**CHATBOT USING PYTORCH AND NLP**

**Workflow**:

1. Theory
2. Create training data
3. Build a pytorch model and training
4. Implement chat
5. Build GUI

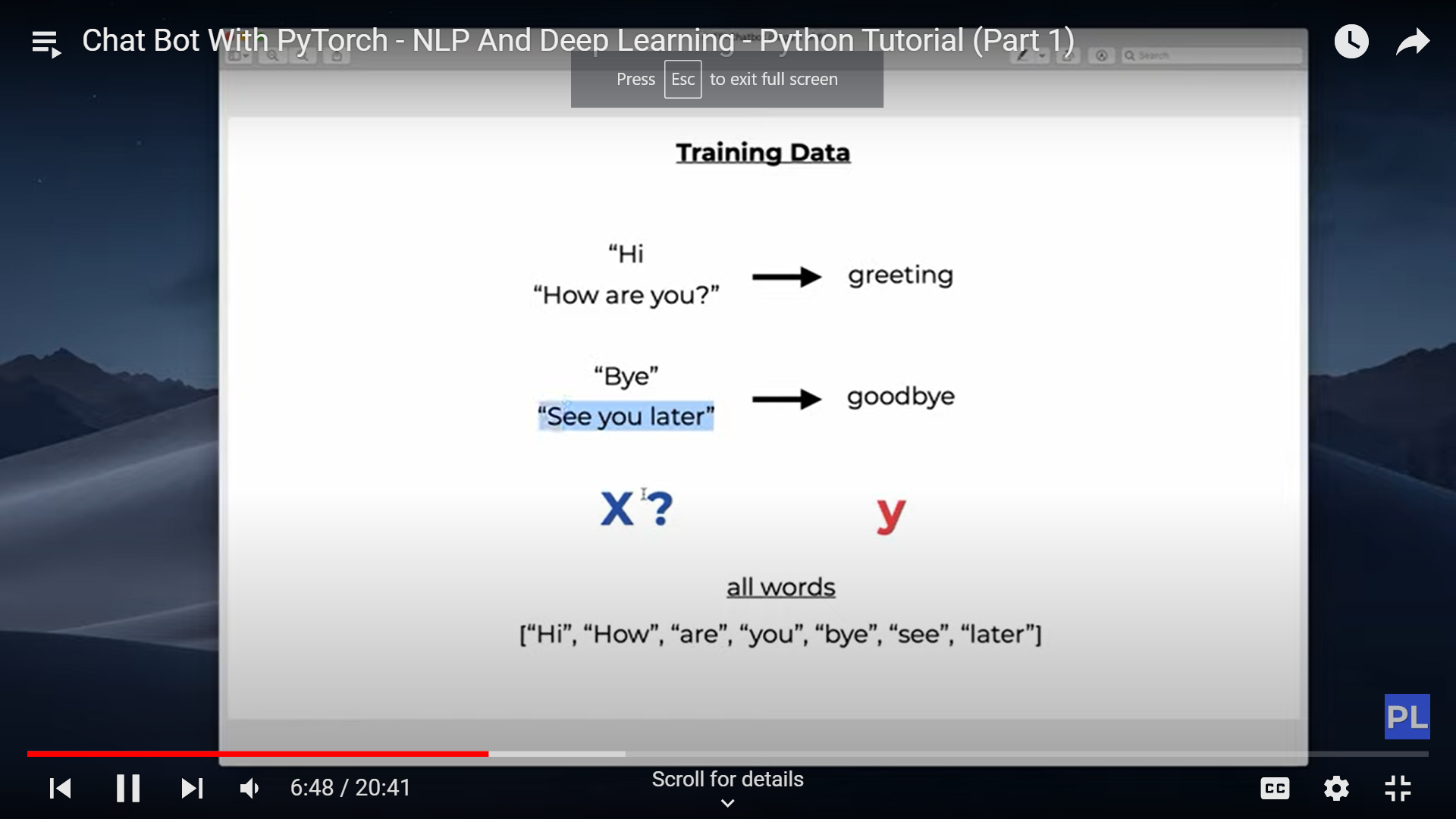
**References:**

<https://chatbotsmagazine.com/contextual-chat-bots-with-tensorflow-4391749d0077>

**Theory:**

**Intents.json explanation:**

In the json file we have certain fields such as a tag, pattern and response. The tag field is like a category for the question that is asked to the bot. pattern is like a base for different kinds of questions asked to the model. Responses are sample answers to those questions.



