# **Hands-On Natural Language Processing**

### **Gain insights from unstructured text data**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] It is estimated that between 80 to 90% of business data is unstructured and the organizations that can effectively glean insight from unstructured data, such as text, have a competitive advantage over those who don't. In this course, I'll teach you how to use different natural language processing technique to analyze text data. My name Wuraola Oyewusi. As an experienced data scientist, some of my most important work is related to natural language processing. If one fact about me, I have a lot of experience learning online. So, come on this journey with me as we explore different natural language processing tasks with practical use cases.

### **What you should know**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] To get the best out of this course, I'm assuming that you have a fundamental understanding of **Python programming, machine learning, and natural language processing**. Feel free to follow along, but also just to explore courses on those topics.

### **Exercise files**

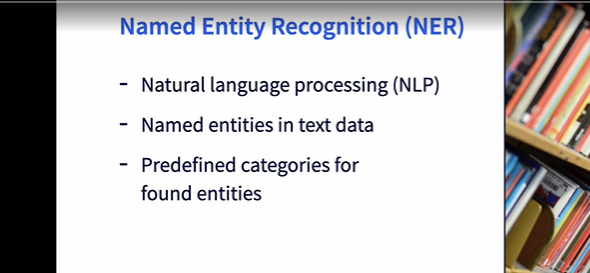
Selecting transcript lines in this section will navigate to timestamp in the video

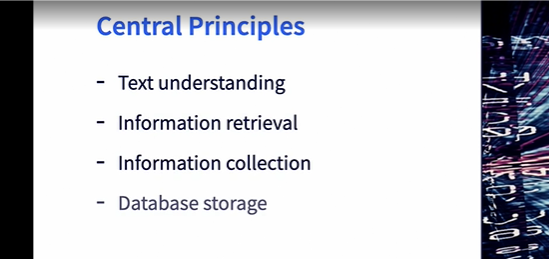
- This course is designed to be hands on. So, it is encouraged that you walk along with the provided exercise files. We will be using Google Colab to run our code. Let's take a brief tour of how it works by going to the file I shared on colab.research.google. To upload an IPython file, you click on the file button, and then upload Notebook. Now that we have our Notebook, to run a cell, you click here. And to upload another file type, you come to files and then click on upload to storage session and select the file. Sometimes when you pip install the library you may need to restart the page by clicking on runtime, restart runtime. That's it for now. Let's get started.

### **What is named entity recognition (NER)?**

Selecting transcript lines in this section will navigate to timestamp in the video

- Named entity recognition is a natural language processing technique that locates named entities in a structured text data and classifies entities into predefined categories. **It is also called entity extraction, entity identification or entity chunking.** The algorithm is able to recognize named entities, searchers, people's and companies names, addresses, dates, expression quantities, monetary values and percentages. **And the machine learning used for named entity recognition can either be supervised, where the training data is labeled, or unsupervised, where the training data is not labeled. In practice, the supervised machine learning is the most popular approach. The central principle of named entity recognition is to understand what's in the text, retrieving collect important information for storage into databases. Some of the most popular frameworks are spaCy, scispaCy, which is like spaCy, but it focuses on scientific and clinical documents. Then there is flairNLP, AllenNLP, Stanza and Stafford NER. All those frameworks are based on Python, except for Stafford NER, which is based on Java.** Now, there are many practical use cases of named entity recognition. In fact, you may already have come across them and may not even know it. You can apply named entity recognition to review customer feedback and detect recurring problems in a certain location. For example, you can automatically categorize customer support tickets by product name or type to route the ticket to the appropriate agent. Now, if you watch movies, listen to music or browse products online, named entity recognition systems are probably improving the efficiency of the recommendation. They can also help recruiters save a lot of time when reviewing hundreds of resumes. The algorithm can extract relevant information about candidates, such as their name, email, degrees, work experience and so on automatically. And then there is plain text cataloging where the type of texts is determined based on valid recognized entities. So as you can see, named entity recognition techniques keep those valuable tools to speed up our work to better evaluate large amounts of texture data to categorize items and even to create item recommendation.





