



Spark NLP for Data Scientists

April 22, 2020

Veysel Kocaman
Sr. Data Scientist
veysel@johnsnowlabs.com

WELCOME!

60 minutes: Overview and key concepts
Splitting, cleaning, and connecting words

30 minutes: Break / Code / Q&A

60 minutes: Named entity recognition
Custom pipelines & training your own models

30 minutes: Break / Code / Q&A

60 minutes: Document classification
Model inference

Certification: Online exam open from April 22nd to May 1st
Completed in a single 90-minute session

Setup

RUNNING CODE:

https://github.com/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/Certification_Trainings/Public

[How to set up Google Colab]

BOOKMARK:

nlp.johnsnowlabs.com/docs/en/concepts
spark-nlp.slack.com

 Open in Colab

```
import os

# Install java
! apt-get install -y openjdk-8-jdk-headless -qq > /dev/null
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["PATH"] = os.environ["JAVA_HOME"] + "/bin:" + os.environ["PATH"]
! java -version

# Install pyspark
! pip install --ignore-installed -q pyspark==2.4.4

# Install Spark NLP
! pip install --ignore-installed -q spark-nlp==2.4.5
```

Branch: master			Create new file	Upload files	Find file	History
 vkocaman	updates					 Latest commit f1c22d8 5 hours ago
	..					
	 data	news dataset added				2 days ago
	 1.SparkNLP_Basics.ipynb	ngram generator added				6 days ago
	 2.Text_Preprocessing_with_SparkNLP_Annotat...	UDF text celaning added				4 days ago
	 3.SparkNLP_Pretrained_Models.ipynb	NER converter added				6 days ago
	 4.1.NerDL_Graph.ipynb	Spark NLP Public training notebooks uploaded				7 days ago
	 4.NERDL_Training.ipynb	updates				5 hours ago
	 5.1_Text_classification_examples_in_SparkML_...	classification notebook updated				2 days ago
	 5.Text_Classification_with_ClassifierDL.ipynb	classification notebook updated				2 days ago
	 6.Playground_DataFrames.ipynb	Spark NLP Public training notebooks uploaded				7 days ago
	 Spark NLP Training-Public-April 2020.pdf	slides are updated				6 days ago

Part - I

- ❖ Introducing JSL and Spark NLP
- ❖ Natural Language Processing (NLP) Basics
- ❖ Spark NLP Pretrained Pipelines
- ❖ Text Preprocessing with Spark NLP
- ❖ Spark NLP Pretrained Models



"John Snow Labs enables healthcare organizations to deploy state-of-the-art artificial intelligence (AI) platforms, models and data in production today."

JOHN SNOW LABS



"John Snow Labs wows in both proven customer success and verifiable state-of-the-art technology – making it a natural winner of the highly competitive 2019

AI Platform of the Year Award."



"Keep an eye on this company – as it represents where the industry and data science are headed."



Introducing Spark NLP

- Natural Language Toolkit (NLTK): The complete toolkit for all NLP techniques.
 - TextBlob: Easy to use NLP tools API, built on top of NLTK and Pattern.
 - SpaCy: Industrial strength NLP with Python and Cython.
 - Gensim: Topic Modelling for Humans
 - Stanford Core NLP: NLP services and packages by Stanford NLP Group.
 - Fasttext: NLP library by Facebook's AI Research (FAIR) lab
 - ...
- A blue curly brace is positioned to the right of the first seven items in the list, grouping them together.
- Spark NLP is an open-source natural language processing library, built on top of Apache Spark and Spark ML. (initial release: Oct 2017)
 - A single unified solution for all your NLP needs
 - Take advantage of transfer learning and implementing the latest and greatest SOTA algorithms and models in NLP research
 - Lack of any NLP library that's fully supported by Spark
 - Delivering a mission-critical, enterprise grade NLP library (used by multiple Fortune 500)
 - Full-time development team (26 new releases in 2018. 30 new releases in 2019.)

TRUSTED BY



Imperial College
London

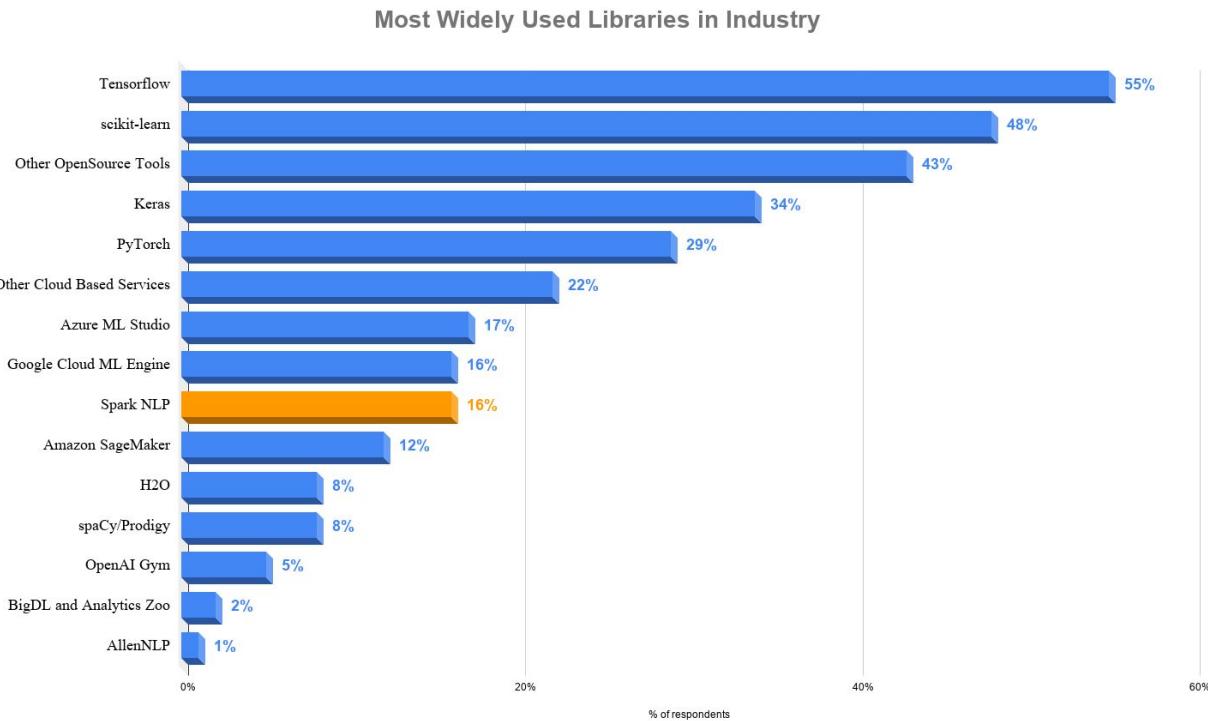


STANFORD
UNIVERSITY

Introducing Spark NLP

Name	Spark NLP	spaCy	NLTK	CoreNLP
Sentence detection	Yes	Yes	Yes	Yes
Tokenization	Yes	Yes	Yes	Yes
Stemming	Yes	Yes	Yes	Yes
Lemmatization	Yes	Yes	Yes	Yes
POS tagger	Yes	Yes	Yes	Yes
NER	Yes	Yes	Yes	Yes
Dependency parse	Yes	Yes	Yes	Yes
Text matcher	Yes	Yes	No	Yes
Date matcher	Yes	No	No	Yes
Chunking	Yes	Yes	Yes	Yes
Spell checker	Yes	No	No	No
Sentiment detector	Yes	No	No	Yes
Pretrained models	Yes	Yes	Yes	Yes
Training models	Yes	Yes	Yes	Yes

