Big Scale ML Frameworks

Good ones

Open Source

Manifold contributors

Well documented

State of the art algorithms

Performance

What about these ones ?

Microsoft CNTK – distributed deep learning framework ; open source but low amount of contributors

Caffe(2) – caffe2 has been integrated in PyTorch

MapReduce – we (Criteo) used to train some models on MapReduce some time ago ; implied mapreduce boilerplate ; sharing information between executors was a pain

Scikit-learn – out of core algorithms for single machine learning ; no distributed algorithms

h2O.ai – proposes automatic feature engineering as well as user interfaces, that can interface with Spark, TensorFlow, etc… Open source ; proposes a platform (with fee)

MOA – data stream mining ; only supports java ; does not seem prod-grade (no integration with kafka…)

Shogun – machine learning library ; no out-of-core algorithm ; not meant to be used in distributed environment

Baidu AllReduce / Horovod – nope – will talk about it in course 3;4

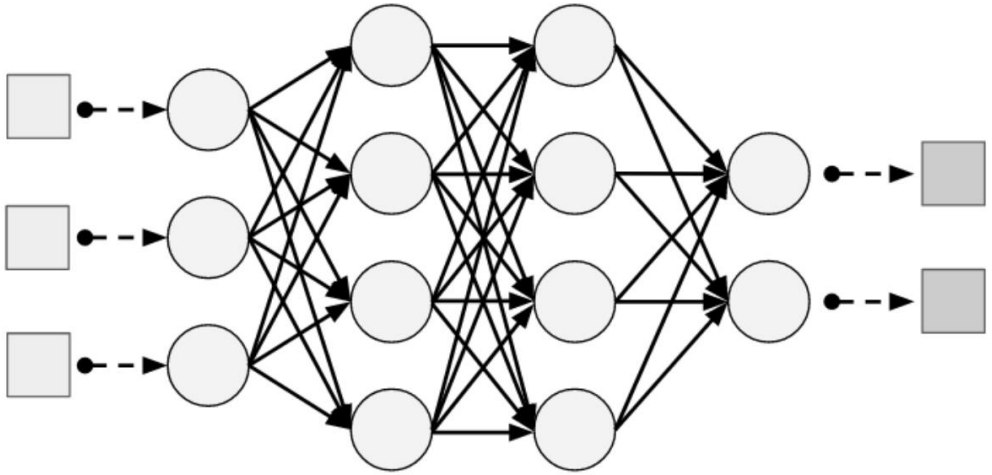
Theano – library to evaluate mathematical expressions on vectors

Breeze, BLAS, LAPACK – efficient linear algebra libraries

Tensorflow

Model is a graph of parameterized transformations

Joint of optimization of all parameters



Automatic differentiation (vs numeric differentiation)

➔ Need to use tensorflow functions, code is data !