### **NER with spaCy**

Selecting transcript lines in this section will navigate to timestamp in the video

- In this notebook we will demonstrate how to leave it Spacey a Python natural longer processing framework for name daily recognition or any harmful shot. The sample text we are going to use is a snippet from LinkedIn Wikipedia page. First we input Spacey for NER and Placey for visualization of the extracted entities. Then we load an English pretrained pipeline into a variable called NER pipeline. We then assess the names of the available entity labels and save them into a variable code NER pipeline labels. To understand what each label means. We can run spacey.explain using the label name we are interested in in this case. We see that the label ORJ is for companies, agencies, institutions, and so on. We then create a variable called sample text to save the text. And then we estimate the total number of sentences in the sample text is 59. Now we apply the NER pipeline to sample the text and save it into a verbal code NER text with any rate over the NER text to assess de-identified entities its label, the start and the ending spans. The first start. What we see is December, 2022 and it's been tacked correctly as a date. We can also explore all the details such as the number of occurrences of an extracted entity. For example, there are threat entities doctors money now to have a better appreciation of identities extracted, we will go ahead and view them using displacing here. We can see all the identified I entities in their collocated labels. In this exercise, we explored how to use Spacey and industrial strength, natural longer processing library for and name day recognition task. The task goes fast, efficient, and easy to use.

### **Data preprocessing for custom NER**

Selecting transcript lines in this section will navigate to timestamp in the video

- The general idea behind data annotation for named entity recognition is that the entities of interest get tagged with the desired labels in an appropriate format, depending on the training data. In this notebook, we will learn how to notate data in a format that is compatible with Spacy a natural longer processing library that uses an open source notebook compatible library called Spacy annotator. The sample text used in this video is a snippet from LinkedIn's Wikipedia page. First we install the library using pip. Then we input the necessary models. Then we save the text into a variable called sample text. We then split the sample text using the next line corrector and save into a list called sample text list. After removing the empty strings there are 15 sentences in total. Now go ahead to save the sample text list into the pd data frame called DF with a column name: text. To process our raw text. We load a phython Spacey pipeline into a variable called NLP. We then iinitiate an annotator object with our customs labels and model. Our customs labels here are: company, person, money, plays, and dates. Now we can start a data annotation by initiating annotates, for the annotator object and saved as a variable called TF labels. Now we have an interactive view with submit, skip and finish function where appropriate entities can be assigned to their labels by copying from the displayed sentence. For the first sentence of my view we have the company was founded in December, 2022. A date by Reid Hoffman, a person from PayPal and SocialNet companies. After annotation, we can view the DF labels. You will notice a new column called annotations. To understand the difference between the raw text and the annotated text. We print the data type and content of a row from the text column. We can see the data and its type, our strength and for the annotations column, the data and its type it's a Topo Denotation from Spacey training data is usually a Topo. So we are on the right path. For future use we save our data frame as a CSV file in download.

### **Custom model training with spaCy**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Now, let's learn how to train a custom Named Entity Recognition model, NER, with spaCy. To successfully walk along, some configuration files are provided in the exercise folder. This section shows how to annotate data for NER. First, we install the spaCy library using pip. Then we go ahead to input the necessary modules like spaCy, and pandas. And we verify the version of spaCy is 3.2.3. Then we load the data set into a variable called df. And we use the head() command to see the data and get the available columns. Then we apply ast.literal\_eval to the annotations column to change the data from string to tuple. We then use the info() command to see the data frame as 15 rows. In practice, it is advisable to have at least 15 instances of an entity in the training data. 15 rows of annotation will not provide that. So for the purpose of this exercise, we increase our data instances by multiplying our data frame by 100. And great. Now we have 1,500 instances. Let's go ahead and save the first one to the list called train\_data. And save the remaining 300 into another list called test\_data. We then load the blank spaCy pipeline into a variable called nlp, and create an instance of a DOCBin. And save it to a variable called db. Now we can iterate over our training data, and format it into a spaCy format, and save to disk as train.spaCy. We repeat the above step for test data, and save to disk as test.spaCy in the directory. You should have two files in your notebook, train.spaCy and test.spaCy. We can verify the data by running spaCy debug data on the default config file. We then run the command throughout the field.training configuration. You may need to open the config file generated, and point the variables to the train and test spaCy files. Great! Now we are ready to train our model. We then load the best of the model outputs into variable called test\_nlp. We can see that the available labels are company, dates, and place. We then test that train.NER model on test data, and visualize with displacy. Our custom NER model correctly predicted September, 2022 as a date, Lagos as a place, LinkedIn as a company. Due to the limitation in the training data, it's predicted Nigeria which fits as a place, as a company. And we, a pronoun, as a container as well. In this exercise, we successfully trained a custom Named Entity Recognition model using spaCy.