Available in **Python, R, Scala and Java**



"AI Adoption in the Enterprise", February 2019
Most widely used ML frameworks and tools survey of 1,300 practitioners by O'Reilly

OFFICIALLY SUPPORTED RUNTIMES



databricks

CLOUDERA

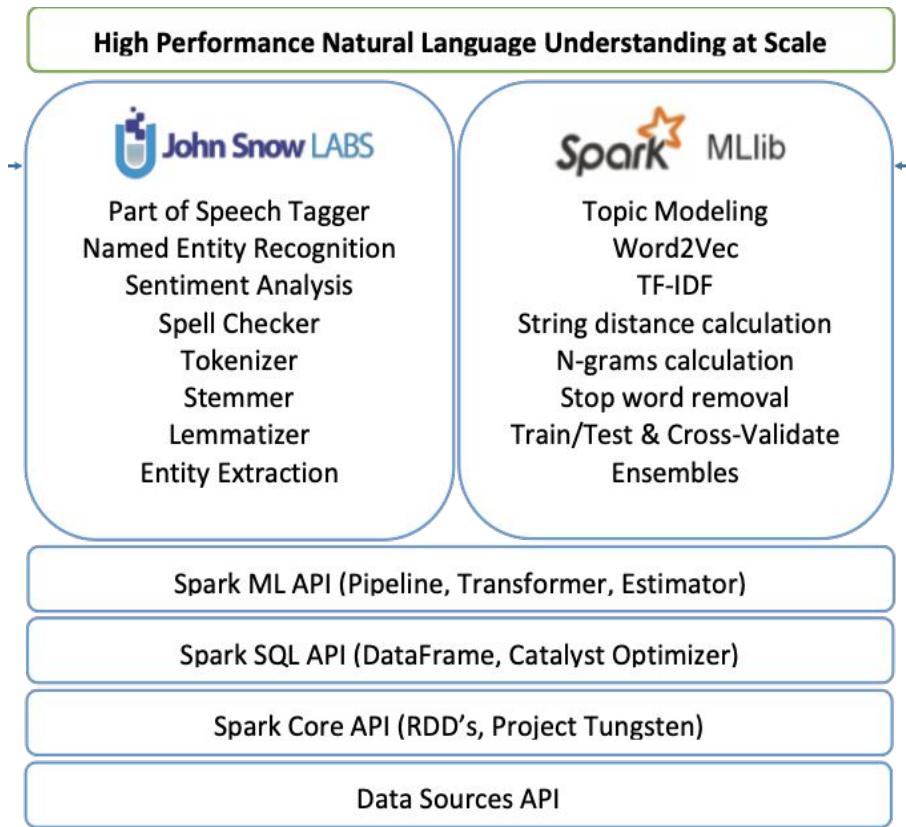


Azure



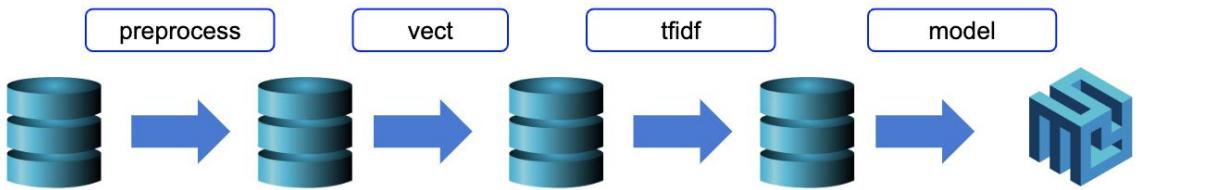
BUILT ON THE SHOULDERS OF SPARK ML

- Reusing the Spark ML Pipeline
 - Unified NLP & ML pipelines
 - End-to-end execution planning
 - Serializable
 - Distributable
- Reusing NLP Functionality
 - TF-IDF calculation
 - String distance calculation
 - Topic modeling
 - Distributed ML algorithms

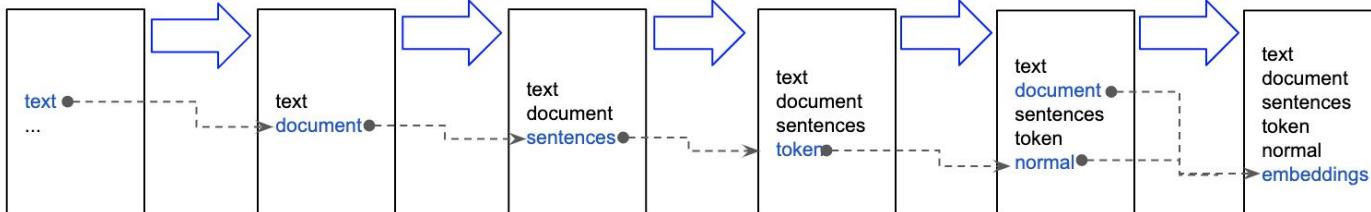


Introducing Spark NLP

Pipeline of annotators



DocumentAssembler() SentenceDetector() Tokenizer() Normalizer() WordEmbeddings()



DataFrame

```
from pyspark.ml import Pipeline
documentAssembler = DocumentAssembler()\
.setInputCol("text")\
.setOutputCol("document")
sentenceDetector = SentenceDetector()\
.setInputCols(["document"])\
.setOutputCol("sentences")
tokenizer = Tokenizer() \
.setInputCols(["sentences"]) \
.setOutputCol("token")
normalizer = Normalizer()\
.setInputCols(["token"])\
.setOutputCol("normal")
word_embeddings=WordEmbeddingsModel.pretrained()\
.setInputCols(["document","normal"])\
.setOutputCol("embeddings")
nlpPipeline = Pipeline(stages=[documentAssembler,
sentenceDetector,
tokenizer,
normalizer,
word_embeddings,
])
nlpPipeline.fit(df).transform(df)
```

Natural Language Processing

Information Retrieval

Doc A



Doc 1

Doc 2

Doc 3

Sentiment Analysis



Information Extraction



Machine Translation



Question Answering



Human: When was Apollo sent to space?

Machine: First flight -
AS-201,
February 26,
1966

NLP Basics

LEMMATIZATION

Find the **lemma** of each word:

- How does it show in the dictionary?

Uses a lookup from a full dictionary.

am, are, is → be

liver → liver

lives → live

STEMMING

Find the **stem** of each word.

Uses rules: e.g, remove common suffixes.

Form	Suffix	Stem
studies	-es	studi
study ing	- ing	study
niñ as	- as	niñ
niñ ez	- ez	niñ

- The goal of both **stemming** and **lemmatization** is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form for normalization purposes.
- Lemmatization always returns real words, **stemming** doesn't.

