**Training a Named Entity Recognition Model with BeautifulSoup and spaCy**

**Introduction:**

In this presentation, we embark on a journey into the realm of Named Entity Recognition (NER) and explore its profound applications in the real world. The primary goal of this presentation is to elucidate the process of training a custom NER model using the dynamic duo of BeautifulSoup and spaCy.

**Agenda:**

* In the coming sections, we will delve into the following topics:
* The Significance of NER
* Leveraging BeautifulSoup for Data Extraction
* Crafting Clean Data: Preprocessing Techniques
* Unveiling spaCy: A Powerful NLP Library
* NER Model Training in Action
* Reflecting on Model Performance
* Post-Training Refinements
* Demonstrations and Real-World Application

**Data Collection with BeautifulSoup (bs4):**

Our journey commences with data collection, where we extract invaluable information from the web. BeautifulSoup, a Python library, serves as our trusted companion for parsing HTML content from web pages. It excels in navigating the labyrinthine structures of web data, providing us access to the treasures hidden within.

The web scraping process revolves around retrieving data from a designated URL. The HTML structure of the target website is dissected, and we meticulously extract data, guided by the principles of BeautifulSoup. The data obtained is a goldmine of textual content.

**Data Preprocessing:**

The importance of data cleanliness cannot be overstated. Data preprocessing is the crucible where raw data is transformed into a refined dataset suitable for NER model training.

Our preprocessing journey entails techniques like HTML tag removal, whitespace stripping, and text standardization through lowercase conversion. These processes pave the way for consistent and intelligible text data, preparing it for the rigors of NER training.

**Named Entity Recognition with spaCy:**

spaCy, a versatile and robust NLP library, is our instrument of choice for NER model training. We entrust spaCy with the task of identifying and classifying entities within our cleaned text data.

The training process hinges on providing annotated data to spaCy, specifying entity boundaries, and labelling them as 'PRODUCT' in our case. The language model adapts and learns to recognize these entities within the text.

**Model Training and Evaluation:**

With our training data prepared, the model is primed for learning. Training a custom NER model is a meticulous process, involving iterative updates and adjustments.

Data is split into training and testing sets, and the model's performance is assessed using standard evaluation metrics, including precision, recall, and F1-score. The journey is marked by challenges and discoveries as we fine-tune the model.

**Post-Training Refinements:**

The journey does not conclude with model training. Post-training refinements allow us to address any limitations or errors that may have arisen during the process. It is a process of continuous improvement and adaptation.

**Demonstrations and Real-World Applications:**

To conclude our journey, we present live demonstrations showcasing the NER model's prowess in recognizing named entities within actual text data. We bridge theory and practical application, underscoring the versatility and utility of NER models in various domains.

In this presentation, we have navigated the intricate process of training a custom NER model, from web data extraction to model evaluation and refinement. This journey exemplifies the fusion of Beautiful Soup’s web scraping capabilities and spaCy's NLP prowess, creating an asset in the world of text analysis.

**URL Accessibility Check:**

Before we embark on the Named Entity Recognition (NER) model testing, it's essential to ensure the validity of the URLs from which we plan to extract data.

A quick pre-processing step involves checking whether the URLs are accessible, saving valuable time and resources by excluding invalid or unreachable links.

**Code for URL Accessibility Check:**

We implement asynchronous techniques to efficiently verify the status of multiple URLs. This allows us to assess accessibility while minimizing waiting times.

The *is\_url\_accessible* function utilizes the aiohttp library to send GET requests to each URL, with the ability to retry a specified number of times in case of initial failures.

Additionally, we simulate the behaviour of a web browser by randomizing user-agent strings, ensuring a more comprehensive test of URL accessibility.

**Results and Benefits:**

The outcome of this check is a list of URLs categorized as either accessible (True) or inaccessible (False).

This process offers a valuable prelude to the subsequent stages of web scraping and NER model testing, reducing potential disruptions and enhancing the overall efficiency of our project.

**Model Testing with Real Data:**

A critical phase of our NER model development is testing its efficacy on real-world data. The model's ability to recognize named entities in actual text is a crucial metric of its performance.

**Code for Model Testing:**

* We load our custom NER model, which has been meticulously trained and refined, into memory using spaCy.
* A list of URLs is obtained from a pre-generated file.
* We make requests to each URL to retrieve HTML content. If the response status is 200 (OK), we parse the content using BeautifulSoup.
* The NER model is applied to the parsed text, and any recognized entities are extracted.
* In this specific use case, we focus on extracting product names as named entities from the web content.

**Results and Output:**

The output of this testing phase is a list of URLs paired with the product names recognized by our NER model.

This practical demonstration highlights the model's capability to identify and categorize named entities in web content, such as product names.

**Statistics and Model Performance:**

After rigorous testing, let's dive into the statistics that reveal the performance of our NER model on real-world data.

To visually represent these statistics, we present two pie charts that conveys the distribution of URL accessibility and product name extraction accuracy.

**URL Accessibility:**

We initially assessed the accessibility of a total of 705 URLs:

* Bad URLs (inaccessible): 388 (55.04%)
* Good URLs (accessible): 317 (44.96%)

**Product Name Extraction:**

We then applied the NER model to 279 secure URLs:

* Legit Products Extracted (correct): 195 (69.89%)
* Not Legit Products Extracted (incorrect): 83 (29.75%)
  + Blank Product Names (incorrect due to empty results): 15 (5.38%)
  + Wrong Product Names (incorrect due to inaccuracies): 33 (11.68%)