What is the extension used for spaCy files?

* .spacy  
  Correct
* .csv  
  Incorrect
* .pickle
* .json

## **Question 2 of 3**

When annotating data, what are entities tagged with?

* Labels  
  Correct
* Titles
* Verifier
* Focus

## **Question 3 of 3**

How can you visualize extracted entities with spaCy ?

* Displacy
* SpaCy viewer  
  Incorrect
* Plotly  
  Incorrect
* SpaCyplot  
  Incorrect

### **What is sentiment analysis?**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Narrator] Sentiment analysis is a text analysis technique that infers opinions, sentiments, emotions, and attitude from entities, such as products, services, organization and event from data and classifies them according to polarity, which I'll talk about shortly. Now, the sentiment can be explicit where the text directly expresses an opinion. For example, this dress is beautiful. So for obvious reason explicit sentiments are easier to analyze. On the other hand, a sentiment can be implicit where the opinion is implied. For example, the dress is a bargain. The statement could be categorized as an excitement or is it just a factual statement with no sentiment? Now , depending on how you use sentiment analysis, it can also be called opinion mining, subjectivity analysis, appraisal extraction, or polarity classification. Sentiment polarity is usually represented as a range of classes, such as positive, negative, and neutral, or in numerical scale, like zero to five. The machine learning used for sentiment analysis can either be supervised where there's access to input and output data representation during training or unsupervised where the only input data is accessible during training. The central principles of sentiment analysis is that opinions are essential to most human activities and shape and they shape action. The sentiment can be at the word, sentence, or document level. Typically, you first run the sentiment detection and then you run polarity classification to identify patterns and make data-driven decisions. Some of the most popular libraries and framework for sentiment analysis are N L T K **Natural Language Toolkit Vader, Valence Aware Dictionary for Sentiment Reasoning, TextBlob, Hugging Face, spaCy. All those frameworks are based on Python except for Stanford CoreNLP, which is based on Java. Now, there are many practical uses of sentiment analysis and more uses are emerging everyday.** People make decisions based on what others feel. So you can apply sentiment analysis to review how customers feel about a product. For example, you can try to predict how the stock and commodities markets will trend by measuring sentiment of news articles in market research. You can also use it to forecast the outcome of elections. On social media platform. you can use opinion mining to determine the topic of discourse. Online information is also monitored by security agencies in order to detect and identify key situation in developing event. While there are limitation in language structure for sentiment detection of implicit expression, such as sarcasm. Sentiment analysis, as a natural language process and technique, is still a valuable tool to detect patterns, gather insights and enable data-driven decision.