NLP Basics

Remove stop words and apply stemming

it was a bright cold day in april
and the clocks **were** striking
thirteen winston smith **his** chin
nuzzled **into his** breast in an
effort **to** escape **the** vile wind
slipped quickly **through the** glass
doors **of** victory mansions though
not quickly enough **to** prevent a
swirl **of** gritty dust **from** entering
along **with him**



bright cold day april clocks
striking thirteen winston smith
chin nuzzled breast effort
escape vile wind slipped quickly
glass doors victory mansions
though quickly enough prevent
swirl gritty dust entering along

- For tasks like text classification, where the text is to be classified into different categories, **stopwords** are **removed** or excluded from the given text so that more focus can be given to those words which define the meaning of the text.

Stopwords

a
able
about
above
according
accordingly
across
actually
after
afterwards
again
against
ain
all
allow
allows
almost
alone
along
already
also

(520 stopwords)

Spell Checking & Correction



```
val pipeline = PretrainedPipeline("spell_check_ml", "en")
val result = pipeline.annotate("Harry Potter is a graet muvie")

println(result("spell"))
/* will print Seq[String](..., "is", "a", "great", "movie") */
```

- 3 trainable approaches
- **Norvig Approach:**
 - Retrieves tokens and auto-corrects based on a given dictionary
- **Symmetric Delete:**
 - Uses distance metrics to find possible words
- **Context Aware:**
 - Most accurate: Judges words in context
 - Deep learning based

NORMALIZATION

Remove or replace undesirable characters or regular expressions:

from: @Have a\$ #2great birth) day>!
to: Have a great birth day!

Spark NLP also comes with a Slang normalizer:

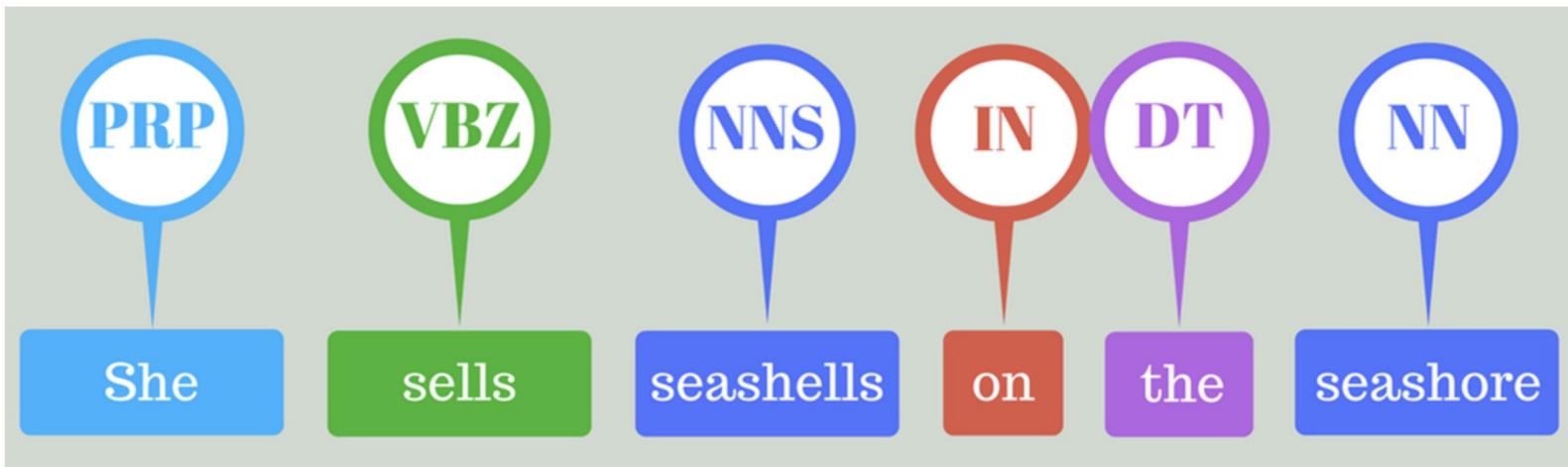
Original tweet
@USER, r u cuming 2 MidCorner dis Sunday?

Normalized tweet

@USER, are you coming to MidCorner this Sunday?

PART OF SPEECH TAGGING

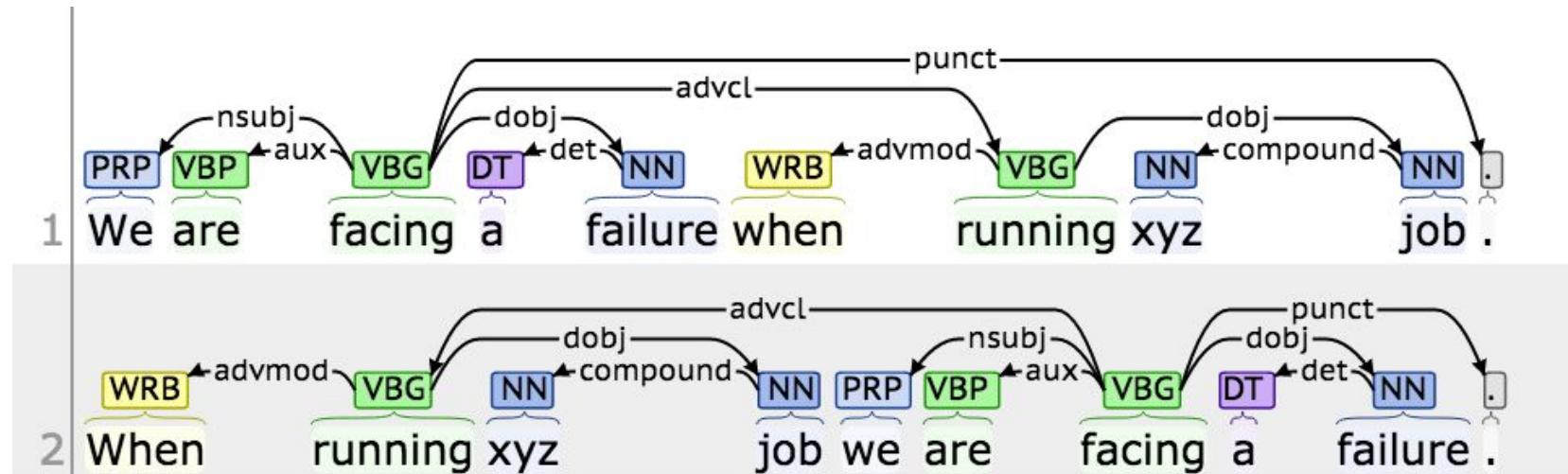
Often useful for recognizing named entities or word relationships.



A **POS tag** (or **part-of-speech tag**) is a special label assigned to each token (word) in a text corpus to indicate the **part of speech** and often also other grammatical categories such as tense, number (plural/singular), case etc.