Some features

Parallel, Distributed, Accelerator support

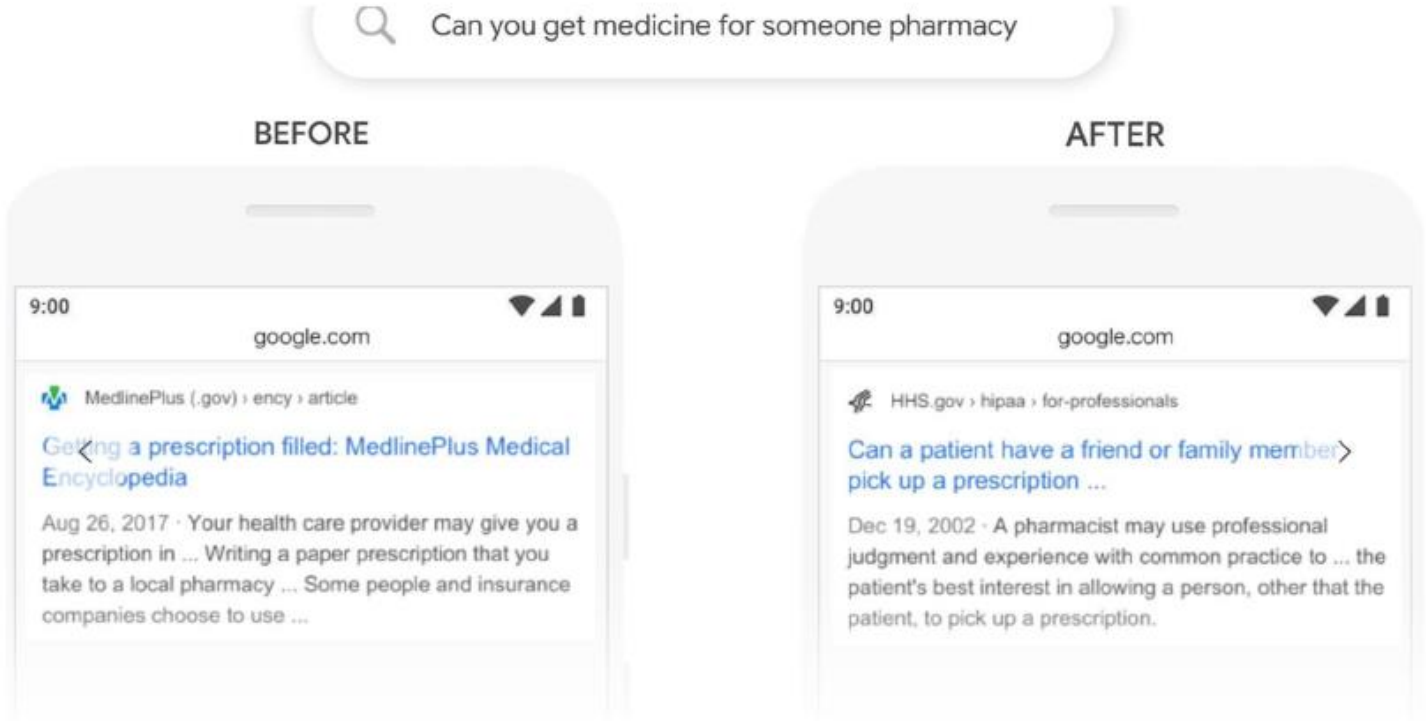
Languages (inference)