### **Sentiment analysis with VADER**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Let's practice how to do sentiment analysis with Valence Aware Dictionary and sEntiment Reasoner, VADER. At the end of the exercise, you will be able to calculate your own sentiment score and label them according to polarity. We'll be using the sentiment labeled sentences dataset from the open source UCI Machine Learning Repository and the Python library, VADER. First, download and unzip the dataset using the wget and unzip commands. Then combine the subsets of the dataset into a single data frame called df using pandas. Now, let's view information about the combined data frame using df.info and df itself. The output shows a total of 2,784 rows or reviews and their pre-labeled sentiment, where zero means negative and one means positive. Now, we can go ahead and generate the VADER sentiment score based on the review column. Looking at the data frame again, we can see we now have a new column named vader\_sentiment\_score, and it has a range of numbers assigned to it. With VADER, sentiment scores greater than 0.05 are positive and those with less than -0.05 are negative. Scores between -0.05 and 0.05 are neutral. So for simplicity, we'll encode the VADER sentiments scores into three distinct numerical values: zero for negative, one for positive, and two for neutral. Now we can assign text polarity as sentiment labels, positive, neutral or negative, based on the numerical encoding. In this exercise, we downloaded our data, used VADER, a rule-based sentiment analysis tool, to calculate the sentiment scores, and labeled them according to polarity.

### **Sentiment analysis with transformers**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Narrator] Now we will explore how to do sentiment analysis with transformers. The transformer is a type of neuro network architecture that captures context and sequences such as text, using a self attention mechanism. In this video, we will be using the sentiment labeled "sentences data sets". You can find how to download and pre-process the data here. First, we install transformers from the organ first Python library using PP-install. Then, then we import pipeline from transformers. The pipeline is where we specify which pre-treat transformer model we want to use. We then create a pipeline instance, and save it into a variable called "distill bed classifier". We're going to use sentiment analysis as the default transformer model, and set truncation to "true". So that sentences longer than the specified token length for the model are truncated. Let's practice, by applying the defined classifier on the second review in our data. The output shows put the sentiment label in score. The review here says good case, excellent value, and the model predicts it as a positive sentiment, and a score of 0.99. We can now go ahead and apply this to the reviews column in our data frame. This may take a few minutes to run, so be patient. So now, you can see the data frame has a new column called "distill bed sentiment". To compare performance, we repeat the above for Roberta and Beth transformer models. Just looking over the data, we see that both the distill Beth and the Roberta results were efficient as they aligned with the original sentiment label. However, the results from Beth were not as precise and had lower sentiment scores. Now, in terms of speed, distilled Beth was the fastest, followed by Bert and then Roberta model the slowest of the three. So, if I had to make a choice about which transformer model to use, I'll go with the distilled Beth model for similar sentiment analysis task.

## **Question 1 of 3**

Sentiment analysis is NOT known as \_\_\_\_\_.

* subjectivity analysis  
  Incorrect
* opinion mining
* opinion leveraging  
  Correct  
  Sentiment analysis is also known as opinion mining, subjectivity analysis, appraisal expression, extraction, and polarity classification.
* polarity classification

## **Question 2 of 3**

A transformer is \_\_\_\_\_

* an electricity regulator for text
* not useful
* a type of neural mixed network
* a type of neural network architecture  
  Correct

## **Question 3 of 3**

What does the acronym VADER stand for?

* Valence Action Dictionary and sEntiment Reasoner
* Valence Aware Dictionary and sEntiment Reasoner  
  Correct
* Valence Aware Dictionary and sEntiment Reasoning
* Valent Aware Dictionary and sEntiment Reasoner

### **What is text summarization?**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Narrator] Text summarization is a computational technique for generating brief, accurate and coherent subsets of lengthier documents. When summarizing text, one or more documents are given as an input, and a summary is obtained as an output. Text summarization typically happens in three key steps. Topic identification, topic interpretation and summary generation. Summarization can either be abstractive or extractive. The abstractive summarization technique leverages natural longer processing algorithm to understand the meaning of text. It is generally considered to be a more complex and computationally intensive of input. The results in summary is comparable to how humans read and then summarize text in their our words. Extractive summarization technique, leverages statistical and linguistic characteristics to assign importance to sentences and paragraphs. The summary with this technique consists of phrases extracted from the original document. The central principle of summarization is to reduce text size while retaining the important content. **They can be tuned to fit the need of the target audience, and their accuracy is measured by Recall-Oriented Understudy for Gisting Evaluation or ROUGE metrics. Some of the most important libraries and frameworks for summarization are pysummarization, sumy, Pyteaser, Gensim TextRank.** All these frameworks are based on Python. Now there are many practical uses of summarization and many more are emerging every day. Summarization is used in the generation of news headline, book summarization, email summaries, research abstract, question answering, study notes and flashcards, social media content generation, and summary from transcripts. As more data is generated globally the need to capture the most important part of text data increases. Understanding text summarization and how to implement its methodology is an important skill in your professional toolbox.