**Bag of words:**

Whenever we apply any algorithm in NLP, it works on numbers. We cannot directly feed our text into that algorithm. Hence, Bag of Words model is used to preprocess the text by converting it into a *bag of words*, which keeps a count of the total occurrences of most frequently used words.

This model can be visualized using a table, which contains the count of words corresponding to the word itself.

For example, consider the following two sentences:

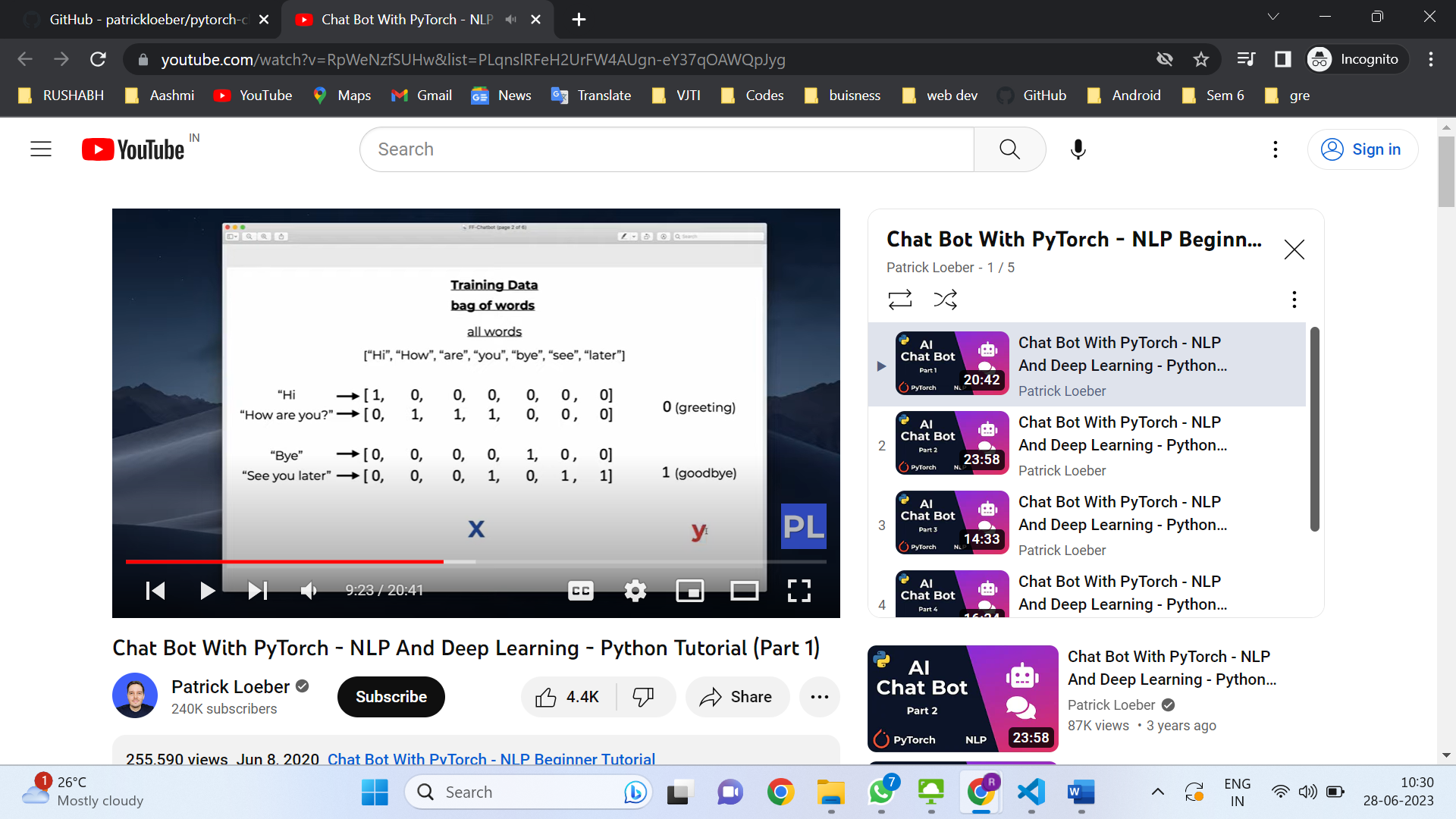
* Sentence 1: "I love to eat ice cream."
* Sentence 2: "I prefer chocolate over vanilla."