Serving

Tensorboard

…

Tensorflow – use cases in production

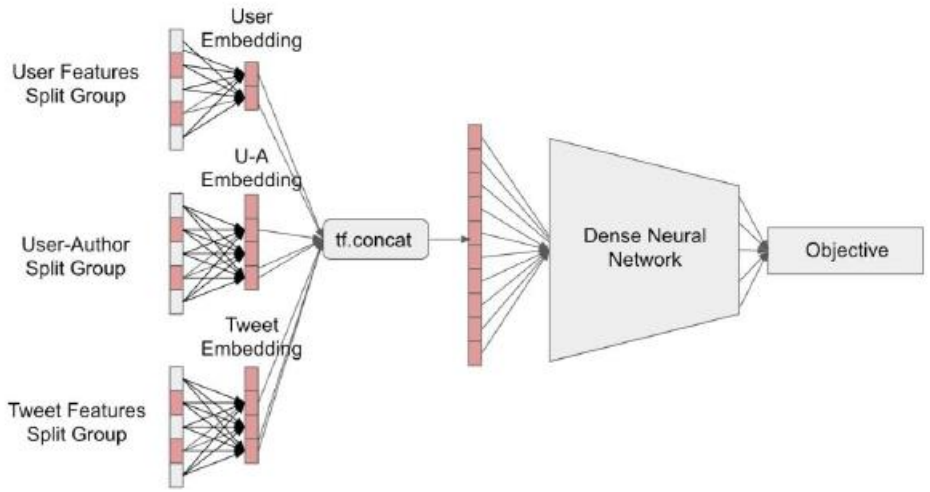


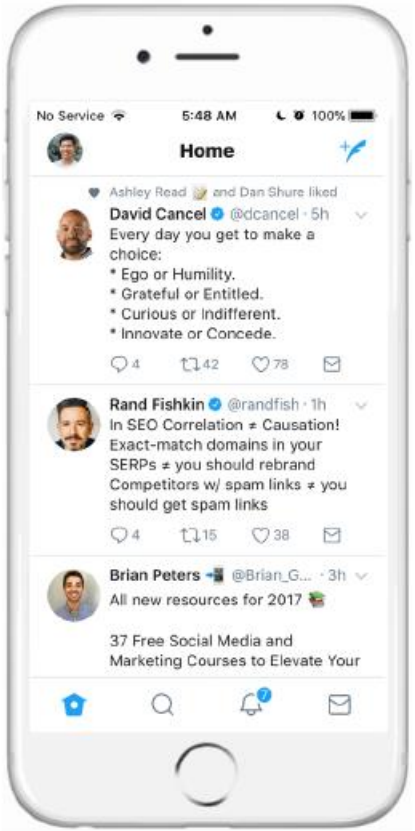
Google

"BERT will help Search better understand one in 10 searches in the U.S. in English"

