is the logit function. It is the natural log of the odds that Y is equal to 1, which is simply the ratio of the probability that Y is 1 divided by the probability that Y is 0. The relationship between the logit of P and P itself is sigmoidal in shape. The regression equation that results is:

ln[P/(1-P)] = B0 + B1\*X1 + B2\*X2 + …

Although the left side of this equation looks intimidating, this way of expressing the probability results in the right side of the equation being linear and looking familiar to us. This helps us understand the meaning of the regression coefficients. The coefficients can easily be transformed so that their interpretation makes sense.

The logistic regression equation can be extended beyond the case of a dichotomous response variable to the cases of ordered categories and polytymous categories (more than two categories).

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

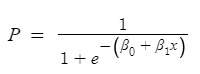
I found this definition on google and now we’ll try to understand it. Logistic Regression is another statistical analysis method borrowed by Machine Learning. It is used when our dependent variable is dichotomous or binary. It just means a variable that has only 2 outputs, for example, A person will survive this accident or not, The student will pass this exam or not. The outcome can either be yes or no (2 outputs). This regression technique is similar to linear regression and can be used to predict the Probabilities for classification problems.

**Logistic Function:**

You must be wondering how logistic regression squeezes the output of linear regression between 0 and 1. If you haven’t read my article on Linear Regression then please have a look at it for a better understanding.

Well, there’s a little bit of math included behind this and it is pretty interesting trust me.

Let’s start by mentioning the formula of logistic function:

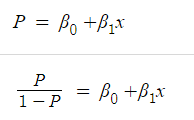


How similar it is too linear regression? If you haven’t read my article on Linear Regression, then please have a look at it for a better understanding.

We all know the equation of the best fit line in linear regression is:

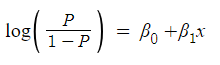
fit linear regression

Let’s say instead of y we are taking probabilities (P). But there is an issue here, the value of (P) will exceed 1 or go below 0 and we know that range of Probability is (0-1). To overcome this issue we take “odds” of P:

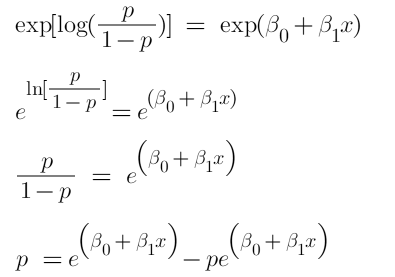


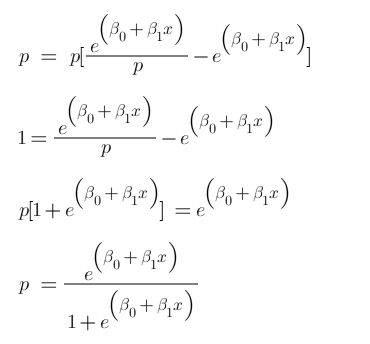
Do you think we are done here? No, we are not. We know that odds can always be positive which means the range will always be (0,+∞ ). Odds are nothing but the ratio of the probability of success and probability of failure. Now the question comes out of so many other options to transform this why did we only take ‘odds’? Because odds are probably the easiest way to do this, that’s it.

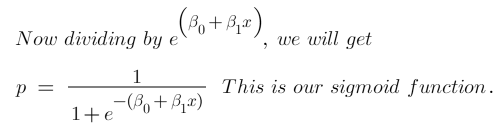
The problem here is that the range is restricted and we don’t want a restricted range because if we do so then our correlation will decrease. By restricting the range we are actually decreasing the number of data points and of course, if we decrease our data points, our correlation will decrease. It is difficult to model a variable that has a restricted range. To control this we take the log of odds which has a range from (-∞,+∞).



If you understood what I did here then you have done 80% of the maths. Now we just want a function of P because we want to predict probability right? not log of odds. To do so we will multiply by exponent on both sides and then solve for P.







Now we have our logistic function, also called a sigmoid function.

**Literature of Learning Management System:**

**Chapter3 System Specification**

Hardware Requirements:

**Hardware Requirements:**

|  |  |
| --- | --- |
| **Hardware Tools** | **Minimum Requirements** |
| Processor | i3 or above |
| Hard Disk | 8GB |
| Monitor | 15” Coloured |
| Mouse | Optical |
| Keyboard | 122 Keys |

Software Requirements:

|  |  |
| --- | --- |
| Software Tools | Minimum Requirements |
| Platform | Windows, Linux or MacOS |
| Operating System | Windows, Linux or MacOS |
| Technology | Windows, Linux or MacOS |
| Scripting Language | Python |
| IDE | Jupyter |

ANACONDA:

INTRODUCTION:

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012.[8] As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, both of which are not free.[6][7]

Package versions in Anaconda are managed by the package management system conda.[9] This package manager was spun out as a separate open-source package as it ended up being useful on its own and for things other than Python.[10] There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depend on, and a small number of other packages.[11]

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator,[12] as a graphical alternative to the command line interface (CLI).

The big difference between conda and the pip package manager is in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists.

Before version 20.3, when pip installed a package, it automatically installed any dependent Python packages without checking if these conflict with previously installed packages. It would install a package and any of its dependencies regardless of the state of the existing installation.[13] Because of this, a user with a working installation of, for example, TensorFlow, could find that it stopped working having used pip to install a different package that requires a different version of the dependent numpy library than the one used by TensorFlow. In some cases, the package would appear to work but produce different results in detail. While pip has since implemented consistent dependency resolution,[13] this difference accounts for a historical differentiation of the conda package manager.[14]

In contrast, conda analyses the current environment including everything currently installed, and, together with any version limitations specified (e.g. the user may wish to have TensorFlow version 2,0 or higher), works out how to install a compatible set of dependencies, and shows a warning if this cannot be done.

Open source packages can be individually installed from the Anaconda repository,[15] Anaconda Cloud (anaconda.org), or the user's own private repository or mirror, using the conda install command. Anaconda, Inc. compiles and builds the packages available in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit and MacOS 64-bit. Anything available on PyPI may be installed into a conda environment using pip, and conda will keep track of what it has installed itself and what pip has installed.

Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud,[16] PyPI or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, it is possible to create new environments that include any version of Python packaged with conda.[17]

**Anaconda Navigator:**

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:[18]

JupyterLab

Jupyter Notebook

QtConsole[19]

Spyder

Glue

Orange

RStudio

Visual Studio Code

**INSTALLATION OF ANACONDA:**

1. Download the Anaconda installer.

2. RECOMMENDED: Verify data integrity with SHA-256. For more information on hashes, see What about cryptographic hash verification?

3. Double click the installer to launch.

|  |
| --- |
| Note   * To prevent permission errors, do not launch the installer from the Favourite’s folder. * If you encounter issues during installation, temporarily disable your anti-virus software during install, and then re-enable it after the installation concludes. If you installed for all users, uninstall Anaconda and re-install it for your user only and try again. |

4. Click Next.

5. Read the licensing terms and click “I Agree”.

6. Select an install for “Just Me” unless you’re installing for all users (which requires Windows Administrator privileges) and click Next.

7. Select a destination folder to install Anaconda and click the Next button. See FAQ.

|  |
| --- |
| Note   * Install Anaconda to a directory path that does not contain spaces or unicode characters. * Do not install as Administrator unless admin privileges are required. |

8. Choose whether to add Anaconda to your PATH environment variable. We recommend not adding Anaconda to the PATH environment variable, since this can interfere with other software. Instead, use Anaconda software by opening Anaconda Navigator or the Anaconda Prompt from the Start Menu.

9. Choose whether to register Anaconda as your default Python. Unless you plan on installing and running multiple versions of Anaconda or multiple versions of Python, accept the default and leave this box checked.

10. Click the Install button. If you want to watch the packages Anaconda is installing, click Show Details.

11. Click the Next button.

12. Optional: To install PyCharm for Anaconda, click on the link to https://www.anaconda.com/pycharm.

Or to install Anaconda without PyCharm, click the Next button.

13. After a successful installation you will see the “Thanks for installing Anaconda” dialog box:

14. If you wish to read more about Anaconda.org and how to get started with Anaconda, check the boxes “Anaconda Individual Edition Tutorial” and “Learn more about Anaconda”. Click the Finish button.

15. Verify your installation.

Python

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. The Python programming language is a dynamically-typed, object-oriented interpreted language. Although, its primary strength lies in the ease with which it allows a programmer to rapidly prototype a project, its powerful and mature set of standard libraries make it a great fit for large-scale production-level software engineering projects as well. Python has a very shallow learning curve and an excellent online learning resource.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

Python Natural Language Processing, gives you the chance to learn about all the aspects of natural language processing (NLP) from scratch. Python natural language analysis package. It contains tools, which can be used in a pipeline, to convert a string containing human language text into lists of sentences and words, to generate base forms of those words, their parts of speech and morphological features, to give a syntactic structure dependency parse, and to recognize named entities.

The first question we usually get asked is what is NLP? The second one is why is Python mainly used for developing NLP applications? And last but not least, the most critical question is what the resources.

Natural language processing strives to build machines that understand and respond to text or voice data—and respond with text or speech of their own—in much the same way humans do.

**What is natural language processing?**

Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to ‘understand’ its full meaning, complete with the speaker or writer’s intent and sentiment.