### **Text extraction for summarization**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Let's demonstrate how the extractive text summarization method works, using the Python library Sumy. We'll be using a snippet of the blog article by LinkedIn Learning as our data set. First, we install Sumy using pip. Then we input all the necessary modules including LexRank summarizer and LSA summarizer, which provide us with the paths in summarization algorithms. We then create a variable called sample\_text to save the text. And to confirm the number of words in the sample\_text is 606. We then pass those string using plain text summarizer with English as a tokenizer. Tokenization is the process of breaking down sentences into words. And that is why we have to specify in what language we want that to occur. Next, we instantiate the LexRank summarizer and generate the summary with a sentence count of 20. Voila! We have a summary. Note that we can increase or decrease the sentence count depending on the length of the desired summary. Now let's try the LSA summarizer on the same data. We first instantiate the algorithm, and then run it with a sentence count of 20 as well. There you go. A new summary. Notice that some of the sentences are exactly the same as the ones in the first summary, but there are some new ones. Great! Now that we know our loading data from plain text works, let's explore loading data from a URL. To start, you specify a URL, and use the html\_parser to import the text element from the site, using the English tokenizer. We then generate our summary using the Lex summarizer. And then we run it again, but now using the LSA summarizer. Now, if you compared the output of both summarizers, you might notice some difference in performance. But glancing over the results, it is clear that both summaries could be useful. Therefore, making a decision on which algorithm to use, is determined by wide range of factors. But for sure, domain expertise is the greatest contributing factor.

### **Text summarization with sumy**

Selecting transcript lines in this section will navigate to timestamp in the video

- Now let's demonstrate the abstractive text summarization method, using pre-trained transformers from Huggingface. Transformers are a type of deep planning models that leverages attention mechanism when learning sequences like text. Will be using the same data set from the previous video, the snippet from the blog article by LinkedIn learning. First, we install transformers using PIP and we input pipeline from transformers that's what we will actually use as a pipeline for all that we want to do. Then we save a text into a variable called sample text and verify that the total number of characters 3,654. And the number of words is 606. With an instant sheet the summarization pipeline by simply giving it a name, as you can say from the outputs it's default to a distilled bat model. Now, let's say the performance of this model on our sample text, we have a minimal full summary of the simple text in about six lines. Now, let's try another transformer called Pegasus, this time when instant sharing, the summarization pipeline we specify the model, the tokenizer and the private framework or PT. Now let's see the performance of this summary. This model summarized the sample text in four lines and made a great attempted personalization, which is similar to how humans summarize. Great. So now let's repeat the above steps four bats transformer, but this time using the tense of flu framework, printed the output the bat model summarized the sample text into about four lines. In this video, we explored using transformers for abstractive text summarization and we saw that out of the three transformer models used for login phase. Pegasus seemed to be the most promising. Notes that we used to pre-train models to optimize their performances. They can be tweaked to use your data in your specific use cases.

You answered 3 of 3 questions correctly.

Continue watching Retake quiz

## **Question 1 of 3**

You can use language models for summarization tasks.

* FALSE
* TRUE  
  Correct

## **Question 2 of 3**

Extractive summarization returns exact phrases from the original text.

* TRUE  
  Correct
* FALSE

## **Question 3 of 3**

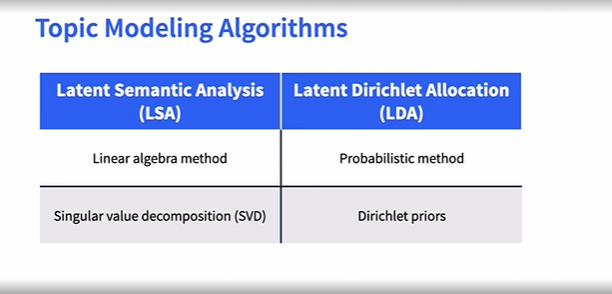
Which of the following is not a step in the text summarization?