The vocabulary in this case would consist of the following unique words: ["I", "love", "to", "eat", "ice", "cream", "prefer", "chocolate", "over", "vanilla"].

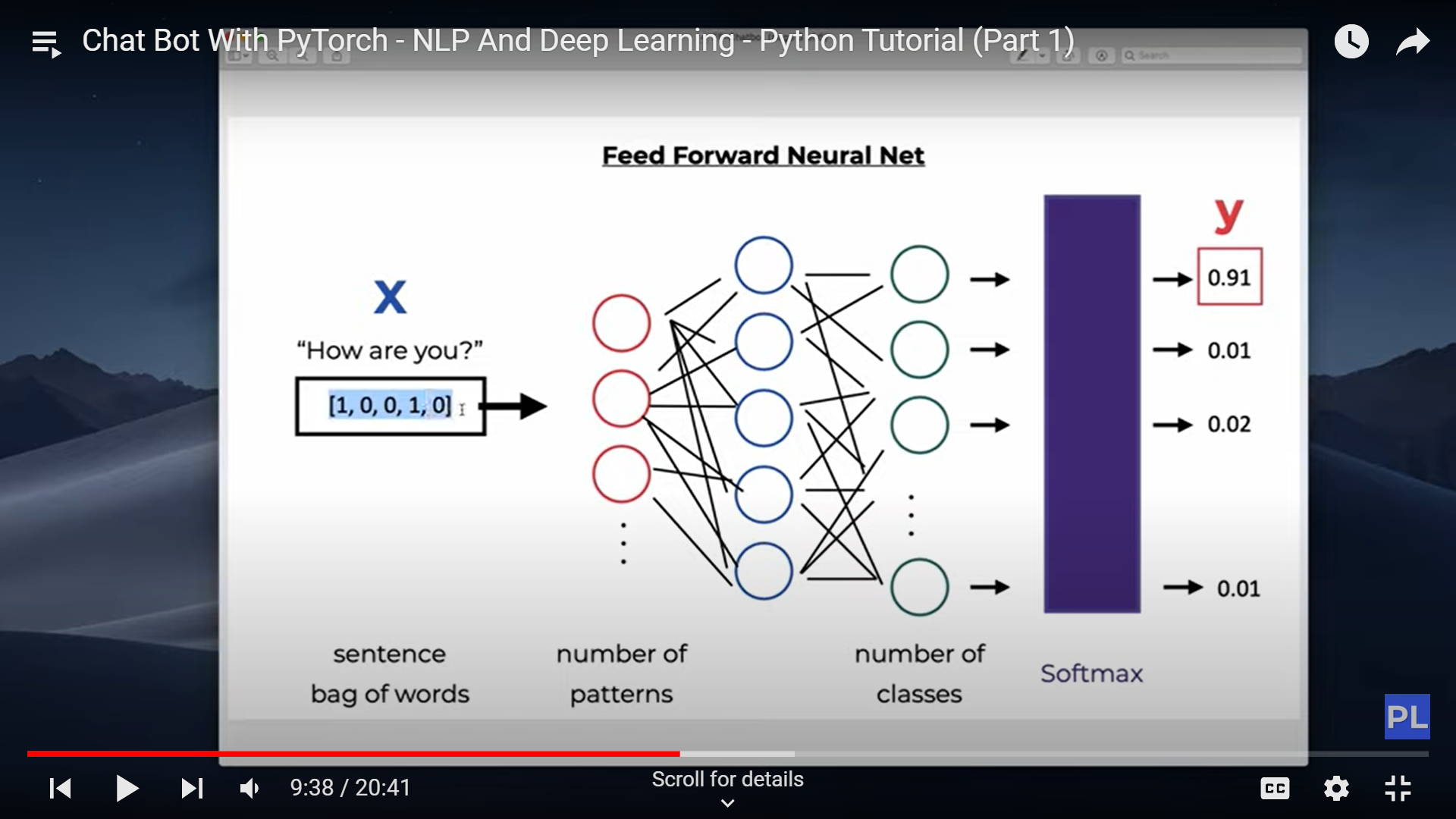
The vector representations of the sentences in the bag of words model would be:

* Sentence 1: [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
* Sentence 2: [1, 0, 0, 0, 0, 0, 1, 1, 1, 1]

LIMITATION: The bag of words completely ignores the order f the words in a sentence.



**Model Architecture:**



**Tokenization:**

Tokenization is the splitting of a sentence into meaningful units such as words, punctuations, symbols etc.

Example:

“Hi, how are you doing today?”

[“Hi”, “,” ,”how”, “are”, “you”, “doing”, “today”, “?”]

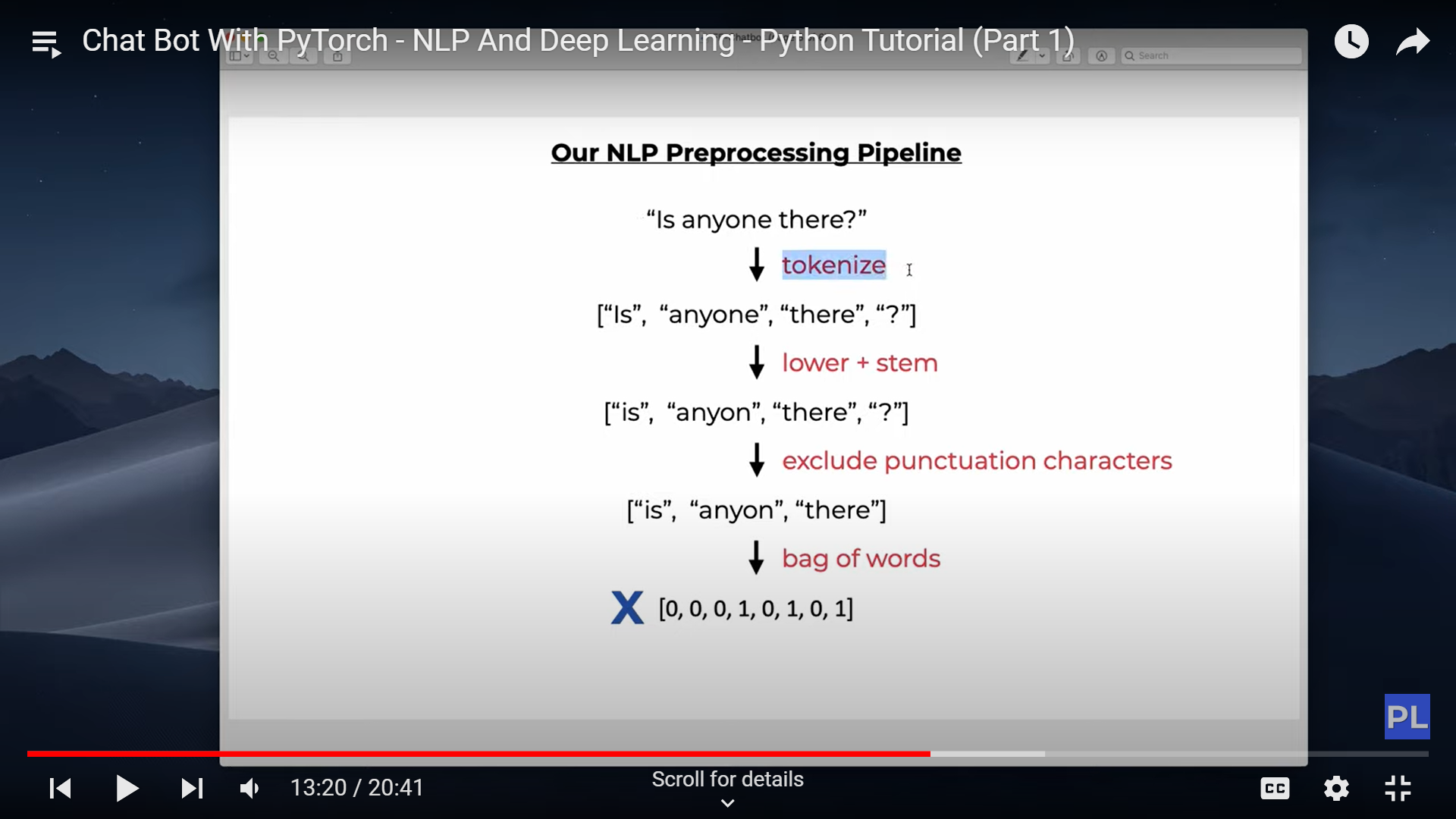
**Stemming:**

Stemming is used to find out the root form of the words. Here, it is a crude heuristic that chops off the end of words.

Example:

[‘organize’, ‘organizes’, ‘organizing’] 🡺 organ

NLP Pipeline used here:



**Why do we use batch training:**

In PyTorch, batch training refers to the practice of training a neural network model using mini-batches of input data rather than processing individual samples one at a time. Instead of updating the model's parameters after each sample, the model is updated after processing a group of samples, known as a batch.

Batch training is commonly used for several reasons:

1. Improved computational efficiency: By processing multiple samples in parallel, batch training takes advantage of hardware acceleration and optimized matrix operations provided by modern GPUs. This allows for faster training compared to processing samples sequentially.
2. Stable and faster convergence: Batch training tends to provide more stable updates to the model's parameters. The gradient updates are averaged across the samples in the batch, reducing the impact of individual noisy or outlier samples. This stability can lead to faster convergence and more reliable training.
3. Regularization effects: When using techniques like dropout or batch normalization, the effects of these regularization methods are better utilized when applied to mini-batches. Dropout, for example, randomly masks out neurons during training, and its effectiveness increases with larger batch sizes.
4. Memory efficiency: In some cases, the entire dataset may not fit into memory. By using batch training, only a small subset of the data needs to be loaded into memory at a time, making it feasible to train on larger datasets.