https://blog.google/products/search/search-language-understanding-bert/

Tensorflow – use cases in production





https://blog.tensorflow.org/2019/03/ranking-tweets-with-tensorflow.html

Spark MLLib

Version >= 3.0

Spark MLlib - Features

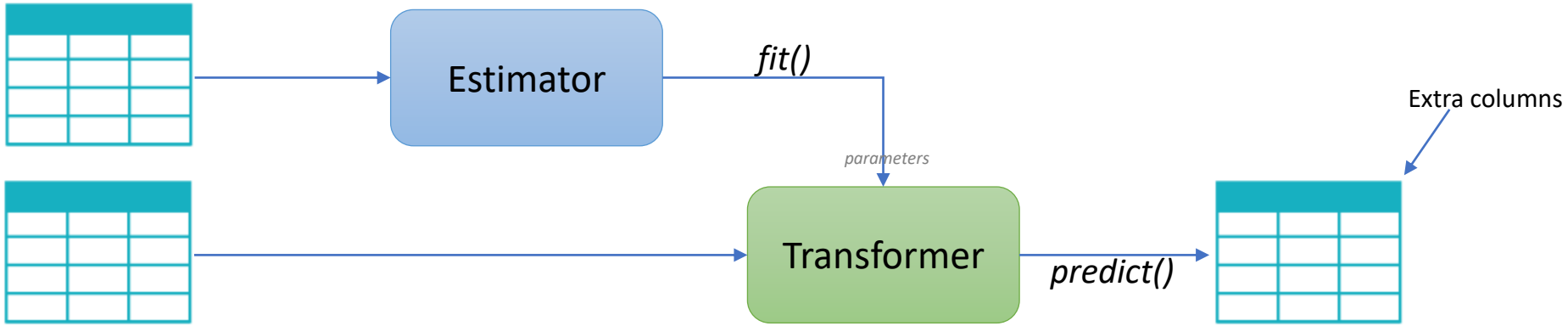
ML library built on top of Spark

- Python / Scala / Java
- Distributed

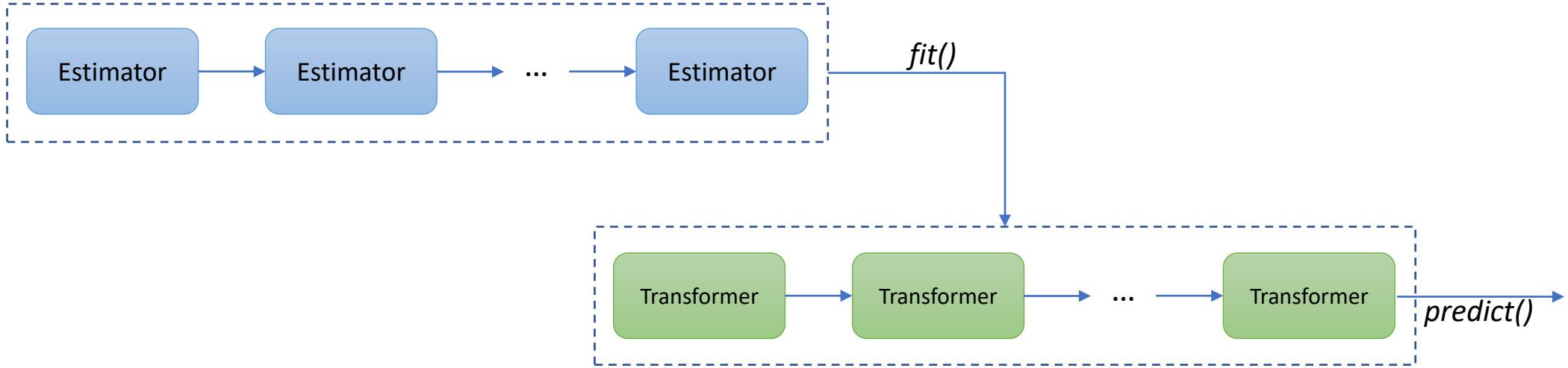
Features

- ML algorithms : classification, regression, clustering, collaborative filtering…
- Feature extraction and transformation
- Pipelines (with persistence)

Spark MLlib – Concepts



Pipeline



Spark MLlib – ML algorithms

Classification and Regression

Clustering

Collaborative Filtering…

```
from pyspark.ml.classification import LogisticRegression
estimator = LogisticRegression(maxIter=100, regParam=0.3, featuresCol="features", labelCol="label")
transformer = estimator.fit(df)
df_with_prediction = model.predict(df)
```

Vector !

Regression : numeric
Classification : [0;…;K-1]

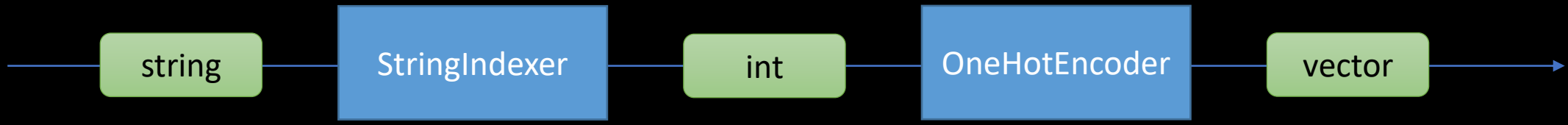
Spark MLlib – I want to … feed model with numeric column

VectorAssembler is your friend !

```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import LogisticRegression
from pyspark.ml import Pipeline

stages=[]
stages += [VectorAssembler(inputCols=["int_age"], outputCol="vec_age")]
stages += [LogisticRegression(featuresCol="vec_age", labelCol="label")]
pipeline = Pipeline(stages=stages)
```

Spark MLlib – I want to … use categorical feature




```
from pyspark.ml.feature import OneHotEncoder, StringIndexer
from pyspark.ml.classification import LogisticRegression

stages = []
stages += [StringIndexer(inputCol="cat_class", outputCol="int_class")]
stages += [OneHotEncoder(inputCols=["int_class"], outputCols=["vec_class"])]
stages += [LogisticRegression(featuresCol="vec_class", labelCol="label")]
pipeline = Pipeline(stages=stages)
```

Other methods : learning with counts
https://www.oreilly.com/library/view/strata-hadoop/9781491928004/video228193.html