* summary generation
* topic arrangement  
  Correct  
  Summarization happens in 3 key steps. 1) topic identification 2) topic interpretation 3) summary generation
* topic identification
* topic interpretation  
  Incorrect

### **Introduction to topic modeling**

Selecting transcript lines in this section will navigate to timestamp in the video

- Topic modeling is one of the most important and useful techniques used when analyzing large text data sets. It is an unsupervised machine learning technique that automatically finds patterns. And in first topics within text based data, the findings are usually context aware. In other words, the categories of text elements or documents into classes is based not only on text **similarity but also on semantic similarity. F**or example, documents containing words like health, doctor, patient, hospital, will be categorized under a topic called healthcare. And for a topic called famine, you might see words like farm, crops, corn, and wheat. The assumption of topic modeling is that every document comprises a statistical distribution of topics that can be obtained by combining all the distributions for all the topics covered. In other words, the algorithms try to figure out which topics are present in the data set and how strong that presence is. In this course, we are going to focus on two main topic modeling algorithms: LSA for latent semantic analysis and LDA for latent dirichlet allocation. LSA is one of the foundational techniques used for component analysis applied to text data. It is a well-known linear algebra method called singular value decomposition, or SVD. On the other hand, LDA uses a probabilistic method using dirichlet powers, making it less prone to over-fitting. In general, the best practice is to try several different algorithms and see which one works best with your data set. Topic modeling is used in several application, such as gathering insights to improve your product from customer reviews and customer support emails, who cluster millions and millions of news articles and published papers to find common patterns. Topic modeling is used in recommendation engine or information retriever systems to find similar products, topics, or articles related to what you're looking for, making for a better user experience. Another application of topic modeling is for text annotation. One of the most tedious tasks one can do manually, it automatically categorizes your data set into classes, making it easier to collect insight and analyze your information. These are just a few examples of what this powerful technique of topic modeling can do for you and your large text-based data sets.



### **Data preprocessing for topic modeling**

Selecting transcript lines in this section will navigate to timestamp in the video

- Now, let's see how all these work in practice. We will be using the 20 news script data set, from the open source UCI Machine Learning Repository, in this notebook. And to get a clean version of this data, we are going to use Psyche plan to input. As you can see here in the first cell we start by importing the required Python libraries escalan dot data sets, pay, print, and pandas. We then save the data from Fetch 20 news group into a variable called "Data Sets" on the function call. We specified that the subset of the data retrieved should be whole. We set the random state to 32 in order to ensure consistency. And finally, we specify that we want to remove headers, photos, and quotes, that are present in the original dataset. Once we get the data, we can print out the keys using dataset dot keys function. The results is a dictionary key array with data, file names, target names, and description. Now let's get the value for the data key with data sets, data command, and then print in the data set's targets name using pay print. You can also check the length of the data using the lens command. In the same fashion you can get the shape of the target and the file names. We can now save the data sets into an appendix data frame called "Data Sets DF", renaming the data to "news" and the target to "label," which we can see its view here by simply calling the variable data sets "DF." Now, if you want more context about the label you can apply a simple lamb function to map the numeric label to the string label name. And there we go. The data set data frame now adds three columns, news, label, and label name. Great. So up to now, we have inputted the data set, transferred it to a data frame for ease of manipulation, and renamed the columns appropriately. Now let's pre process the data so it can be used for topic modeling using the Jensen Python library. We first update and import the appropriate functions from the Jensen Library and use the help command to see that we stop pre processing filters available to be used on our text data. As an example, we create the clean news column to show what the pre processed strength function does. So now let's pre process strengths, but this time we are going to set the filters to convert the text to lowercase and also apply the strip tag, strip punctuation, strip numeric, remove stop wood, strip shot filters and then put the outputs in a column called "clean use one" applying the filters to the dataset "DF news column." To see the results we called the dataset "DF" and see the new column. Now the Jensen Library requires that the input data is a dictionary object. So we transform the data into one, install it in the dataset dictionary variable. I usually like to check that the operation went well by checking the length of the dictionary. Great. It's not zero. So we're good to go! Looking to ID a map of text talking to its numerical representation is assessable by the dictionary dot talking to ID function. We then vectorized the data sets dictionary using the bag-of-words or ductible method. Remember: the topic modeling algorithms accept vector representation, not text! We check the results using the length method and print the results to see an instance of the vectorized text. Now let's explore another vectorization technique called TF IDF; temp, frequency, inverse document frequency. TF IDF is vectorized by getting the bag of words vectors into a TF IDF model function. As shown here, we can then check the length of the TF IDF output and inspect an instance of the vectorized text. Notice that the second value isn't decimals and not all numeric like in the bag-of-words instance above. We have learned how to download and pre process our data. Now we are ready for the next step in applying topic modeling to add data.