NLP drives computer programs that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly—even in real time. There’s a good chance you’ve interacted with NLP in the form of voice-operated GPS systems, digital assistants, speech-to-text dictation software, customer service chatbots, and other consumer conveniences. But NLP also plays a growing role in enterprise solutions that help streamline business operations, increase employee productivity, and simplify mission-critical business processes.

**Why is Python mainly used for developing NLP applications?**

The obvious question that we need to encounter at this point is why we should choose Python for AI over others.

Python offers the least code among others and is in fact 1/5 the number compared to other OOP languages. No wonder it is one of the most popular in the market today.

Python has Prebuilt Libraries like Numpy for scientific computation, Scipy for advanced computing and Pybrain for machine learning (Python Machine Learning) making it one of the best languages For AI.

Python developers around the world provide comprehensive support and assistance via forums and tutorials making the job of the coder easier than any other popular languages.

Python is platform Independent and is hence one of the most flexible and popular choiceS for use across different platforms and technologies with the least tweaks in basic coding.

Python is the most flexible of all others with options to choose between OOPs approach and scripting. You can also use IDE itself to check for most codes and is a boon for developers struggling with different algorithms.

Jupyter Notebook:

Introduction:

The notebook extends the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results. The Jupyter notebook combines two components:

A web application: a browser-based tool for interactive authoring of documents which combine explanatory text, mathematics, computations and their rich media output.

Notebook documents: a representation of all content visible in the web application, including inputs and outputs of the computations, explanatory text, mathematics, images, and rich media representations of objects.

Main features of the web application

In-browser editing for code, with automatic syntax highlighting, indentation, and tab completion/introspection. The ability to execute code from the browser, with the results of computations attached to the code which generated them. Displaying the result of computation using rich media representations, such as HTML, LaTeX, PNG, SVG, etc. For example, publication-quality figures rendered by the matplotlib library, can be included inline.

**Notebook documents:**

Notebook documents contains the inputs and outputs of an interactive session as well as additional text that accompanies the code but is not meant for execution. In this way, notebook files can serve as a complete computational record of a session, interleaving executable code with explanatory text, mathematics, and rich representations of resulting objects. These documents are internally JSON files and are saved with the .ipynb extension. Since JSON is a plain text format, they can be version-controlled and shared with colleagues.

Notebooks may be exported to a range of static formats, including HTML (for example, for blog posts), reStructuredText, LaTeX, PDF, and slide shows, via the nbconvert command.

Furthermore, any .ipynb notebook document available from a public URL can be shared via the Jupyter Notebook Viewer <nbviewer>. This service loads the notebook document from the URL and renders it as a static web page. The results may thus be shared with a colleague, or as a public blog post, without other users needing to install the Jupyter notebook themselves. In effect, nbviewer is simply nbconvert as a web service, so you can do your own static conversions with nbconvert, without relying on nbviewer.

Packages:

Packages Used for Chat Bot

tensorflow==2.3.1

nltk==3.5

colorama==0.4.3

numpy==1.18.5

scikit\_learn==0.23.2

Flask==1.1.2

**TensorFlow** has APIs available in several languages both for constructing and executing a TensorFlow graph. The Python API is at present the most complete and the easiest to use, but other language APIs may be easier to integrate into projects and may offer some performance advantages in graph execution.

A word of caution: the APIs in languages other than Python are not yet covered by the API stability promises.

Python

JavaScript

C++

Java

We encourage the community to develop and maintain support for other languages with the approach recommended by the TensorFlow maintainers.

TensorFlow makes it easy for beginners and experts to create machine learning models for desktop, mobile, web, and cloud.

Tensors are multi-dimensional arrays with a uniform type (called a dtype). You can see all supported dtypes at tf.dtypes.DType.

If you're familiar with NumPy, tensors are (kind of) like np.arrays.

All tensors are immutable like Python numbers and strings: you can never update the contents of a tensor, only create a new one.

Basics

Let's create some basic tensors.

Here is a "scalar" or "rank-0" tensor. A scalar contains a single value, and no "axes".

For beginners

The best place to start is with the user-friendly Sequential API. You can create models by plugging together building blocks. Run the “Hello World”.

To learn ML, check out our education page. Begin with curated curriculums to improve your skills in foundational ML areas.

For experts

The Subclassing API provides a define-by-run interface for advanced research. Create a class for your model, and then write the forward pass imperatively. Easily author custom layers, activations, and training loops. Run the “Hello World”.

When to use a Sequential model

A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.

A Sequential model is not appropriate when:

* Your model has multiple inputs or multiple outputs
* Any of your layers has multiple inputs or multiple outputs
* You need to do layer sharing
* You want non-linear topology (e.g. a residual connection, a multi-branch model)

**Natural Language Toolkit**

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project.

NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and “an amazing library to play with natural language.”

Natural Language Processing with Python provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more. The online version of the book has been updated for Python 3 and NLTK 3.

• Token: Before any real processing can be done on the input text, it needs to be segmented into linguistic units such as words, punctuation, numbers or alphanumerics. These units are known as tokens.

• Sentence: An ordered sequence of tokens.

• Tokenization: The process of splitting a sentence into its constituent tokens. For segmented languages such as English, the existence of whitespace makes tokenization relatively easier and uninteresting.

However, for languages such as Chinese and Arabic, the task is more difficult since there are no explicit boundaries. Furthermore, almost all characters in such non-segmented languages can exist as one-character words by themselves but can also join together to form multi-character words.

**• Corpus:** A body of text, usually containing a large number of sentences.

**• Part-of-speech (POS) Tag:** A word can be classified into one or more of a set of lexical or part-of-speech categories such as Nouns, Verbs, Adjectives and Articles, to name a few. A POS tag is a symbol representing such a lexical category - NN(Noun), VB(Verb), JJ(Adjective), AT(Article). One of the oldest and most commonly used tag sets is the Brown Corpus tag set. We will discuss the Brown Corpus in more detail below.

**• Parse Tree:** A tree defined over a given sentence that represents the syntactic structure of the sentence as defined by a formal grammar. Now that we have introduced the basic terminology, let’s look at some common NLP tasks:

**• POS Tagging:** Given a sentence and a set of POS tags, a common language processing task is to automatically assign POS tags to each word in the sentences. For example, given the sentence The ball is red, the output of a POS tagger would be The/AT ball/NN is/VB red/JJ. State-of-the-art POS taggers [9] can achieve accuracy as high as 96%. Tagging text with parts-of-speech turns out to be extremely useful for more complicated NLP tasks such as parsing and machine translation, which are discussed below.

**• Parsing:** In the parsing task, a parser constructs the parse tree given a sentence. Some parsers assume the existence of a set of grammar rules in order to parse but recent parsers are smart enough to deduce the parse trees directly from the given data using complex statistical models. Most parsers also operate in a supervised setting and require the sentence to be POS-tagged before it can be parsed. Statistical parsing is an area of active research in NLP.

**• Machine Translation (MT):** In machine translation, the goal is to have the computer translate the given text in one natural language to fluent text in another language without any human in the loop. This is one of the most difficult tasks in NLP and has been tackled in a lot of different ways over the years. Almost all MT approaches use POS tagging and parsing as preliminary steps.

**Colorama**

Makes ANSI escape character sequences (for producing colored terminal text and cursor positioning) work under MS Windows.

ANSI escape character sequences have long been used to produce colored terminal text and cursor positioning on Unix and Macs. Colorama makes this work on Windows, too, by wrapping stdout, stripping ANSI sequences it finds (which would appear as gobbledygook in the output), and converting them into the appropriate win32 calls to modify the state of the terminal. On other platforms, Colorama does nothing.

This has the upshot of providing a simple cross-platform API for printing colored terminal text from Python, and has the happy side-effect that existing applications or libraries which use ANSI sequences to produce colored output on Linux or Macs can now also work on Windows, simply by calling colorama.init().

An alternative approach is to install ansi.sys on Windows machines, which provides the same behaviour for all applications running in terminals. Colorama is intended for situations where that isn’t easy (e.g., maybe your app doesn’t have an installer.)

Demo scripts in the source code repository print some colored text using ANSI sequences. Compare their output under Gnome-terminal’s built in ANSI handling, versus on Windows Command-Prompt using

Initialisation

Applications should initialise Colorama using:

from colorama import init

init()

On Windows, calling init() will filter ANSI escape sequences out of any text sent to stdout or stderr, and replace them with equivalent Win32 calls.

On other platforms, calling init() has no effect (unless you request other optional functionality; see “Init Keyword Args”, below). By design, this permits applications to call init() unconditionally on all platforms, after which ANSI output should just work.

To stop using Colorama before your program exits, simply call deinit(). This will restore stdout and stderr to their original values, so that Colorama is disabled. To resume using Colorama again, call reinit(); it is cheaper than calling init() again (but does the same thing).

**‘colorama’ module:** Cross-platform printing of colored text can then be done using Colorama’s constant shorthand for ANSI escape sequences:

**NumPy:**

**NumPy** is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an ndarray will create a new array and delete the original.

The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.

NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python’s built-in sequences.

A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today’s scientific/mathematical Python-based software, just knowing how to use Python’s built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

The points about sequence size and speed are particularly important in scientific computing. As a simple example, consider the case of multiplying each element in a 1-D sequence with the corresponding element in another sequence of the same length.