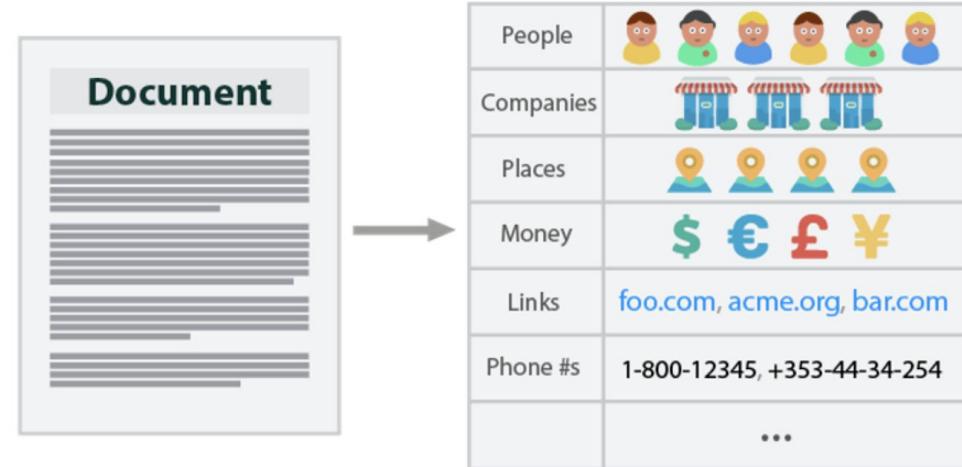
DEPENDENCY PARSING

Useful for extracting relationships (i.e. building knowledge graphs):



Named Entity Recognition (NER)

NER is a subtask of information extraction that seeks to **locate and classify named entity** mentioned in unstructured text into pre-defined categories such as **person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.**



But Google **ORG** is starting from behind. The company made a late push into hardware, and Apple **ORG**'s Siri **PRODUCT**, available on iPhones **PRODUCT**, and Amazon **ORG**'s Alexa **PRODUCT** software, which runs on its Echo **PRODUCT** and Dot **PRODUCT** devices, have clear leads in consumer adoption.



[Open in Colab](#)

Coding ...

Open 1. Spark NLP Basics and 2. Text Preprocessing with SparkNLP notebooks in [Colab](#)

(click on Colab icon or open in a new tab)

https://github.com/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/Certification_Trainings/Public/1.SparkNLP_Basics.ipynb

[https://github.com/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/Certification_Trainings/Public/2.Text_Preprocessing_with_SparkNLP\(Annotators_Transformers\).ipynb](https://github.com/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/Certification_Trainings/Public/2.Text_Preprocessing_with_SparkNLP(Annotators_Transformers).ipynb)

Spark NLP Basics

0. Colab Setup

```
[ ]: import os  
  
# Install java  
! apt-get install -y openjdk-8-jdk-headless -qq > /dev/null  
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"  
os.environ["PATH"] = os.environ["JAVA_HOME"] + "/bin;" + os.environ["PATH"]  
! java -version  
  
# Install pyspark  
! pip install --ignore-installed -q pyspark==2.4.4  
  
# Install Spark NLP  
! pip install --ignore-installed -q spark-nlp==2.4.5  
  
openjdk version "1.8.0_242"  
OpenJDK Runtime Environment (build 1.8.0_242-8u242-b08-0ubuntu3~18.04-b08)  
OpenJDK 64-Bit Server VM (build 25.242-b08, mixed mode)  
[██████████] 215.7MB 59kB/s  
[██████████] 204kB 41.9MB/s  
Building wheel for pyspark (setup.py) ... done  
[██████████] 112kB 2.7MB/s
```



[Open in Colab](#)

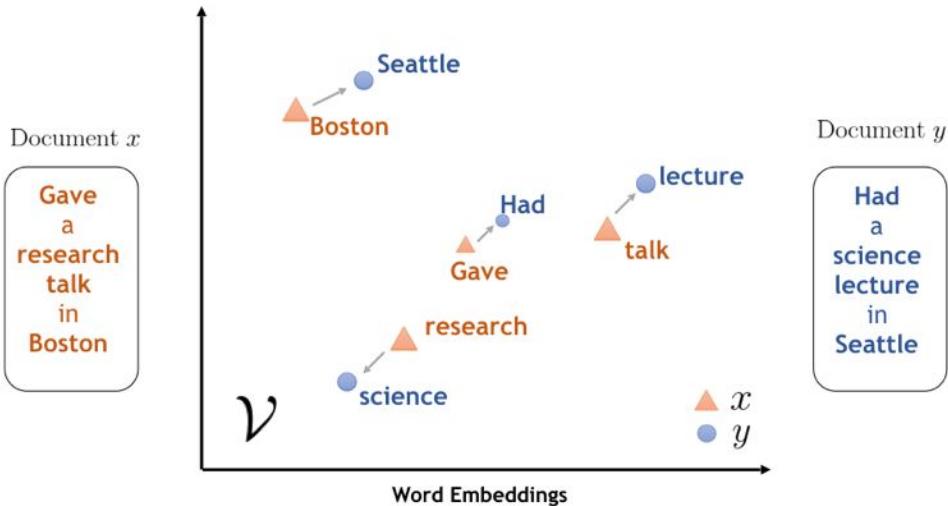
2. Text Preprocessing with Spark NLP

0. Colab Setup

```
[ ]: import os  
  
# Install java  
! apt-get install -y openjdk-8-jdk-headless -qq > /dev/null  
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"  
os.environ["PATH"] = os.environ["JAVA_HOME"] + "/bin;" + os.environ["PATH"]  
! java -version  
  
# Install pyspark  
! pip install --ignore-installed -q pyspark==2.4.4  
  
# Install Spark NLP  
! pip install --ignore-installed -q spark-nlp==2.4.5  
  
openjdk version "1.8.0_242"  
OpenJDK Runtime Environment (build 1.8.0_242-8u242-b08-0ubuntu3~18.04-b08)  
OpenJDK 64-Bit Server VM (build 25.242-b08, mixed mode)  
[██████████] 215.7MB 65kB/s  
[██████████] 204kB 48.9MB/s  
Building wheel for pyspark (setup.py) ... done  
[██████████] 112kB 2.8MB/s
```

1. Annotators and Transformer Concepts

Word & Sentence Embeddings



```
In [9]: doc[3].vector
```

```
Out[9]: array([ 0.037103 , -0.31259 , -0.17857 ,  0.30001 ,  0.078154 ,
  0.17958 ,  0.12048 , -0.11879 , -0.20601 ,  1.2849 ,
 -0.20409 ,  0.80613 ,  0.34344 , -0.19191 , -0.084511 ,
  0.17339 ,  0.042483 ,  2.0282 , -0.16278 , -0.60306 ,
 -0.53766 ,  0.35711 ,  0.22882 ,  0.1171 ,  0.42983 ,
  0.16165 ,  0.407 ,  0.036476 ,  0.52636 , -0.13524 ,
 -0.016897 ,  0.029259 , -0.079115 , -0.32305 ,  0.052255 ,
 -0.3617 , -0.18355 , -0.34717 , -0.3691 ,  0.16881 ,
  0.21018 , -0.38376 , -0.096909 , -0.36296 , -0.37319 ,
  0.0021152,  0.32512 ,  0.063977 ,  0.36249 , -0.26935 ,
 -0.59341 , -0.13625 ,  0.016425 , -0.2474 , -0.07498 ,
  0.034708 , -0.01476 , -0.11648 ,  0.25559 , -0.35002 ,
 -0.52707 ,  0.21221 ,  0.062456 ,  0.26184 ,  0.53149 ,
  0.34957 , -0.22692 ,  0.44076 ,  0.4438 ,  0.6335 ,
 -0.049757 , -0.08134 ,  0.65618 , -0.4716 ,  0.090675 ,
 -0.084873 ,  0.31455 , -0.38495 , -0.19247 ,  0.48064 ,
  0.26688 ,  0.095743 ,  0.13024 ,  0.37023 ,  0.46269 ,
 -0.32844 ,  0.17375 , -0.36325 ,  0.30672 , -0.075042 ,
 -0.64684 , -0.49822 ,  0.12372 , -0.28547 ,  0.61811 ,
 -0.19228 ,  0.0040473 ,  0.1774 ,  0.033154 , -0.54862 ,
  0.34695 , -0.53506 , -0.013381 ,  0.085712 , -0.054447 ,
 -0.64673 ,  0.016749 ,  0.47676 ,  0.037803 , -0.10066 ,
 -0.4165 , -0.20252 ,  0.2794 ,  0.10852 , -0.40154 ])
```

- Deep-Learning-based natural language processing systems.
- They encode **words** and **sentences** in fixed-length dense vectors to drastically improve the processing of textual data.
- Based on **The Distributional Hypothesis**: Words that occur in the same contexts tend to have similar meanings.

