22.2 Lesson Summary - Big Data in the Cloud

Big Data often contains a large amount of text in various forms. Extracting meaning and patterns from this data is an essential part of analysis and there are a number of tools to assist in Natural Language Processing.

Concept: **Natural Language Processing** (**NLP**) is used to extract meaning or patterns from text data. This practice is often used with Big Data to search a wide range of documents or summarize their contents. There are a number of steps in NLP including **Tokenization**, Stopwords Filtering, TF-IDF, and Machine Learning. The first step, tokenization, involves breaking the text up into a sequence of words, phrases, and/or sentences to focus on the most relevant content. To tokenize the column of a Spark DataFrame you could use the following code:

*from pyspark.ml.feature import Tokenizer*

*tokenizer = Tokenizer(inputCol="column\_to\_tokenize", outputCol="words")*

*tokenized = tokenizer.transform(dataframe)*

*tokenized.show(truncate=False)*

* Activity: 01-Ins\_Pyspark\_NLP\_Tokens

Concept: **User-Defined Functions** (**UDF**s) improves Big Data analysis by enabling the calculation of custom values in a DataFrame. To add the number of tokens in a column to a DataFrame you could use the following code:

*from pyspark.ml.feature import Tokenizer*

*from pyspark.sql.functions import col, udf*

*from pyspark.sql.types import IntegerType*

*tokenizer = Tokenizer(inputCol=" Input Column ", outputCol="Output Column")*

*def word\_list\_length(word\_list):*

*return len(word\_list)*

*count\_tokens = udf(word\_list\_length, IntegerType())*

*tokenized = tokenizer.transform(dataframe)*

*tokenized.select("Input Column ", "Output Column")\*

*.withColumn("Token Count", count\_tokens(col("Output Column"))).show(truncate=False)*

* Activity: 03-Stu\_Pyspark\_NLP\_Tokens

Concept: **Stop Words** are words that are necessary but don't add any meaning to text like *this, to, the, a, there,* or *an*. These words should be removed for NLP. To remove stop words from a column in a Spark DataFrame you can use the following code:

*from pyspark.ml.feature import StopWordsRemover*

*remover = StopWordsRemover(inputCol="column to target", outputCol="output column")*

*remover.transform(data\_frame).show(truncate=False)*

* Activity: 04-Ins\_Pyspark\_NLP\_Stopwords, 05-Stu\_Pyspark\_NLP\_Stopwords

Concept: In NLP, meaning can be extracted from text by focusing on **Term Frequency** (**TF**) and **Inverse Document Frequency** (**IDF**). Term Frequency is the count of words in a document. Inverse Document Frequency indicates how important a word is compared to all the words used. IDF is the log of the division of the total number of documents by the number of documents containing the target word. The more documents a word is in the lower it's IDF. The TF-IDF is the product of the TF and IDF. A high TF-IDF indicates a high frequency in the current document but not in the rest of the documents. Because this analysis focuses on the frequency of words but not their sequence this is referred to as a "bag of words" approach. To calculate the TF and IDF for your data you can use the following code:

*from pyspark.ml.feature import HashingTF, IDF, Tokenizer*

*tokenizer = Tokenizer(inputCol="Input Column", outputCol="tokens")*

*tokenized\_df = tokenizer.transform(dataframe)*

*hashing = HashingTF(inputCol="tokens", outputCol="hashedValues", numFeatures=pow(2,4))*

*hashed\_df = hashing.transform(tokenized\_df)*

*idf = IDF(inputCol="hashedValues", outputCol="features")*

*idfModel = idf.fit(hashed\_df)*

*rescaledData = idfModel.transform(hashed\_df)*

*rescaledData.select("tokens", "features").show(truncate=False)*

* Activity: 06-Ins\_Pyspark\_NLP\_HashingTF, 07-Stu\_Pyspark\_NLP\_HashingTF