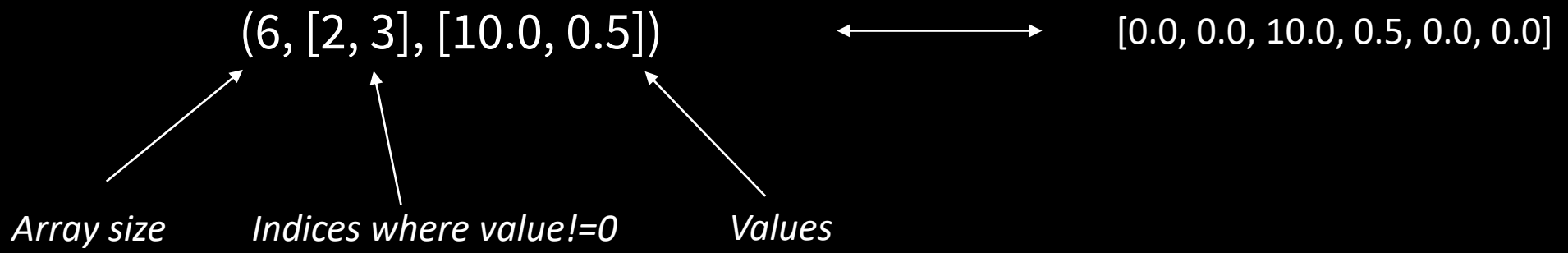
Spark MLlib – Wait my vector looks so weird now !

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	int_class	vec_class
1	0	3	Braund, Mr. Owen ...	male	22.0	1	0	A/5 21171	7.25	null	S	0.0	(2,[0],[1.0])
2	1	1	Cumings, Mrs. Joh...	female	38.0	1	0	PC 17599	71.2833	C85	C	1.0	(2,[1],[1.0])
3	1	3	Heikkinen, Miss. ...	female	26.0	0	0	STON/O2. 3101282	7.925	null	S	0.0	(2,[0],[1.0])
4	1	1	Futrelle, Mrs. Ja...	female	35.0	1	0	113803	53.1	C123	S	1.0	(2,[1],[1.0])
5	0	3	Allen, Mr. Willia...	male	35.0	0	0	373450	8.05	null	S	0.0	(2,[0],[1.0])
6	0	3	Moran, Mr. James	male	20.0	0	0	330877	8.4583	null	Q	0.0	(2,[0],[1.0])
7	0	1	McCarthy, Mr. Tim...	male	54.0	0	0	17463	51.8625	E46	S	1.0	(2,[1],[1.0])
8	0	3	Palsson, Master. ...	male	2.0	3	1	349909	21.075	null	S	0.0	(2,[0],[1.0])
9	1	3	Johnson, Mrs. Osc...	female	27.0	0	2	347742	11.1333	null	S	0.0	(2,[0],[1.0])
10	1	2	Nasser, Mrs. Nich...	female	14.0	1	0	237736	30.0708	null	C	2.0	(2,[],[])
11	1	3	Sandstrom, Miss. ...	female	4.0	1	1	PP 9549	16.7	G6	S	0.0	(2,[0],[1.0])
12	1	1	Bonnell, Miss. El...	female	58.0	0	0	113783	26.55	C103	S	1.0	(2,[1],[1.0])
13	0	3	Saundercock, Mr. ...	male	20.0	0	0	A/5. 2151	8.05	null	S	0.0	(2,[0],[1.0])
14	0	3	Andersson, Mr. An...	male	39.0	1	5	347082	31.275	null	S	0.0	(2,[0],[1.0])
15	0	3	Vestrom, Miss. Hu...	female	14.0	0	0	350406	7.8542	null	S	0.0	(2,[0],[1.0])
16	1	2	Hewlett, Mrs. (Ma...	female	55.0	0	0	248706	16.0	null	S	2.0	(2,[],[])

???

Sparse Vector representation !