NumPy fully supports an object-oriented approach, starting, once again, with ndarray. For example, ndarray is a class, possessing numerous methods and attributes. Many of its methods are mirrored by functions in the outer-most NumPy namespace, allowing the programmer to code in whichever paradigm they prefer. This flexibility has allowed the NumPy array dialect and NumPy ndarray class to become the de-facto language of multi-dimensional data interchange used in Python.

To work the examples, you’ll need matplotlib installed in addition to NumPy.

Learner profile

This is a quick overview of arrays in NumPy. It demonstrates how n-dimensional () arrays are represented and can be manipulated. In particular, if you don’t know how to apply common functions to n-dimensional arrays (without using for-loops), or if you want to understand axis and shape properties for n-dimensional arrays, this article might be of help.

Learning Objectives

After reading, you should be able to:

Understand the difference between one-, two- and n-dimensional arrays in NumPy;

Understand how to apply some linear algebra operations to n-dimensional arrays without using for-loops;

Understand axis and shape properties for n-dimensional arrays.

The Basics

NumPy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of non-negative integers. In NumPy dimensions are called axes.

NumPy’s array class is called ndarray. It is also known by the alias array. Note that numpy. Array is not the same as the Standard Python Library class array. Array, which only handles one-dimensional arrays and offers less functionality.

The buffer containing the actual elements of the array. Normally, we won’t need to use this attribute because we will access the elements in an array using indexing facilities.

**Printing Arrays**

When you print an array, NumPy displays it in a similar way to nested lists, but with the following layout:

The last axis is printed from left to right,

The second-to-last is printed from top to bottom,

The rest are also printed from top to bottom, with each slice separated from the next by an empty line.

NumPy (Numerical Python) is an open source Python library that’s used in almost every field of science and engineering. It’s the universal standard for working with numerical data in Python, and it’s at the core of the scientific Python and PyData ecosystems. NumPy users include everyone from beginning coders to experienced researchers doing state-of-the-art scientific and industrial research and development. The NumPy API is used extensively in Pandas, SciPy, Matplotlib, scikit-learn, scikit-image and most other data science and scientific Python packages.

The NumPy library contains multidimensional array and matrix data structures (you’ll find more information about this in later sections). It provides ndarray, a homogeneous n-dimensional array object, with methods to efficiently operate on it. NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

Installing NumPy

To install NumPy, we strongly recommend using a scientific Python distribution. If you’re looking for the full instructions for installing NumPy on your operating system, you can find all of the details here.

If you already have Python, you can install NumPy with:

conda install numpy

or

pip install numpy

If you don’t have Python yet, you might want to consider using Anaconda. It’s the easiest way to get started. The good thing about getting this distribution is the fact that you don’t need to worry too much about separately installing NumPy or any of the major packages that you’ll be using for your data analyses, like pandas, Scikit-Learn, etc.

You can find all of the installation details in the Installation section at SciPy.

**How to import NumPy**

To access NumPy and its functions import it in your Python code like this:

import numpy as np

We shorten the imported name to np for better readability of code using NumPy. This is a widely adopted convention that you should follow so that anyone working with your code can easily understand it.

**What’s the difference between a Python list and a NumPy array?**

NumPy gives you an enormous range of fast and efficient ways of creating arrays and manipulating numerical data inside them. While a Python list can contain different data types within a single list, all of the elements in a NumPy array should be homogeneous. The mathematical operations that are meant to be performed on arrays would be extremely inefficient if the arrays weren’t homogeneous.

**Why use NumPy?**

NumPy arrays are faster and more compact than Python lists. An array consumes less memory and is convenient to use. NumPy uses much less memory to store data and it provides a mechanism of specifying the data types. This allows the code to be optimized even further.

**What is an array?**

An array is a central data structure of the NumPy library. An array is a grid of values and it contains information about the raw data, how to locate an element, and how to interpret an element. It has a grid of elements that can be indexed in various ways. The elements are all of the same type, referred to as the array dtype.

An array can be indexed by a tuple of nonnegative integers, by booleans, by another array, or by integers. The rank of the array is the number of dimensions. The shape of the array is a tuple of integers giving the size of the array along each dimension.

One way we can initialize NumPy arrays is from Python lists, using nested lists for two- or higher-dimensional data.

**scikit-learn:**

In general, a learning problem considers a set of n samples of data and then tries to predict properties of unknown data. If each sample is more than a single number and, for instance, a multi-dimensional entry (aka multivariate data), it is said to have several attributes or features.

Learning problems fall into a few categories:

Supervised learning, in which the data comes with additional attributes that we want to predict (Click here to go to the scikit-learn supervised learning page).This problem can be either:

Classification: samples belong to two or more classes and we want to learn from already labelled data how to predict the class of unlabelled data. An example of a classification problem would be handwritten digit recognition, in which the aim is to assign each input vector to one of a finite number of discrete categories. Another way to think of classification is as a discrete (as opposed to continuous) form of supervised learning where one has a limited number of categories and for each of the n samples provided, one is to try to label them with the correct category or class.

Regression: if the desired output consists of one or more continuous variables, then the task is called regression. An example of a regression problem would be the prediction of the length of a salmon as a function of its age and weight.

Unsupervised learning, in which the training data consists of a set of input vectors x without any corresponding target values. The goal in such problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of visualization (Click here to go to the Scikit-Learn unsupervised learning page).

**What is Scikit-learn?**

Scikit-learn is an open-source Python library for machine learning. It supports state-of-the-art algorithms such as KNN, XGBoost, random forest, and SVM. It is built on top of NumPy. Scikit-learn is widely used in Kaggle competition as well as prominent tech companies. It helps in preprocessing, dimensionality reduction (parameter selection), classification, regression, clustering, and model selection.

**Training set and testing set**

Machine learning is about learning some properties of a data set and then testing those properties against another data set. A common practice in machine learning is to evaluate an algorithm by splitting a data set into two. We call one of those sets the training set, on which we learn some properties; we call the other set the testing set, on which we test the learned properties.

**Learning and predicting**

In the case of the digits dataset, the task is to predict, given an image, which digit it represents. We are given samples of each of the 10 possible classes (the digits zero through nine) on which we fit an estimator to be able to predict the classes to which unseen samples belong.

In scikit-learn, an estimator for classification is a Python object that implements the methods fit(X, y) and predict (T).

An example of an estimator is the class sklearn.svm.SVC, which implements support vector classification. The estimator’s constructor takes as arguments the model’s parameters.

**An introduction to machine learning with scikit-learn**

* Machine learning: the problem setting
* Loading an example dataset
* Learning and predicting
* Model persistence
* Conventions

Scikit-learn is largely written in Python, and uses NumPy extensively for high-performance linear algebra and array operations. Furthermore, some core algorithms are written in Cython to improve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR. In such cases, extending these methods with Python may not be possible.

Scikit-learn integrates well with many other Python libraries, such as Matplotlib and plotly for plotting, NumPy for array vectorization, Pandas dataframes, SciPy, and many more.

Scikit-learn is not very difficult to use and provides excellent results. However, scikit learn does not support parallel computations. It is possible to run a deep learning algorithm with it but is not an optimal solution, especially if you know how to use TensorFlow.

**Scikit-Learn Example with Machine Learning**

This Scikit tutorial is divided into two parts:

**Machine learning with scikit-learn**

**How to trust your model with LIME**

The first part details how to build a pipeline, create a model and tune the hyperparameters while the second part provides state-of-the-art in term of model selection.

**Flask:**

Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

Flask offers suggestions, but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy.

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries.[2] It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, and upload handling, various open authentication technologies and several common framework related tools

Flask was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004. According to Ronacher, the idea was originally an April Fool's joke that was popular enough to make into a serious application. The name is a play on the earlier Bottle framework.

When Ronacher and Georg Brandl created a bulletin board system written in Python, the Pocoo projects Werkzeug and Jinja were developed.

In April 2016, the Pocoo team was disbanded and development of Flask and related libraries passed to the newly formed Pallets project

Flask has become popular among Python enthusiasts. As of October 2020, it has second most stars on GitHub among Python web-development frameworks, only slightly behind Django, and was voted the most popular web framework in the Python Developers Survey 2018.

**JSON (JavaScript Object Notation)** :

JSON is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

json exposes an API familiar to users of the standard library marshal and pickle modules. It is the externally maintained version of the json library contained in Python 2.6, but maintains compatibility with Python 2.4 and Python 2.5 and (currently) has significant performance advantages, even without using the optional C extension for speedups.

JSON is built on two structures:

A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.

An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

**Python Supports JSON Natively!**

Python comes with a built-in package called json for encoding and decoding JSON data.

Just by using “import JSON”

**Implementation Code snippets**

Required Packages

The required python packages are as follows,

tensorflow==2.3.1

nltk==3.5

colorama==0.4.3

numpy==1.18.5

scikit\_learn==0.23.2

Flask==1.1.2

Define Intents

I will define few simple intents and bunch of messages that corresponds to those intents and also map some responses according to each intent category.

JSON file named “intents.json” including these data as follows.

{"intents": [

{"tag": "greeting",

"patterns": ["Hi", "Hey", "Is anyone there?", "Hello", "Hay","Yo"],

"responses": ["Hello", "Hi", "Hi there"]

},

{"tag": "goodbye",

"patterns": ["Bye", "See you later", "Goodbye"],

"responses": ["See you later", "Have a nice day", "Bye! Come back again"]

},

{"tag": "thanks",

"patterns": ["Thanks", "Thank you", "That's helpful", "Thanks for the help"],

"responses": ["Happy to help!", "Any time!", "My pleasure", "You're most welcome!"]