### **Topic modeling with Gensim**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Now, let's see how to apply topic modeling algorithms to a pre-processed data set. We'll apply the Latent Dirichlet Allocation, LDA, and the Latent Semantic Analysis, LSA, algorithms from the Gensim Python library to the train and use script data set that has been pre-processed, using two methods of vectorization. To start, we import the LDA model from the gensim models module, and then create the variable called "LDA bow" to hold the LDA model (indistinct) with hyper-parameters. For the bag of words corpus, the number of topic and identity to word mapping and the random states of zero to ensure consistency. Now, we'll print the possible topic classes that were inferred by the Lda algorithm. The predictions will include up to 20 topics and about 8 words as we specified. Now, if I take a look at the output, we'll see that the model did a good job. This topic contains words like space, NASA, and shuttle, for example. Take some time to observe the data and see the other topics that follow the same pattern of grouping related words together. Now, let's apply the Lda topic modeling algorithm to the data set that was vectorized using the Term frequency-inverse document frequency or Tf-idf. As you can see, the topics predicted by the Lda model on the Tf-idf vectorized data doesn't seem to be as in-sync as the previous application on the bag of word vectors. This makes a case for always varying hyper-parameters such as number of topics, the data vectorization method, and the algorithm used. To emphasize this point, let's apply the latent semantic analysis, Lsa, or latent semantic indexing, Lsi algorithm to the same data set. We start off by inputting the Lsi model from the Gensim models module, then we create a variable called "Lsi bow" to hold an Lsi model (indistinct) with the parameters that you can see here. And then we'll print the topics to see if the predicted words are related at all. For example, this topic grouped jpeg, file, edu, image together. Great. So now, let's apply the Lsi model to the Tf-idf vectorized data to see how it performs. So you can see, that the topic groupings are different than the ones above. For example, words like Armenian, Israel, Turkish were grouped together. Sure in there could be a relationship with countries or nationalities. In this case, on eyeballing, I'll say the Lda model on the bag of words data performed the best, but this will vary depending on your data set, so always explore all possibilities and variation. So to review, we've applied two of the most popular algorithms used in topic modelings leveraging the Gensim python library. And so our words are grouped differently within the topics depending on the text vectorization methods used to pre-process the data that the algorithm used to group the words into topic.

### **Topic modeling visualization with pyLDAvis**

Selecting transcript lines in this section will navigate to timestamp in the video