Spark MLlib – Sparse Vectors



OneHotEncoder use case:

n different modalities leads to an array size of *n-1*

One modality is mapped to one dimension

Last modality is encoded as all-0 vector. Why ?

int_class	vec_class
0.0	(2,[0],[1.0])
1.0	(2,[1],[1.0])
0.0	(2,[0],[1.0])
1.0	(2,[1],[1.0])
0.0	(2,[0],[1.0])
0.0	(2,[0],[1.0])
1.0	(2,[1],[1.0])
0.0	(2,[0],[1.0])
0.0	(2,[0],[1.0])
2.0	(2,[],[])
0.0	(2,[0],[1.0])
1.0	(2,[1],[1.0])
0.0	(2,[0],[1.0])
0.0	(2,[0],[1.0])
0.0	(2,[0],[1.0])
2.0	(2,[],[])

Spark MLlib – I want to … combine columns

Depends !

FeatureHasher transforms multiple primitive columns in sparse vector

VectorAssembler will concatenate multiple vectors into a single one

Spark MLlib – Transformations Catalog

Category	Transformer	Input	Output	NbInput	Fit ?	Remarks
Bucketization	Binarizer	numeric	int	1	no	binary
	Bucketizer	numeric	int	*	yes	
	QuantileDiscretizer	numeric	int	*	yes	
Feature Selection	ChiSqSelector	vector[numeric]	vector[numeric]	1	no	
	PCA	vector[numeric]	vector[numeric]	1	yes	
	Rformulat	vector[numeric]	vector[numeric]	1	no	
	UnivariateFeatureSelector	vector[numeric]	vector[numeric]	1	no	
	VarianceThresholdSelector	vector[numeric]	vector[numeric]	1	no	
	VectorSlicer	vector[numeric]	vector[numeric]	1	no	
Hashing	FeatureHasher	numeric, boolean, string	vector[int]	*	no	doesn't hash vectors
	HashingTF	array	vector[int]	1	no	
Normalization	MaxAbsScaler	vector[numeric]	vector[numeric]	1	yes	
	MinMaxScaler	vector[numeric]	vector[numeric]	1	yes	
	Normalizer	vector[numeric]	vector[numeric]	1	yes	
	RobustScaler	vector[numeric]	vector[numeric]	1	yes	
	StandardScaler	vector[numeric]	vector[numeric]	1	yes	
Other	DCT	vector[numeric]	vector[numeric]	1	no	
	ElementwiseProduct	vector[numeric]	vector[numeric]	1	no	computes dot product with parameter
	IDF	vector[numeric]	vector[numeric]	1	yes	vector is a document
	Imputer	numeric	numeric	*	yes	
	IndexToString	int	string	1	yes	dual of StringIndexer
	Interaction	numeric, vector[numeric]	vector[numeric]	*	no	
	PolynomialExpansion	vector[numeric]	vector[numeric]	1	no	useful to compute cross-features
	StringIndexer	string	int	1	yes	
	VectorIndexer	vector[numeric]	vector[numeric]	1	yes	transforms each element to category indice if algorithm judges element worth to be categorized
Text Processing	CountVectorizer	array[string]	vector[int]	1	yes	
	NGram	array[string]	string	1	no	returns a string containing elements separated by white space (morally, returns array[string])
	StopWordsRemover	array[string]	array[string]	1	no	
	Tokenizer	string	array[string]	1	no	
	Word2Vec	array[string]	vector[numeric]	1	yes	in practice, you will probably train on a given dataset and predict on another one
Vectorization	OneHotEncoder	int	vector[numeric]	*	yes	
	VectorAssembler	numeric, vector[numeric]	vector[numeric]	*	no	Very useful : you need this one if you want to deal with continuous features without bucketization, or if you want to concatenate vectors

Spark MLlib – Want to know more ?

Read the doc

https://spark.apache.org/docs/latest/ml-guide.html

Some examples deprecated though (old api : *ml* vs *mllib*)

Code is open source !

https://github.com/apache/spark

Conclusion

Many technologies, few chosen

MLlib benefits from a nice api, just like dataframe api

Don’t reinvent the wheel

Libraries built by real humans ; they have real bugs

Read the docs !