},

{"tag": "who are you",

"patterns": ["Who are you?", "What are you?", "Who you are?" ],

"responses": ["I.m Personalized AI bot, your Personalized AI bot assistant", "I'm Artificial Intelligent bot which I'm here to help you"]

},

{"tag": "name",

"patterns": ["what is your name", "what should I call you", "whats your name?"],

"responses": ["You can call me Tina.", "I'm Tina!", "Just call me as Tina !!!"]

},

{"tag": "help",

"patterns": ["Could you help me?", "give me a hand please", "Can you help?", "What can you do for me?", "I need a support", "I need a help", "support me please"],

"responses": ["Tell me how can assist you", "Tell me your problem to assist you", "Yes Sure, How can I support you"]

},

{"tag": "createaccount",

"patterns": ["I need to create a new account", "how to open a new account", "I want to create an account", "can you create an account for me", "how to open a new account"],

"responses": ["You can just easily create a new account from our web site or https://internsoft.com/dashboard/", "Just go to our web site and follow the guidelines to create a new account or https://internsoft.com/dashboard/"]

},

{"tag": "complaint",

"patterns": ["have a complaint", "I want to raise a complaint", "there is a complaint about a service"],

"responses": ["Please provide us your complaint in order to assist you", "Please mention your complaint, we will reach you and sorry for any inconvenience caused"]

},

{"tag": "course",

"patterns": ["How many courses are there","what all courses can i get","courses"],

"responses": ["There are of 1.Full Stack Software Development, 2.Data Analysis Using Python, 3.Internet Of Things Architecture And Embedded Systems Development, 4.Machine Learning With Robotics Language Automations, 5.Specialization in Data Science & Artificial Intelligence."]

},

{"tag": "contact",

"patterns":["Contact","Need more information","speak to manager"],

"responses":["Additional queries ? whatsapp Us .We'll reach you back ASAP +91 99644 83 843 +91 74833 00 668"]

},

{"tag": "management",

"patterns":["who is the founder of this company"],

"responses":[" Syed Asad Founder & CEO - Airobosoft products and services. Data Scientist in Artificial intelligence and Robotics. https://internsoft.com/"]

},

{"tag": "about",

"patterns":["About the company","Company"],

"responses":[" Internsoft is the website Launched by AiRobosoft Products & Services which is intentend to showcase the learning thy have created this website just to value the learning making it easy for the people who are triving to learn his was setup on year 2020"]

}

]

}

Data Preparation

Importing all the required packages:

import json

import numpy as np

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Embedding, GlobalAveragePooling1D

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from sklearn.preprocessing import LabelEncoder

load the json file and extract the required data.

with open('intents.json') as file:

data = json.load(file)

training\_sentences = []

training\_labels = []

labels = []

responses = []

for intent in data['intents']:

for pattern in intent['patterns']:

training\_sentences.append(pattern)

training\_labels.append(intent['tag'])

responses.append(intent['responses'])

if intent['tag'] not in labels:

labels.append(intent['tag'])

num\_classes = len(labels)

The variable “training\_sentences” holds all the training data (which are the sample messages in each intent category) and the “training\_labels” variable holds all the target labels correspond to each training data.

Then we use “LabelEncoder()” function provided by scikit-learn to convert the target labels into a model understandable form.

lbl\_encoder = LabelEncoder()

lbl\_encoder.fit(training\_labels)

training\_labels = lbl\_encoder.transform(training\_labels)

Next, we vectorize our text data corpus by using the “Tokenizer” class and it allows us to limit our vocabulary size up to some defined number. When we use this class for the text pre-processing task, by default all punctuations will be removed, turning the texts into space-separated sequences of words, and these sequences are then split into lists of tokens. They will then be indexed or vectorized. We can also add “oov\_token” which is a value for “out of token” to deal with out of vocabulary words(tokens) at inference time.

vocab\_size = 1000

embedding\_dim = 16

max\_len = 20

oov\_token = "<OOV>"

tokenizer = Tokenizer(num\_words=vocab\_size, oov\_token=oov\_token)

tokenizer.fit\_on\_texts(training\_sentences)

word\_index = tokenizer.word\_index

sequences = tokenizer.texts\_to\_sequences(training\_sentences)

padded\_sequences = pad\_sequences(sequences, truncating='post', maxlen=max\_len)

The “pad\_sequences” method is used to make all the training text sequences into the same size.

Model Training

Let’s define our Neural Network architecture for the proposed model and for that we use the “Sequential” model class of Keras.

model = Sequential()

model.add(Embedding(vocab\_size, embedding\_dim, input\_length=max\_len))

model.add(GlobalAveragePooling1D())

model.add(Dense(16, activation='relu'))

model.add(Dense(16, activation='relu'))

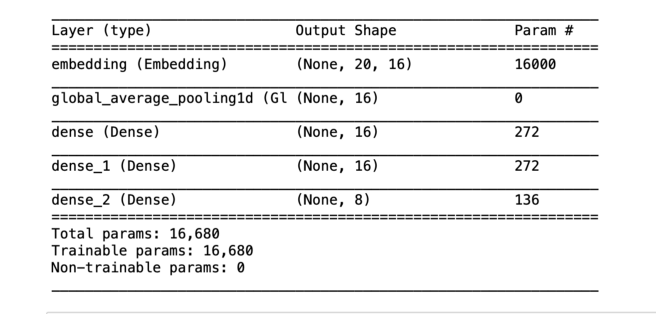
model.add(Dense(num\_classes, activation='softmax'))

model.compile(loss='sparse\_categorical\_crossentropy',

optimizer='adam', metrics=['accuracy'])

model.summary()

Our model architecture looks as follows.



Now we are ready to train our model. Simply we can call the “fit” method with training data and labels.

epochs = 500

history = model.fit(padded\_sequences, np.array(training\_labels), epochs=epochs)

After training, it is better to save all the required files in order to use it at the inference time. So that we save the trained model, fitted tokenizer object and fitted label encoder object.

# to save the trained model

model.save("chat\_model")

import pickle

# to save the fitted tokenizer

with open('tokenizer.pickle', 'wb') as handle:

pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST\_PROTOCOL)

# to save the fitted label encoder

with open('label\_encoder.pickle', 'wb') as ecn\_file:

pickle.dump(lbl\_encoder, ecn\_file, protocol=pickle.HIGHEST\_PROTOCOL)

Implementation a chat function to engage with a real user. When a new user message is received, the Chabot will calculate the similarity between the new text sequence and training data. Considering the confidence scores got for each category, it categorizes the user message to intent with the highest confidence score.

import json

import numpy as np

from tensorflow import keras

from sklearn.preprocessing import LabelEncoder

import colorama

colorama.init()

from colorama import Fore, Style, Back

import random

import pickle

with open("intents.json") as file:

data = json.load(file)

def chat():

# load trained model

model = keras.models.load\_model('chat\_model')

# load tokenizer object

with open('tokenizer.pickle', 'rb') as handle:

tokenizer = pickle.load(handle)

# load label encoder object

with open('label\_encoder.pickle', 'rb') as enc:

lbl\_encoder = pickle.load(enc)

# parameters

max\_len = 20

while True:

print(Fore.LIGHTBLUE\_EX + "User: " + Style.RESET\_ALL, end="")

inp = input()

if inp.lower() == "quit":

break

result = model.predict(keras.preprocessing.sequence.pad\_sequences(tokenizer.texts\_to\_sequences([inp]),

truncating='post', maxlen=max\_len))

tag = lbl\_encoder.inverse\_transform([np.argmax(result)])

for i in data['intents']:

if i['tag'] == tag:

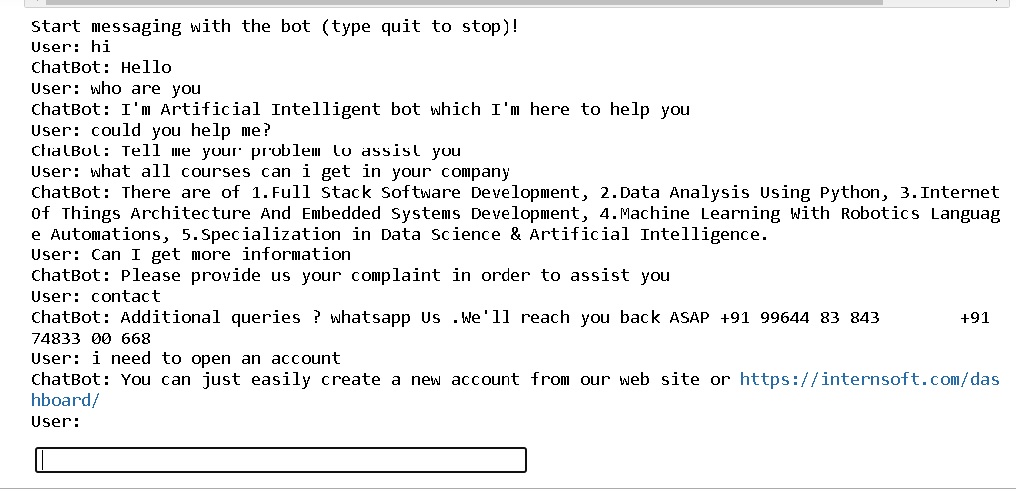
print(Fore.GREEN + "ChatBot:" + Style.RESET\_ALL , np.random.choice(i['responses']))