Word & Sentence Embeddings

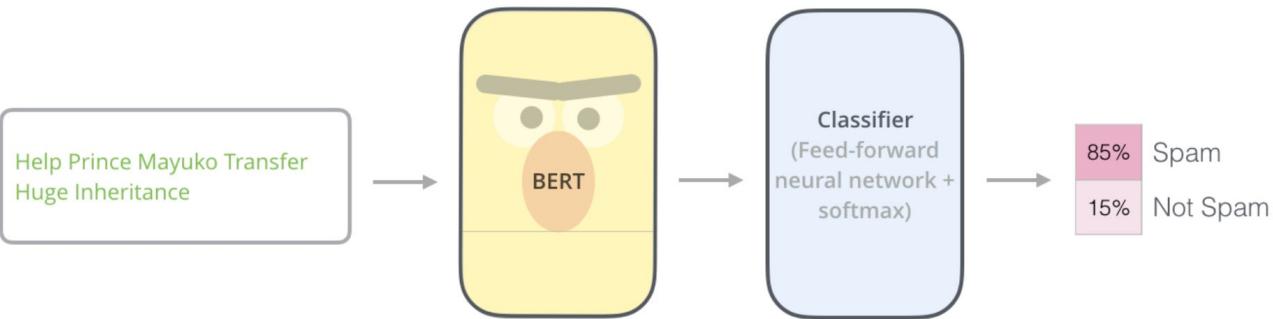
Glove
(100, 200, 300)

ELMO
(512, 1024)

BERT
(768d)

Universal Sentence Encoders
(512)

Input
Features



Output
Prediction



Coding ...

Open 3. Spark NLP Pretrained Models notebooks in [Colab](#)

(click on Colab icon or open in a new tab)

https://github.com/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/Certification_Trainings/Public/3.SparkNLP_Pretrained_Models.ipynb



[Open in Colab](#)

3. Spark NLP Pretrained Models

Spark NLP offers the following pre-trained models in five languages (English, French, German, Italian, Russia) and all you need to do is to load the pre-trained model into your disk by specifying the model name and then configuring the model parameters as per your use case and dataset. Then you will not need to worry about training a new model from scratch and will be able to enjoy the pre-trained SOTA algorithms directly applied to your own data with transform().

In the official documentation, you can find detailed information regarding how these models are trained by using which algorithms and datasets.

<https://github.com/JohnSnowLabs/spark-nlp-models>

 English - Models

Model	Name	Build	Description	Notes	Offline
LemmatizerModel (Lemmatizer)	lemma_antbnc	2.0.2			Download
PerceptronModel (POS)	pos_anc	2.0.2			Download
PerceptronModel (POS UD)	pos_ud_ewt	2.2.2			Download
NerCrFModel (NER with GloVe)	ner_crft	2.4.0			Download
NerDLModel (NER with GloVe)	ner_dl	2.4.3			Download
NerDLModel (NER with BERT)	ner_dl_bert	2.4.3			Download
NerDLModel (OntoNotes with GloVe 100d)	onto_100	2.4.0			Download
NerDLModel (OntoNotes with GloVe 300d)	onto_300	2.4.0			Download
WordEmbeddings (GloVe)	glove_100d	2.4.0			Download
BertEmbeddings (base_uncased)	bert_base_uncased	2.4.0			Download
BertEmbeddings (base_cased)	bert_base_cased	2.4.0			Download
BertEmbeddings (large_uncased)	bert_large_uncased	2.4.0			Download
BertEmbeddings (large_cased)	bert_large_cased	2.4.0			Download
ElmoEmbeddings	elmo	2.4.0			Download
UniversalSentenceEncoder	tfhub_use	2.4.0			Download
UniversalSentenceEncoder	tfhub_use_lg	2.4.0			Download
NerDLModel	ner_dl_sentence	2.4.0			Download
SymmetricDeleteModel (Spell Checker)	spellcheck_sd	2.0.2			Download
NorvigSweetingModel (Spell Checker)	spellcheck_norvig	2.0.2			Download
ViveknSentimentModel (Sentiment)	sentiment_vivekn	2.0.2			Download
DependencyParser (Dependency)	dependency_conllu	2.0.8			Download
TypedDependencyParser (Dependency)	dependency_typed_conllu	2.0.8			Download

Part - II

- ❖ Named Entity Recognition (NER) in Spark NLP

Named Entity Recognition (NER)

In fact, the Chinese NORP market has the three CARDINAL most influential names of the retail and tech space – Alibaba GPE, Baidu ORG, and Tencent PERSON (collectively touted as BAT ORG), and is betting big in the global AI GPE in retail industry space. The three CARDINAL giants which are claimed to have a cut-throat competition with the U.S. GPE (in terms of resources and capital) are positioning themselves to become the ‘future AI PERSON platforms’. The trio is also expanding in other Asian NORP countries and investing heavily in the U.S. GPE based AI GPE startups to leverage the power of AI GPE. Backed by such powerful initiatives and presence of these conglomerates, the market in APAC AI is forecast to be the fastest-growing one CARDINAL, with an anticipated CAGR PERSON of 45% PERCENT over 2018 - 2024 DATE.

To further elaborate on the geographical trends, North America Loc has procured more than 50% PERCENT of the global share in 2017 DATE and has been leading the regional landscape of AI GPE in the retail market. The U.S. GPE has a significant credit in the regional trends with over 65% PERCENT of investments (including M&As, private equity, and venture capital) in artificial intelligence technology. Additionally, the region is a huge hub for startups in tandem with the presence of tech titans, such as Google ORG, IBM ORG, and Microsoft ORG.