5. Exploiting vectorized operations: Modern deep learning frameworks, such as PyTorch, are optimized to perform matrix operations efficiently. When training with batches, these frameworks can take advantage of vectorized operations, where computations are parallelized across multiple samples within a batch, leading to improved computational efficiency.

**SOFTMAX ACTIVATION FUNCTION:**

*The term softmax is used because this activation function represents a smooth version of the winner-takes-all activation model in which the unit with the largest input has output +1 while all other units have output 0.*

Example:

i/p: [1,3,2]

o/p: [0,1,0]

**RELU ACTIVATION FUNCTION:**

The ReLU function is another non-linear activation function that has gained popularity in the deep learning domain. ReLU stands for Rectified Linear Unit. The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time.

This means that the neurons will only be deactivated if the output of the linear transformation is less than 0. The plot below will help you understand this better-

f(x)=max(0,x)

WORKING OF NeuralNet model:

1. **self.l1** represents the first linear layer in the neural network model. It is an instance of the **nn.Linear** class, which performs a linear transformation on the input.
2. **self.l1(x)** applies the linear transformation to the input **x**. The linear transformation involves multiplying the input by a weight matrix and adding a bias term. The weight matrix and bias terms are automatically initialized when creating the **nn.Linear** layer.
3. The output of **self.l1(x)** will be a tensor that represents the result of the linear transformation. The shape of this tensor will depend on the parameters **input\_size** and **hidden\_size** passed during the initialization of the **NeuralNet** class. Specifically, the output tensor will have a shape of **(batch\_size, hidden\_size)**.
4. The output tensor from **self.l1(x)** will then be passed through the activation function **self.relu** (ReLU, in this case). The ReLU activation function applies an element-wise non-linearity, setting negative values to zero and keeping non-negative values unchanged. This introduces non-linearity into the network, allowing it to learn more complex patterns and relationships in the data.
5. The output of **self.relu** will be the input to the next layer, **self.l2**, following the same process of linear transformation, activation function, and so on.