# print(Fore.GREEN + "ChatBot:" + Style.RESET\_ALL,random.choice(responses))

print(Fore.YELLOW + "Start messaging with the bot (type quit to stop)!" + Style.RESET\_ALL)

chat()

RESULT:



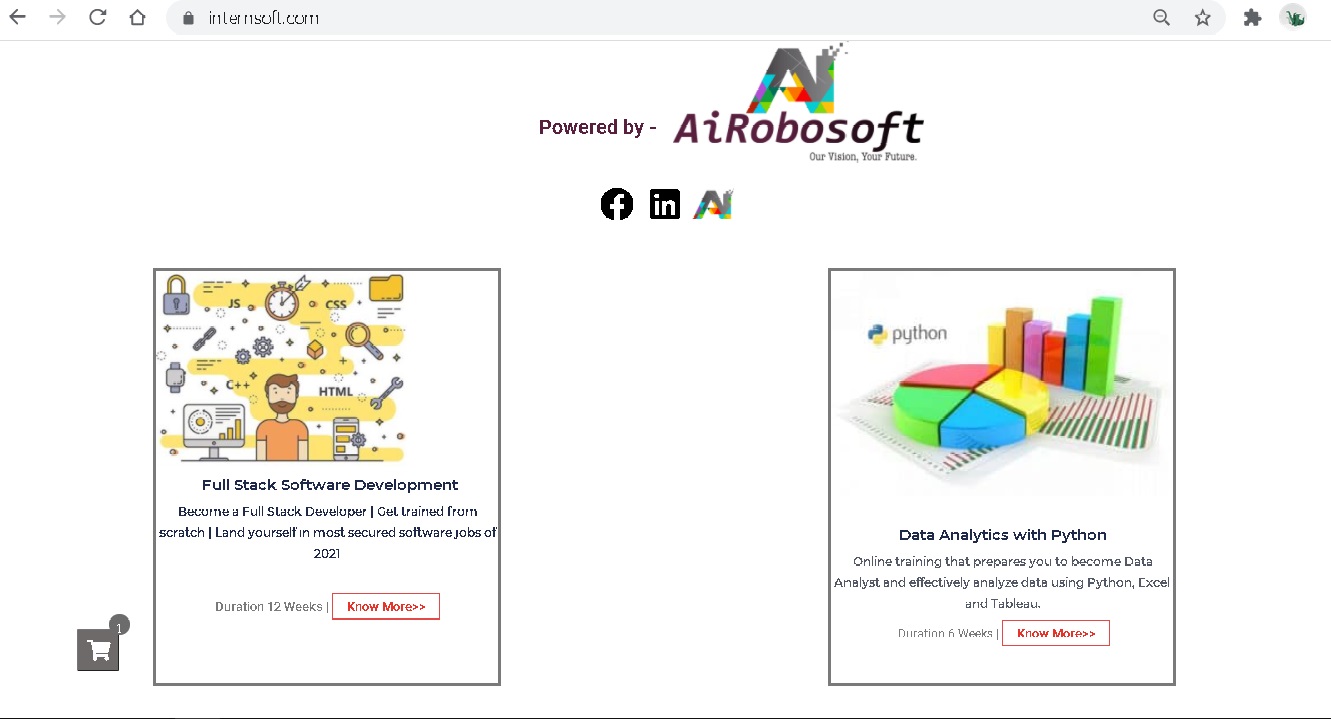
Chapter 4:

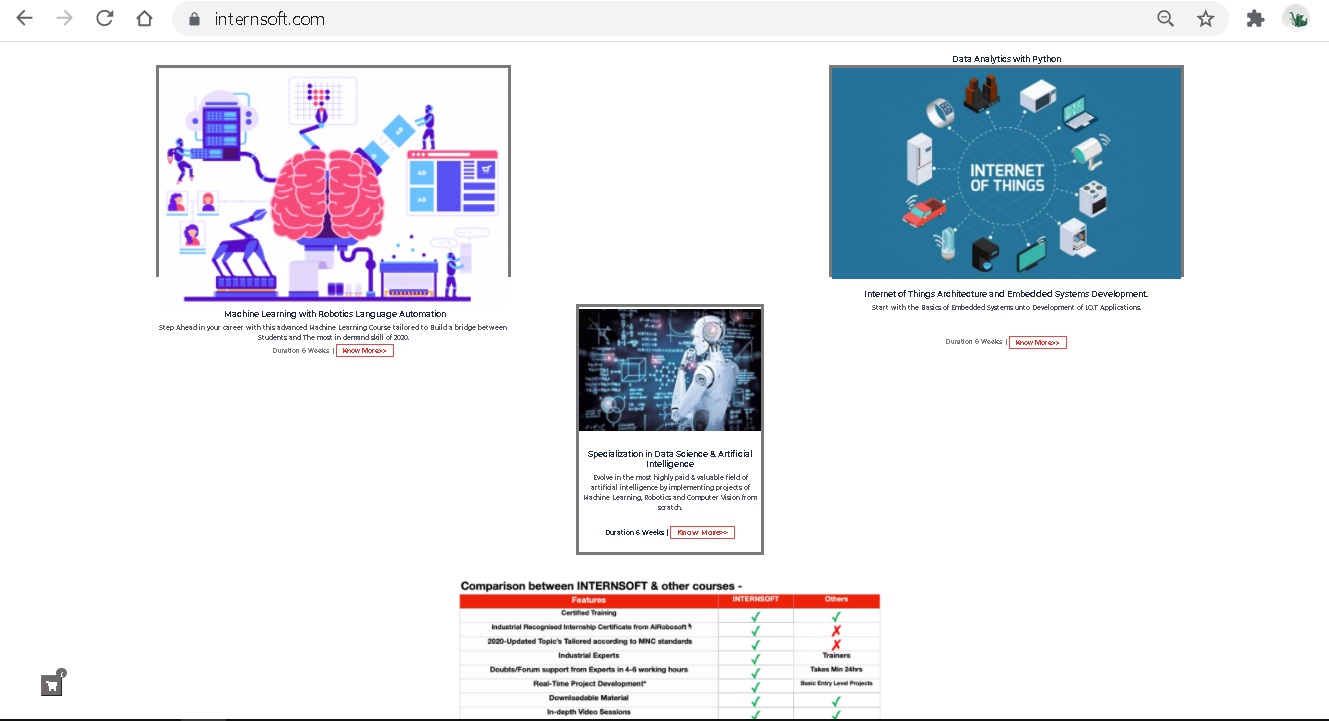
Modules

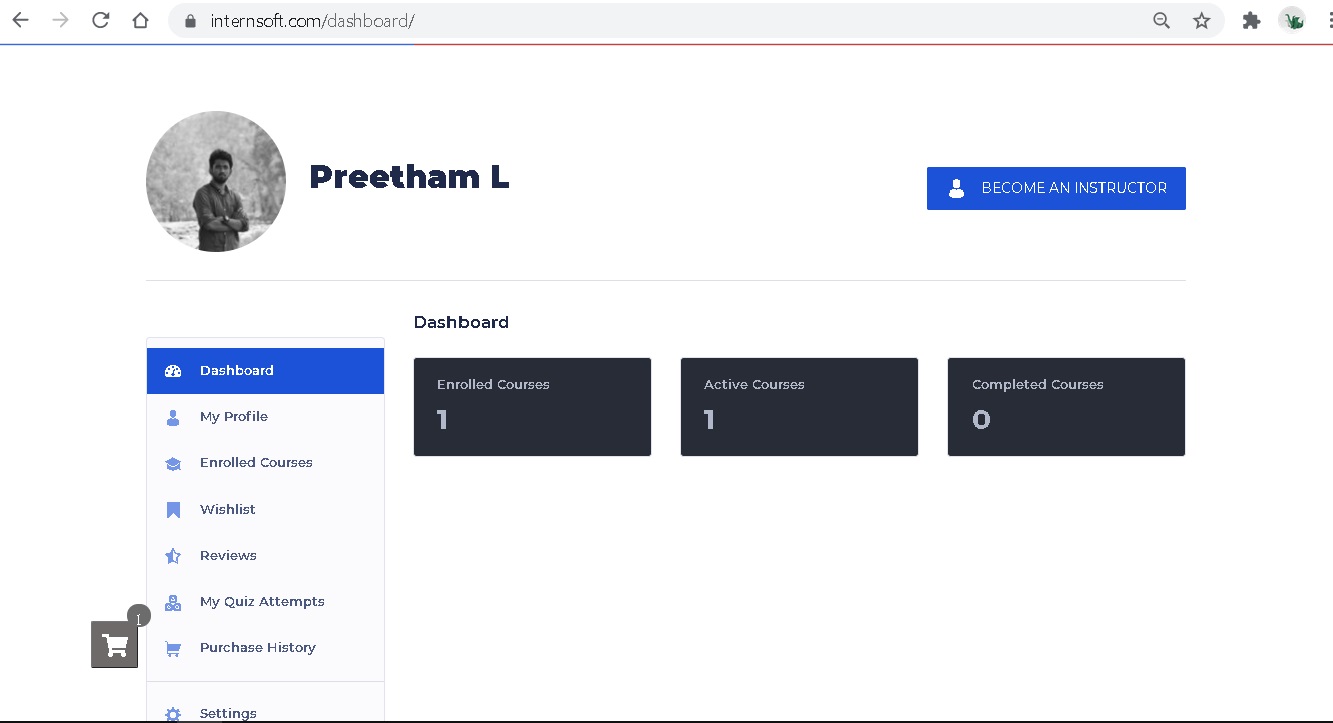
Chapter 5:

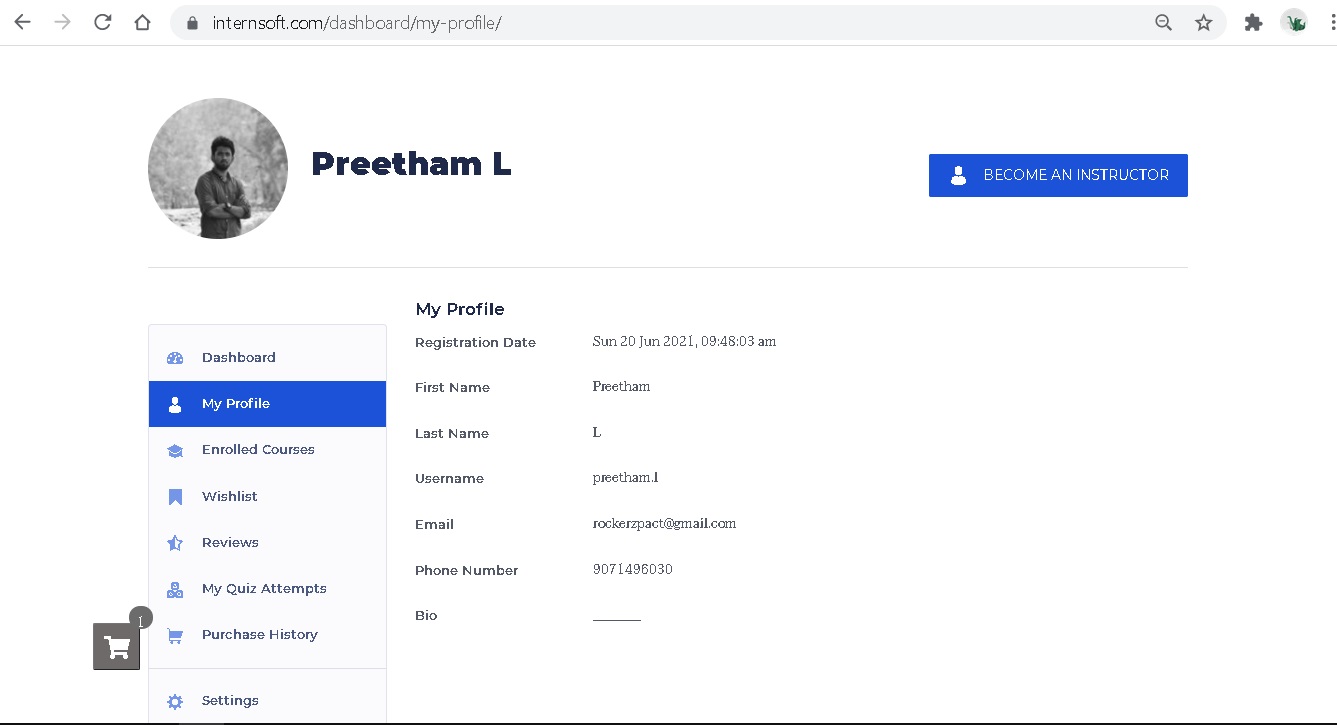
Screenshots:

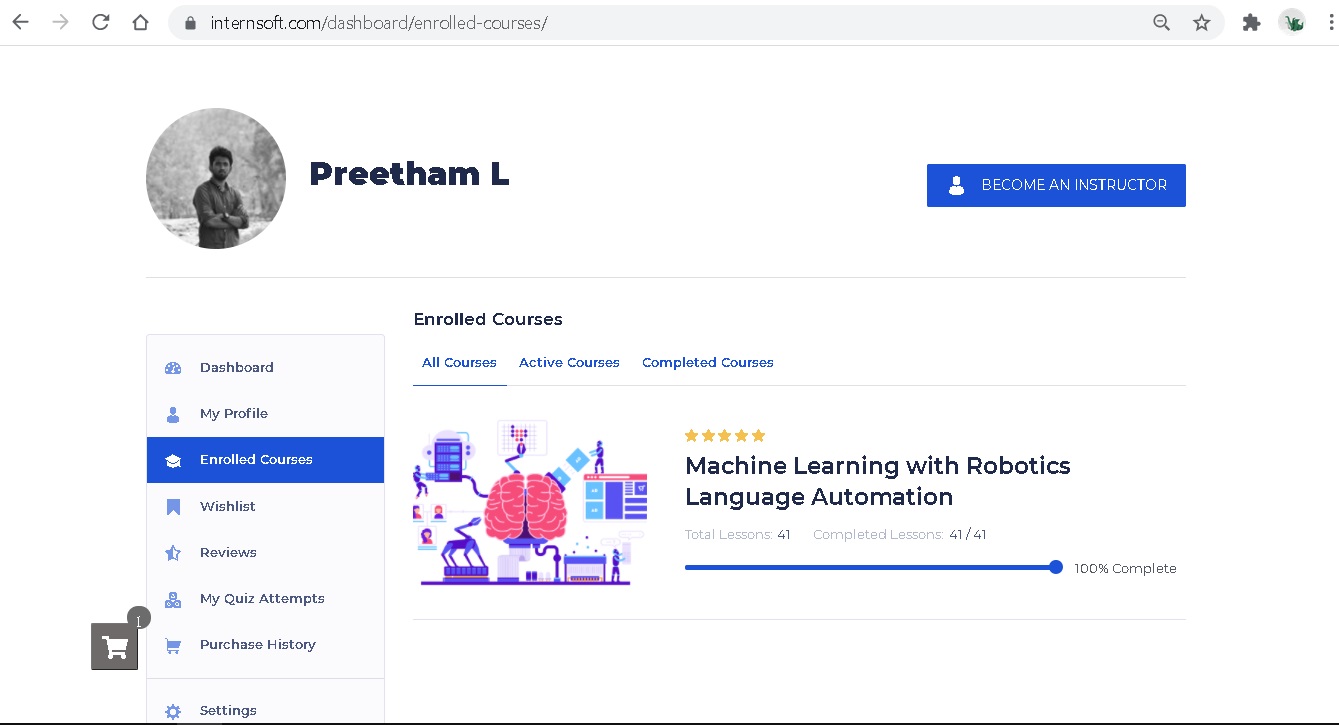


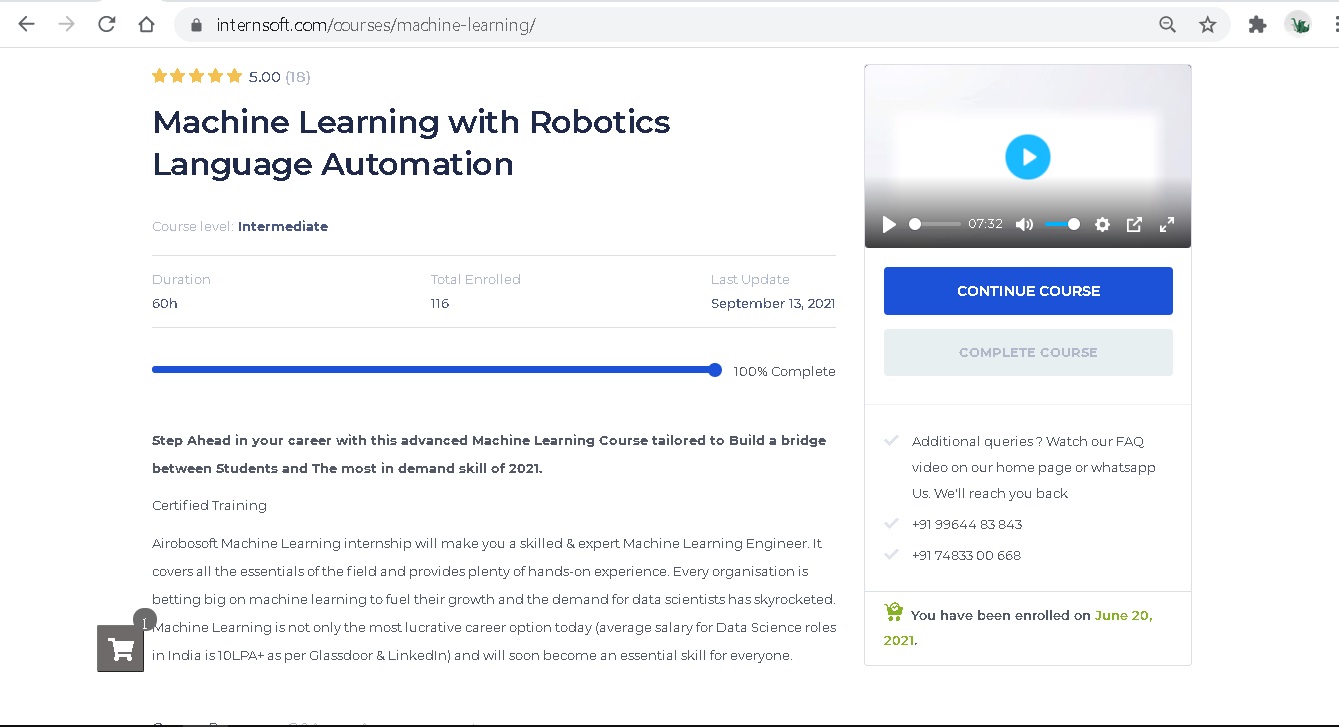


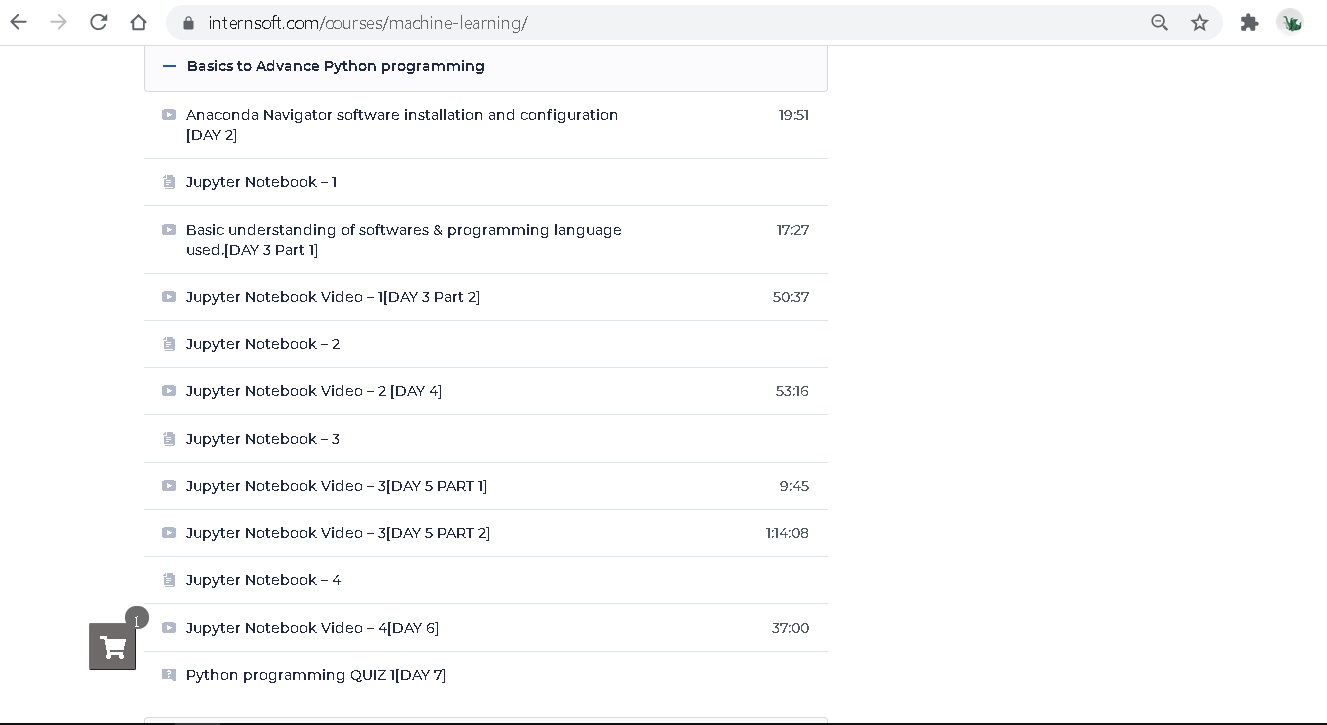


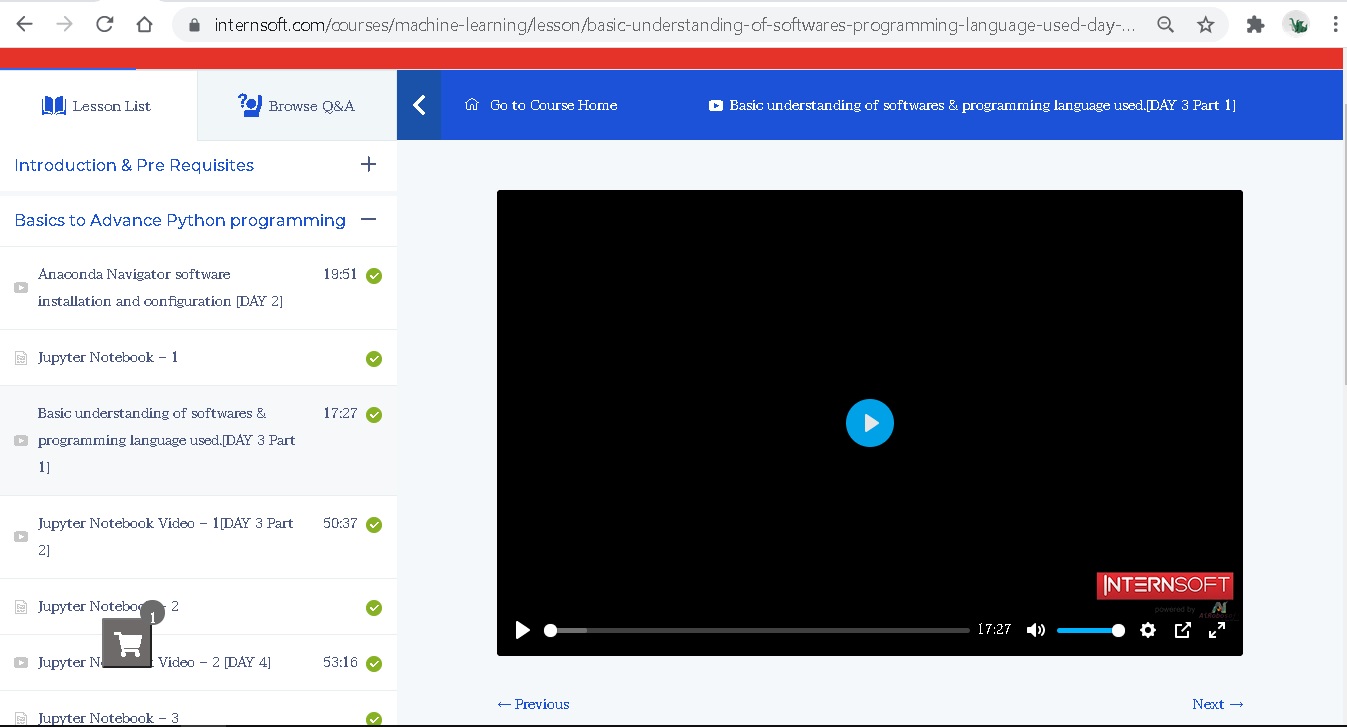


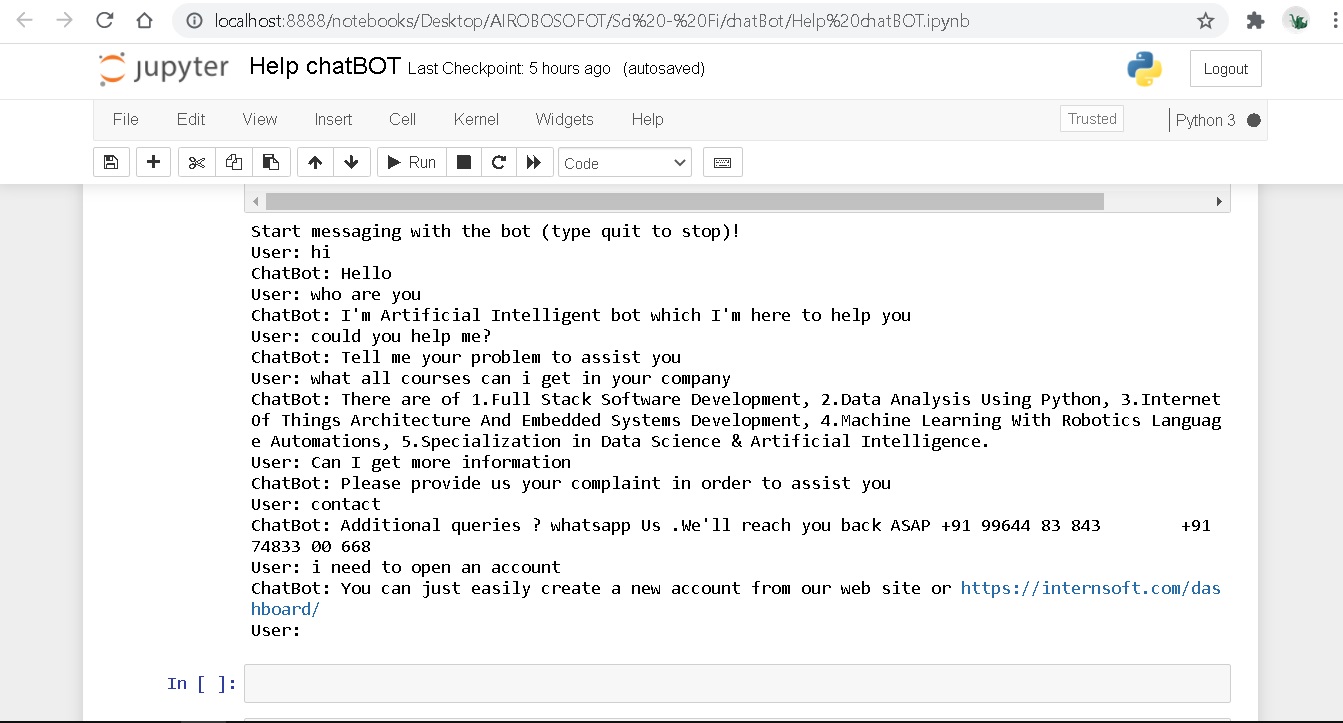












References :

* Li J, Monroe W, Ritter A, Galley M, Gao J, Jurafsky D. Deep reinforcement learning for

dialogue generation. arXiv preprint arXiv:1606.01541. 2016 Jun 5.