- Visualization is an important tool that enables you to gain insight from your data. For topic modeling, the AVI Python library enables us to do that easily. We did. We can view, interact and make decisions about our topic models. Now, you might have guessed by the name that this library uses the Latent Dirichlet Allocation, LDA Algorithm. Two steps will install the pyLDAVIS Python library using PIP. Then we import the pyLDAVIS library and the Gensim models to be able to interact with the Jensen library. We also have to enable the notebook, in order for the visualization to show within a code notebook. Okay, now let's create the variable called "vis\_bov", that contains the LDA model, the dataset coppers, and the dictionary that has been preprocessed using the bag-of-word vectorization on the 20 new script dataset. Now, when we run the visible variable, we can see an interactive visualization within our code notebook. The bubbles on the left represents the different topics identified in our close or interconnected data. The bigger the bubble there is, the more words are associated with the particular topic. We can also see how close the topics are. For example, it is obvious that topic three is far from topic 17, and there is an intersection between topic three and topic ten, which may be a prompt to reduce the number of topics defined in the LDA model. On the right side of the visualization, there is a Lambda slider on the top, and the blue and red resulted bar below the Lambda slider. Adjust the relevance matrix by ordering the word frequency bar, the consensus estats 0.6 is the optimal value for Lambda. The blue bars represent the word frequency in the entire document and the red bar represents the word frequency in the selected topic. To learn more about our data, we'll apply the same visualization to the 20 new script dataset that was vectorized using TF-IDF. The frequency in this document frequency. Install the value in a variable called "visTFIDF". The visualization parameters are the same as the ones above, but we can see a distinct difference in the distribution of the topic. For example, topics two and five were very close in the back of our visualization, while they are distinct and far from each other in TFIDF visualization, even though it is the same dataset in algorithm. In this model, we explored the application of the pylDAVIS library to visualize topic models with interactive graphs that helps us to gain valuable insight into our data.

### **Model evaluation for topic modeling**

Selecting transcript lines in this section will navigate to timestamp in the video

- Model evaluation is one of the most important steps of any analytics task. Topic coherence is the quantitative measure of the quality of topics. **We're going to implement the u\_mass Coherence measure, which is usually set between -14 and 14. And then, the c\_v measure which is usually set between zero and one. The higher the score, the higher the Coherence of the topic models. To start, we import the Coherence model from gensin.models.** Then we run the Coherence and see the human Coherence score using the Latent Dirichlet model on the bag of word vectors of the 20 used data sets. The output is -2.61. Now, when we run it using the Latent Semantic Index, the score of the same data and the same vector is -4.87. Now, we can fill out the rows with empty list and save the results into a variable called texts. If we use the c\_v Coherence parameter, the score for the LDA on bag of word vectors is 0.50. While using the LSI score on the same data in vector is 0.45. For this use case, the data vectors created with the bag of word model and analyzed using LDA, had better Coherence scores, for both u\_mass and c\_v. Go ahead and try the same analysis with the data vectorized using TF-IDF.

## **Question 1 of 5**

Latent Semantic Index topic models can be visualized using pyLDAvis.

* TRUE  
  Incorrect
* FALSE



Replay

Review this video

Topic modeling visualization with pyLDAvis

3m 35s

## **Question 2 of 5**

What does ‘Lda’ in LdaModel mean?

* Latent diagonal allocation
* Latent dirichlet allocation  
  Correct
* Latent direct action
* Latent direction allocation

## **Question 3 of 5**

Topic coherence is a quantitative measure of the quality of topics.

* TRUE  
  Correct
* FALSE

## **Question 4 of 5**

How do you import the twenty newsgroups dataset from sklearn.datasets?

* from sklearn.dataset import fetch\_20newsgroups
* from sklearner.datasets import fetch\_20newsgroups  
  Incorrect
* from sklearn.datasets import fetch\_10newsgroups
* from sklearn.datasets import fetch\_20newsgroups  
  Correct

## **Question 5 of 5**

Topic modelling is \_\_\_\_\_.

* an unsupervised machine learning technique  
  Correct
* a mixed model machine learning technique
* a supervised machine learning technique
* an uncommon machine learning technique

### **Next steps**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] We've been on the learning journey. We've explored natural language processing tasks such as sentiment analysis, named entity recognition, topic modeling and summarization together. Now, I will encourage you to study the principles and frameworks further, attempt to replicate what you've learned, and apply to your own data, and practice until you've achieved mastery. I hope this learning experience has been interesting. Feel free to connect with me via LinkedIn or ask you questions about the course below in the Q and A section.