NER Benchmarks

SYSTEM	YEAR	LANGUAGE	ACCURACY
Spark NLP v2.4	2020	Python/Scala/Java/R	93.3 (test F1) - 95.9 (dev F1)
Spark NLP v2.x	2019	Python/Scala/Java/R	93
Spark NLP v1.x	2018	Python/Scala/Java/R	92
spaCy v2.x	2017	Python/Cython	92.6
spaCy v1.x	2015	Python/Cython	91.8
ClearNLP	2015	Java	91.7
CoreNLP	2015	Java	89.6
MATE	2015	Java	92.5
Turbo	2015	C++	92.4

- Spark NLP 2.4.x obtained the best performing academic peer-reviewed results
- State-of-the-art Deep Learning algorithms
- Achieve high accuracy within a few minutes
- Achieve high accuracy with a few lines of codes
- Blazing fast training
- Use CPU or GPU
- Easy to choose Word Embeddings
- Pre-trained GloVe models
- Pre-trained BERT models from TF Hub
- Pre-trained ELMO models from TF Hub

NER-DL in Spark NLP

CoNLL2003 format

All data files contain one word per line with empty lines representing sentence boundaries. At the end of each line there is a tag which states whether the current word is inside a named entity or not. The tag also encodes the type of named entity. Here is an example sentence:

U.N.	NNP	I-NP	I-ORG
official	NN	I-NP	O
Ekeus	NNP	I-NP	I-PER
heads	VBZ	I-VP	O
for	IN	I-PP	O
Baghdad	NNP	I-NP	I-LOC
.	.	O	O

* Each line contains four fields: the word, its part-of-speech tag, its chunk tag and its named entity tag.

* CoNLL: Conference on Computational Natural Language Learning

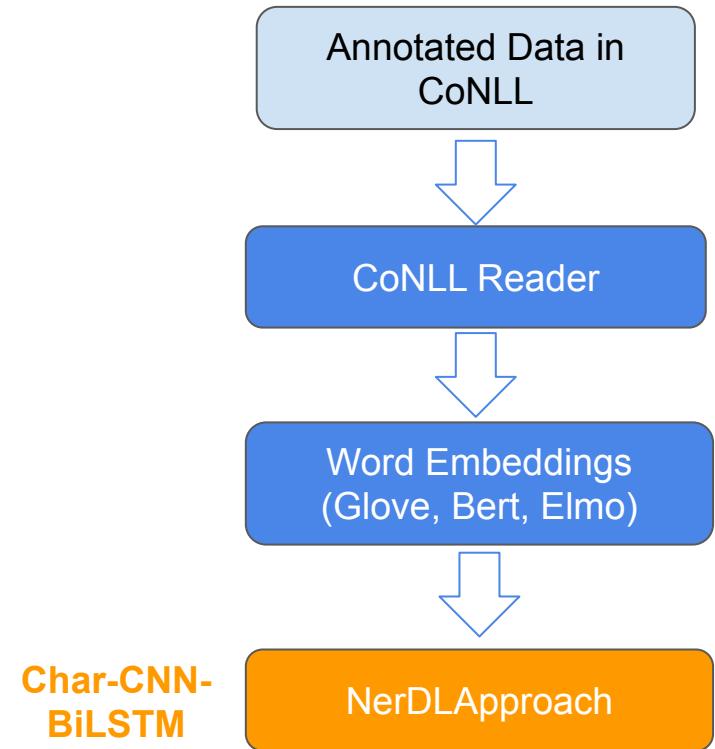
BIO schema

John	B-PER
Smith	I-PER
lives	O
in	O
New	B-LOC
York	I-LOC

John Smith ⇒ PERSON
New York ⇒ LOCATION

NER-DL in Spark NLP

Model	Name	Build	Description	Notes	Offline
LemmatizerModel (Lemmatizer)	lemma_antbnc	2.0.2			Download
PerceptronModel (POS)	pos_anc	2.0.2			Download
NerCrfModel (NER with GloVe)	ner_crf	2.4.0			Download
NerDLModel (NER with GloVe)	ner_dl	2.4.0			Download
NerDLModel (OntoNotes with GloVe 100d)	onto_100	2.4.0			Download
NerDLModel (OntoNotes with GloVe 300d)	onto_300	2.4.0			Download
WordEmbeddings (GloVe)	glove_100d	2.4.0			Download
BertEmbeddings (base_uncased)	bert_base_uncased	2.4.0			Download
BertEmbeddings (base_cased)	bert_base_cased	2.4.0			Download
BertEmbeddings (large_uncased)	bert_large_uncased	2.4.0			Download
BertEmbeddings (large_cased)	bert_large_cased	2.4.0			Download
ElmoEmbeddings	elmo	2.4.0			Download
UniversalSentenceEncoder	tf_hub_use	2.4.0			Download
UniversalSentenceEncoder	tf_hub_use_lg	2.4.0			Download
NerDLModel	ner_dl_sentence	2.4.0			Download
SymmetricDeleteModel (Spell Checker)	spellcheck_sd	2.0.2			Download
NorvigSweetingModel (Spell Checker)	spellcheck_norvig	2.0.2			Download
ViveknSentimentModel (Sentiment)	sentiment_vivekn	2.0.2			Download
DependencyParser (Dependency)	dependency_conllu	2.0.8			Download
TypedDependencyParser (Dependency)	dependency_typed_conllu	2.0.8			Download



You can also train your own Word Embeddings in Gensim and load in Spark NLP.

Coding ...

Open 4. NERDL Training notebook in Colab

(click on Colab icon or open in a new tab)

https://github.com/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/Certification_Trainings/Public/4.NERDL_Training.ipynb

Test set evaluation

```
In [ ]: import pyspark.sql.functions as F  
  
predictions.select(F.explode(F.arrays_zip('token.result','label.result','ner.result')).alias("cols")) \  
.select(F.expr("cols['0']").alias("token"),  
        F.expr("cols['1']").alias("ground_truth"),  
        F.expr("cols['2']").alias("prediction")).show(truncate=False)
```

token	ground_truth	prediction
CRICKET	o	o
-	o	o
LEICESTERSHIRE	B-ORG	B-ORG
TAKE	o	o
OVER	o	o
AT	o	o
TOP	o	o
AFTER	o	o
INNINGS	o	o
VICTORY	o	o
.	o	o
LONDON	B-LOC	B-LOC
1996-08-30	o	o
West	B-MISC	B-MISC
Indian	I-MISC	I-MISC
all-rounder	o	o
Phil	B-PER	B-PER
Simmons	I-PER	I-PER
took	o	o
four	o	o

only showing top 20 rows

```
In [ ]: from sklearn.metrics import classification_report  
  
preds_df = predictions.select(F.explode(F.arrays_zip('token.result','label.result','ner.result')).alias("cols")) \  
.select(F.expr("cols['0']").alias("token"),  
        F.expr("cols['1']").alias("ground_truth"),  
        F.expr("cols['2']").alias("prediction")).toPandas()  
  
print (classification_report(preds_df['ground_truth'], preds_df['prediction']))
```

	precision	recall	f1-score	support
B-LOC	0.88	0.93	0.90	1837
B-MISC	0.80	0.82	0.81	922
B-ORG	0.92	0.73	0.81	1341
B-PER	0.94	0.95	0.95	1842

