**1. StopWordsRemover**

Removes unnecessary words, e.g. articles and to-be forms.

Input should be a bag of words

>>> rdd = sc.textFile("data.txt")

>>> from pyspark.ml.feature import StopWordsRemover

>>> rdd = sc.textFile("data.txt")

>>> rdd2 = rdd.map(lambda x: x.split(" "))

>>> from pyspark.sql import Row

>>> rdd3 = rdd2.map(lambda x: Row(x))

>>> df = spark.createDataFrame(rdd3, ["words"])

>>> remover = StopWordsRemover(inputCol="words", outputCol="filtered")

>>> remover.transform(df).show(truncate=False).show()

**2. N-gram**

Gives a pair of n-consecutive words:

E.g. 2-gram of sentence: “Hi my name is John”

[“Hi my”, “my name”, “name is”, “is John”]

>>> rdd = sc.textFile("data1.txt")

>>> rdd2 = rdd.map(lambda x: x.split(" "))

>>> rdd3 = rdd2.map(lambda x: Row(x))

>>> df = spark.createDataFrame(rdd3, ["words"])

>>> from pyspark.ml.feature import NGram

>>> ngram = NGram(n=2, inputCol="words", outputCol="ngrams")

>>> ngramDataFrame = ngram.transform(df)

>>> ngramDataFrame.select("ngrams").show(truncate=False)

**3. Binarizer**

Transform a column into binary values. A value below threshold=0 and above threshold=1

>>> from pyspark.ml.feature import Binarizer

>>> continuousDataFrame = spark.createDataFrame([(0, 0.1), (1, 0.8), (2, 0.2)], ["id", "feature"])

>>> binarizer = Binarizer(threshold=0.5, inputCol="feature", outputCol="binarized\_feature")

>>> binarizedDataFrame = binarizer.transform(continuousDataFrame)

>>> binarizedDataFrame.show()

**4. Normalizer**

Normalize the values of a vector to make it a unit vector

>>> from pyspark.ml.feature import Normalizer

>>> from pyspark.ml.linalg import Vectors

>>> dataFrame = spark.createDataFrame([(0, Vectors.dense([4, 1, 2, 2]),), (1, Vectors.dense([1, 3, 9, 3]),), (2, Vectors.dense([5, 7, 5, 1]),)], ["id", "features"])

>>> normalizer = Normalizer(inputCol="features", outputCol="normFeatures", p=1.0)

>>> normData = normalizer.transform(dataFrame)

>>> normData.show()

**5. SQLTransformer**

Apply a SQL on features to generate other feature columns

>>> from pyspark.ml.feature import SQLTransformer

>>> df = spark.createDataFrame([(0, 1.0, 3.0), (2, 2.0, 5.0)], ["id", "v1", "v2"])

>>> sqlTrans = SQLTransformer(statement="SELECT \*, (v1 + v2) AS v3, (v1 \* v2) AS v4 FROM \_\_THIS\_\_")

>>> sqlTrans.transform(df).show()