* Neural Machine Translation (seq2seq) Tutorial https://www.tensorflow.org/

tutorials/seq2seq

* <https://www.tensorflow.org/overview>
* <https://www.tensorflow.org/learn>
* https://www.nltk.org/

nltk(

[1] Dan Bikel. 2004. On the Parameter Space of Generative Lexicalized Statistical Parsing Models. Ph.D. Thesis. http://www.cis.upenn.edu/~dbikel/papers/thesis.pdf

[2] David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. Proceedings of ACL.

[3] Kenneth W. Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. Computational Linguistics. 16(1).

[9] Adwait Ratnaparkhi 1996. A Maximum Entropy Part-Of-Speech Tagger. Proceedings of Empirical Methods on Natural Language Processing.)

* <https://pypi.org/project/colorama/>
* <https://numpy.org/doc/stable/index.html>
* <https://scikit-learn.org/stable/>
* *Dreijer, Janto.*[*"scikit-learn"*](https://scikits.appspot.com/scikit-learn)*.*

[**^**](https://en.wikipedia.org/wiki/Scikit-learn#cite_ref-5) [*"About us — scikit-learn 0.20.1 documentation"*](https://scikit-learn.org/stable/about.html#history)*. scikit-learn.org.*

[**^**](https://en.wikipedia.org/wiki/Scikit-learn#cite_ref-6) *Eli Bressert (2012).*[*SciPy and NumPy: an overview for developers*](https://books.google.com/books?id=fLKTuJqQLVEC&pg=PA43)*. O'Reilly. p. 43.*

* The Official Python Tutorial. <http://docs.python.org/tut/tut.html>
* <https://flask.palletsprojects.com/>
* [*"Flask Foreword"*](https://web.archive.org/web/20171117015927/http:/flask.pocoo.org/docs/0.10/foreword). Archived from [*the original*](http://flask.pocoo.org/docs/0.10/foreword) on 2017-11-17.
* [**^**](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_ref-3) [*"Flask Extensions"*](https://web.archive.org/web/20180517082208/http:/flask.pocoo.org/extensions/). Archived from [*the original*](http://flask.pocoo.org/extensions/) on 2018-05-17.
* [*"Pocoo Team"*](https://web.archive.org/web/20180315200205/http:/www.pocoo.org/team/). Archived from [*the original*](http://www.pocoo.org/team/) on 2018-03-15.
* ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_ref-openingflask_7-0) [***b***](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_ref-openingflask_7-1) Ronacher, Armin. [*"Opening the Flask"*](https://web.archive.org/web/20161217230320/http:/mitsuhiko.pocoo.org/flask-pycon-2011.pdf) *(PDF)*. Archived from [*the original*](http://mitsuhiko.pocoo.org/flask-pycon-2011.pdf) *(PDF)* on 2016-12-17*. Retrieved 2011-09-30*.
* [**^**](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_ref-8) Ronacher, Armin (3 April 2010). [*"April 1st Post Mortem"*](https://web.archive.org/web/20180514202042/http:/lucumr.pocoo.org/2010/4/3/april-1st-post-mortem/). Armin Ronacher's Thoughts and Writings. Archived from [*the original*](http://lucumr.pocoo.org/2010/4/3/april-1st-post-mortem/) on 2018-05-14*. Retrieved 2015-07-25*.
* [**^**](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_ref-aprilfoolspage_9-0) [*"Denied: the next generation python micro-web-framework (April Fools page)"*](https://web.archive.org/web/20110904053655/http:/denied.immersedcode.org/). Archived from [*the original*](http://denied.immersedcode.org/) on 2011-09-04*. Retrieved 2011-09-30*.
* [**^**](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_ref-10) [*"History"*](https://web.archive.org/web/20171119213325/http:/www.pocoo.org/history/). Pocoo Team. Archived from [*the original*](http://www.pocoo.org/history/) on 2017-11-19*. Retrieved 2015-03-25*.
* [**^**](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_ref-11) Ronacher, Armin (2016-04-01). [*"Hello Pallets Users"*](https://palletsprojects.com/blog/hello/). The Pallets Projects*. Retrieved 2021-05-08*.
* [**^**](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_ref-12) [*"Pocoo"*](https://www.pocoo.org/). www.pocoo.org*. Retrieved 2021-05-08*.
* [**^**](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_ref-13) [*"Python libraries by GitHub stars"*](https://github.com/search?l=Python&q=stars%3A%3E100&s=stars&type=Repositories). Github*. Retrieved 2020-01-27*.
* [**^**](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_ref-14) [*"Python Developers Survey 2018"*](https://www.jetbrains.com/research/python-developers-survey-2018). www.jetbrains.com. 2018-11-01.
* <https://www.ibm.com/cloud/learn/natural-language-processing>
* <https://www.json.org/json-en.html>
* [*"What is Anaconda, Inc.?"*](https://docs.anaconda.com/anaconda-cloud/faq#what-is-anaconda-inc). docs.anaconda.com. Anaconda is a software development and consulting company of passionate open source advocates based in Austin, Texas, USA. We are committed to the open source community. We created the Anaconda Python distribution and contribute to many other open source-based data analytics tools.
* [**^**](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-2) Collison, Scott (2017-06-28). [*"Continuum Analytics Officially Becomes Anaconda"*](https://www.anaconda.com/continuum-analytics-officially-becomes-anaconda/). Anaconda Inc. corporate website*. Retrieved 22 April 2020*.
* [**^**](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-3) [*"Archived copy"*](https://web.archive.org/web/20181012114953/http:/docs.anaconda.com/anaconda/release-notes/#id53). Archived from [*the original*](https://docs.anaconda.com/anaconda/release-notes#id53) on 2018-10-12*. Retrieved 2017-10-26*.
* [**^**](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-4) [*"Release notes — Anaconda documentation"*](https://docs.anaconda.com/anaconda/reference/release-notes/).
* [**^**](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-5) [*"End User License Agreement - Anaconda® Individual Edition"*](https://www.anaconda.com/eula-anaconda-individual-edition). anaconda.com. Anaconda, Inc*. Retrieved 29 January 2021*.
* ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-l1_6-0) [***b***](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-l1_6-1) [*"Cover page"*](https://docs.anaconda.com/). Anaconda Documentation. Anaconda, Inc*. Retrieved 27 April 2020*.
* ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-l2_7-0) [***b***](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-l2_7-1) [*"Products and Pricing"*](https://www.anaconda.com/pricing). anaconda.com. Anaconda, Inc*. Retrieved 28 April 2020*.
* [**^**](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-8) [*"About Anaconda"*](https://web.archive.org/web/20200419034550/https:/www.anaconda.com/media-kit/). Archived from [*the original*](https://www.anaconda.com/media-kit/) on 19 April 2020*. Retrieved 27 April 2020*.
* [**^**](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-9) [*"Conda – Conda documentation"*](http://conda.pydata.org/docs/)*. Retrieved February 25, 2016*.
* [**^**](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-10) [*"What's the difference between Anaconda, conda, and Miniconda?"*](https://bioconda.github.io/contributor/faqs.html#conda-anaconda-minconda). FAQ - Bioconda documentation*. Retrieved 22 April2020*.
* [**^**](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-11) [*"Miniconda"*](https://web.archive.org/web/20180910152348/https:/conda.io/docs/glossary.html#miniconda-glossary). conda.io. Archived from [*the original*](https://conda.io/docs/glossary.html#miniconda-glossary) on 2018-09-10*. Retrieved 2018-07-11*.
* [**^**](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-12) [*"Anaconda Navigator"*](https://docs.anaconda.com/anaconda/navigator/). docs.anaconda.com.
* ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-:0_13-0) [***b***](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-:0_13-1) [*"User Guide - pip documentation v21.1.dev0"*](https://pip.pypa.io/en/latest/user_guide/#changes-to-the-pip-dependency-resolver-in-20-3-2020). pip.pypa.io*. Retrieved 2021-04-07*.
* [**^**](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-14) [*"Anaconda | Understanding Conda and Pip"*](https://www.anaconda.com/blog/understanding-conda-and-pip). Anaconda*. Retrieved 2021-04-07*.
* ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-AnacondaRepo_15-0) [***b***](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-AnacondaRepo_15-1) [*"Anaconda repository"*](https://anaconda.org/anaconda/repo). anaconda.org.
* [**^**](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-AnacondaCloud_16-0) [*"Anaconda Cloud"*](https://anaconda.org/). anaconda.org.
* [**^**](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_ref-17) [*"Managing Python with conda"*](https://web.archive.org/web/20180613160441/https:/conda.io/docs/user-guide/tasks/manage-python.html). conda.io. Archived from [*the original*](https://conda.io/docs/user-guide/tasks/manage-python.html) on 2018-06-13*. Retrieved 2018-06-13*.