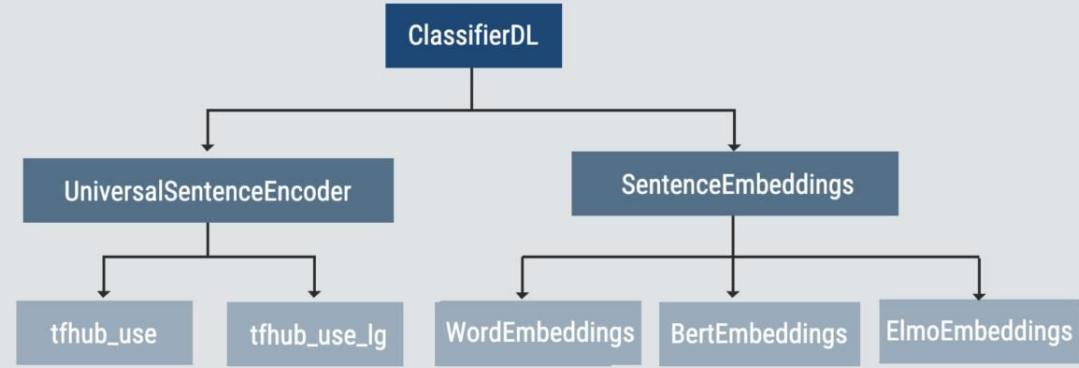
Part - III

- ❖ Text Classification with Classifier DL in Spark NLP

Text Classification with Classifier DL in Spark NLP

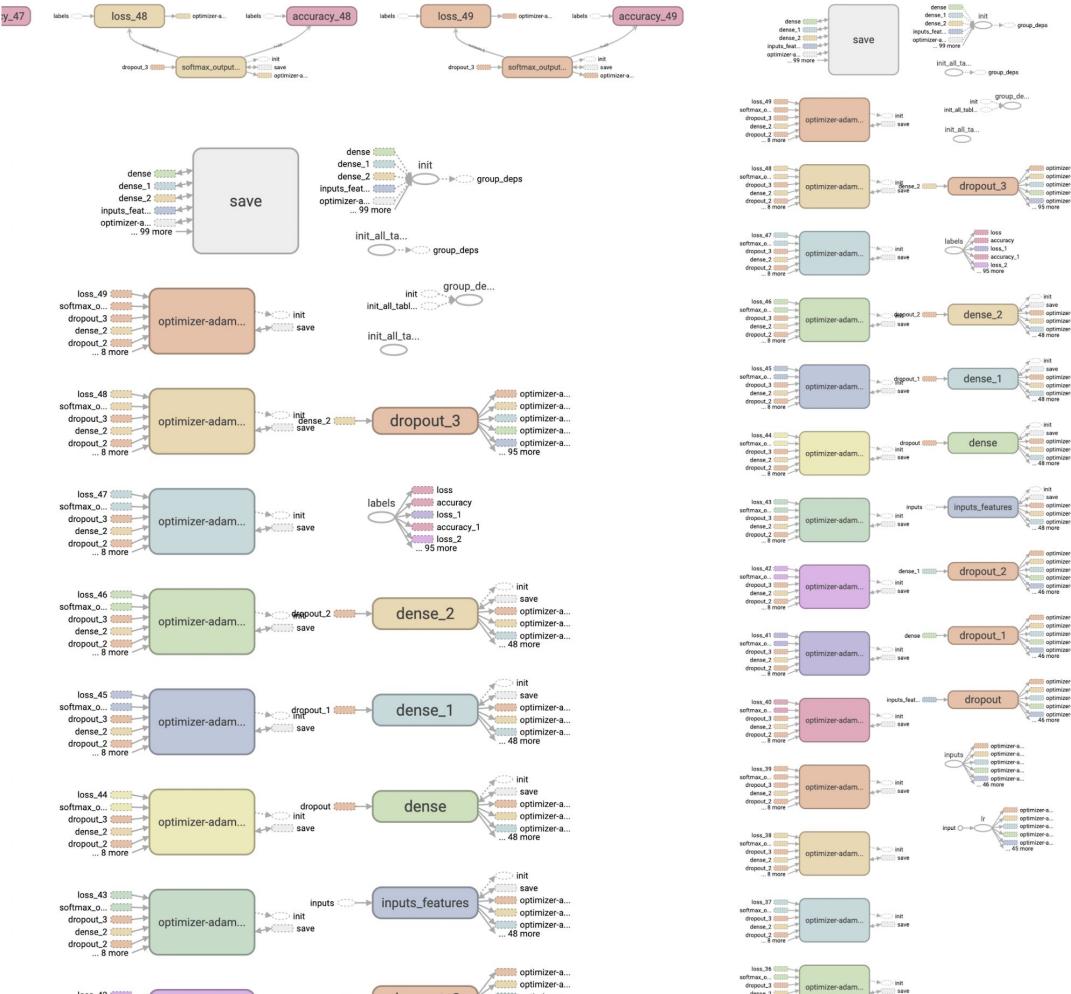
Spark NLP: ClassifierDL

Up to 50 classes, 11 models and 1024 dimensions



- 2 classes (positive/negative)
- 3 classes (0, 1, 2)
- 4 classes (Sports, Business, etc.)
- 5 classes (1.0, 2.0, 3.0, 4.0, 5.0)
- 50 classes!
- 100 dimensions
- 200 dimensions
- 300 dimensions
- 512 dimensions
- 768 dimensions
- 1024 dimensions
- tfhub_use
- tfhub_use_lg
- glove_6B_100
- glove_6B_300
- glove_840B_300
- bert_base_cased
- bert_base_uncased
- bert_large_cased
- bert_large_uncased
- bert_multi_cased
- elmo

Classifier DL Tensorflow Architecture



Coding ...

Open 5. Text Classification with ClassifierDL notebook in Colab

(click on Colab icon or open in a new tab)

https://github.com/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/Certification_Trainings/Public/4.NERDL_Training.ipynb

ClassifierDL with Universal Sentence Embeddings

```
In [ ]: # actual content is inside description column
document = DocumentAssembler() \
    .setInputCol("description") \
    .setOutputCol("document")

# we can also use sentence detector here if we want to train on and get predictions for each sentence

use = UniversalSentenceEncoder.pretrained() \
    .setInputCols(["document"]) \
    .setOutputCol("sentence_embeddings")

# the classes/labels/categories are in category column
classifierdl = ClassifierDLApproach() \
    .setInputCols(["sentence_embeddings"]) \
    .setOutputCol("class") \
    .setLabelColumn("category") \
    .setMaxEpochs(5) \
    .setEnableOutputLogs(True)

use_clf_pipeline = Pipeline(
    stages = [
        document,
        use,
        classifierdl
    ]
)
tfhub_use.download started this may take some time.
Approximate size to download 923.7 MB
[OK!]

In [ ]: use_pipelineModel = use_clf_pipeline.fit(trainDataset)
# 5 epochs takes around 10 min

In [ ]: !cd ~/annotator_logs && ls -l
total 8
-rw-r--r-- 1 root root 533 Apr  7 17:14 ClassifierDLApproach_a0fdc9e970b.log
-rw-r--r-- 1 root root 976 Apr  7 16:59 ClassifierDLApproach_f222663dfb2c.log

In [ ]: !cat ~/annotator_logs/ClassifierDLApproach_a0fdc9e970b.log
Training started - total epochs: 5 - learning rate: 0.005 - batch size: 64 - training examples: 120000
Epoch 0/5 - 34.868680102%.2fs - loss: 1620.7466 - accuracy: 0.8803833 - batches: 1875
Epoch 1/5 - 35.627811455%.2fs - loss: 1604.4518 - accuracy: 0.8915333 - batches: 1875
Epoch 2/5 - 34.687788982%.2fs - loss: 1597.8773 - accuracy: 0.8966333 - batches: 1875
Epoch 3/5 - 34.629944711%.2fs - loss: 1593.4987 - accuracy: 0.900275 - batches: 1875
Epoch 4/5 - 34.714090256%.2fs - loss: 1590.3165 - accuracy: 0.90335834 - batches: 1875
```

Spark NLP Resources

Spark NLP Official page

Spark NLP Workshop Repo

JSL Youtube channel

JSL Blogs

Introduction to Spark NLP: Foundations and Basic Components (Part-I)

Introduction to: Spark NLP: Installation and Getting Started (Part-II)

Spark NLP 101 : Document Assembler

Spark NLP 101: LightPipeline

<https://www.oreilly.com/radar/one-simple-chart-who-is-interested-in-spark-nlp/>

<https://blog.dominodatalab.com/comparing-the-functionality-of-open-source-natural-language-processing-libraries/>

<https://databricks.com/blog/2017/10/19/introducing-natural-language-processing-library-apache-spark.html>

<https://databricks.com/fr/session/apache-spark-nlp-extending-spark-ml-to-deliver-fast-scalable-unified-natural-language-processing>

<https://medium.com/@saif1988/spark-nlp-walkthrough-powered-by-tensorflow-9965538663fd>

<https://www.kdnuggets.com/2019/06/spark-nlp-getting-started-with-worlds-most-widely-used-nlp-library-enterprise.html>

<https://www.forbes.com/sites/forbestechcouncil/2019/09/17/winning-in-health-care-ai-with-small-data/#1b2fc2555664>

<https://medium.com/hackernoon/mueller-report-for-nerds-spark-meets-nlp-with-tensorflow-and-bert-part-1-32490a8f8f12>

<https://www.analyticsindiamag.com/5-reasons-why-spark-nlp-is-the-most-widely-used-library-in-enterprises/>

<https://www.oreilly.com/ideas/comparing-production-grade-nlp-libraries-training-spark-nlp-and-spacy-pipelines>

<https://www.oreilly.com/ideas/comparing-production-grade-nlp-libraries-accuracy-performance-and-scalability>

<https://www.infoworld.com/article/3031690/analytics/why-you-should-use-spark-for-machine-learning.html>

Slides:

https://github.com/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/Certification_Trainings/Public/Spark%20NLP%20Training-Public-April%202020.pdf

Colab:

https://colab.research.google.com/github/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/Certification_Trainings/Public/1.SparkNLP_Basics.ipynb

https://colab.research.google.com/github/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/Certification_Trainings/Public/2.Text_Preprocessing_with_SparkNLP_Annotators_Transformers.ipynb

https://colab.research.google.com/github/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/Certification_Trainings/Public/3.SparkNLP_Pretrained_Models.ipynb

https://colab.research.google.com/github/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/Certification_Trainings/Public/4.NERD_L_Training.ipynb

https://colab.research.google.com/github/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/Certification_Trainings/Public/5.Text_Classification_with_ClassifierDL.ipynb



John Snow LABS

**Spark NLP
for Data Scientists**

April 22, 2020

**Spark NLP
for Healthcare Data Scientist**

May 13, 2020

Veysel Kocaman
Sr. Data Scientist
veysel@johnsnowlabs.com

Welcome !

Spark NLP for Data Scientists

Overview and key concepts in Spark NLP

Text preprocessing and cleaning

Named Entity Recognition (NER) and Train your own NER

Text Classification and model inference

Spark NLP for Healthcare Data Scientists

Cleaning medical text: Normalization, stop-words, clinical POS, spell checking

Common medical NLP use cases

Clinical named entity recognition

Assertion status detection

Medical Entity Resolution (ICD, RxNorm, SNOMED)

Healthcare Data De-identification

Object Character Recognition (OCR)



Table of contents

- Text Preprocessing with Spark NLP (Annotators and Transformers)
- Colab Setup
- Annotators and Transformer Concepts
- Transformers
- Document Assembler
- Sentence Detector
- Tokenizer
- Stacking Spark NLP Annotators in Spark ML Pipeline
- Normalizer
- Stopwords Cleaner
- Stemmer
- Lemmatizer
- TextMatcher
- LightPipeline
- Section

Section



[Open in Colab](#)

Open in Colab

2. Text Preprocessing with Spark NLP (Annotators and Transformers)

0. Colab Setup

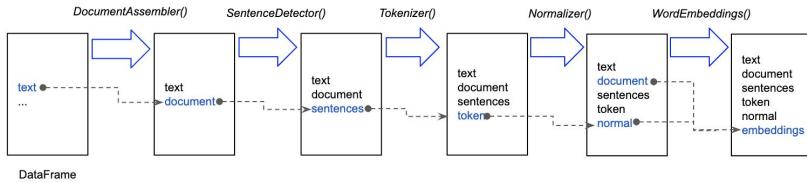
```
[ ] import os

# Install java
! apt-get install -y openjdk-8-jdk-headless -qq >/dev/null
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["PATH"] = os.environ["JAVA_HOME"] + "/bin:" + os.environ["PATH"]
! java -version

# Install pyspark
! pip install --ignore-installed -q pyspark==2.4.4

# Install Spark NLP
! pip install --ignore-installed -q spark-nlp==2.4.5

openjdk version "1.8.0_242"
OpenJDK Runtime Environment (build 1.8.0_242-8u242-b08-0ubuntu3-18.04-b08)
OpenJDK 64-Bit Server VM (build 25.242-b08, mixed mode)
Building wheel for pyspark (setup.py) ... done
    215.7MB 65kB/s
    204kB 48.9MB/s
    112kB 2.8MB/s
```



```
DOCUMENTAssembler = Tokenizer(). \
    setInputCols(["document"]) \
    .setOutputCol("token")

stemmer = Stemmer() \
    .setInputCol("token") \
    .setOutputCol("stem")

nlpPipeline = Pipeline(stages=[

    documentAssembler,
    tokenizer,
    stemmer,
    lemmatizer
])

empty_df = spark.createDataFrame([[""]]).toDF("text")

pipelineModel = nlpPipeline.fit(empty_df)

[ ] result = pipelineModel.transform(spark_df)

result.show()
```

text	document	token	stem	lemma
Peter is a very g...	[document, 0, 27...]	[token, 0, 4, Pe...	[token, 0, 4, Pe...	[Peter, is, a, ve...
My life in Russia...	[document, 0, 37...]	[token, 0, 1, My...	[token, 0, 1, My...	[My, life, in, Ru...
John and Peter ar...	[document, 0, 76...]	[token, 0, 3, Jo...	[token, 0, 3, Jo...	[John, and, Peter,...
Lucas Nogal Dunber...	[document, 0, 67...]	[token, 0, 4, lu...	[token, 0, 4, lu...	[Lucas, Nogal, Du...
Europe is very cu...	[document, 0, 68...]	[token, 0, 5, eu...	[token, 0, 5, Eu...	[Europe, is, ve...

```
[ ] result.select('lemma.result').show(truncate=False)
```

result
[Peter, is, a, ve...
[My, life, in, Ru...
[John, and, Peter,...
[Lucas, Nogal, Du...
[Europe, is, ve...



Spark NLP for Data Scientists

Session start at xx:xx am

Veysel Kocaman

Sr. Data Scientist

veysel@johnsnowlabs.com