



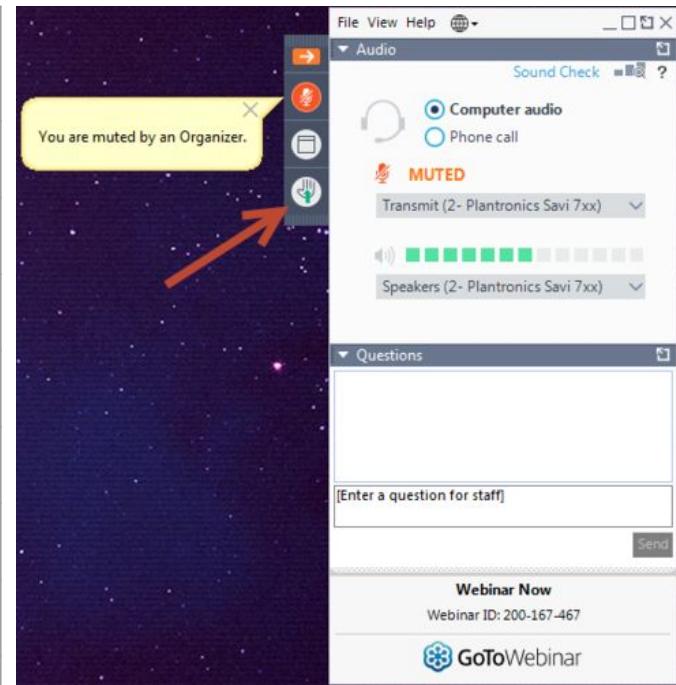
# Spark NLP for Data Scientists

July 28-29, 2020

Veysel Kocaman  
Sr. Data Scientist  
[veysel@johnsnowlabs.com](mailto:veysel@johnsnowlabs.com)

# Welcome

Day-1	60 min	- Intro to John Snow Labs and Spark NLP - Intro to NLP and Spark NLP Basics (colab)
	10 min	Break
	50 min	- Text preprocessing with Spark NLP Notebook
	10 min	Break
	50 min	- Pretrained Models in Spark NLP
Day-2	50 min	- Named Entity Recognition (NER) (part-1)
	10 min	Break
	60 min	- Named Entity Recognition (NER) (part-2)
	10 min	Break
	50 min	- Text Classification with Spark NLP



# Setup

## RUNNING CODE:

[https://github.com/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/Certification\\_Trainings/Public](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](https://nlp.johnsnowlabs.com/docs/en/concepts)  
[spark-nlp.slack.com](https://spark-nlp.slack.com)

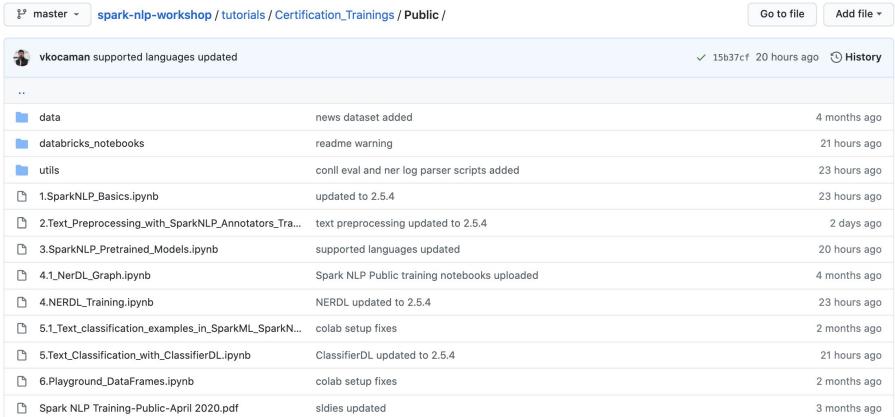
 Open in Colab

```
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
```



A screenshot of a GitHub repository page showing the commit history for the 'Public' folder. The commits are listed in reverse chronological order, with each commit showing the author, message, files changed, and the time since the commit.

Commit	Author	Message	Time Since Commit
15b37cf	vkocaman	supported languages updated	20 hours ago
...			
data		news dataset added	4 months ago
databricks_notebooks		readme warning	21 hours ago
utils		conll eval and ner log parser scripts added	23 hours ago
1.SparkNLP_Basics.ipynb		updated to 2.5.4	23 hours ago
2.Text_Preprocessing_with_SparkNLP_Annotators_Tra...		text preprocessing updated to 2.5.4	2 days ago
3.SparkNLP_Pretrained_Models.ipynb		supported languages updated	20 hours ago
4.1_NerDL_Graph.ipynb		Spark NLP Public training notebooks uploaded	4 months ago
4.NERDL_Training.ipynb		NERDL updated to 2.5.4	23 hours ago
5.1_Text_classification_examples_in_SparkML_SparkN...		colab setup fixes	2 months ago
5.Text_Classification_with_ClassifierDL.ipynb		ClassifierDL updated to 2.5.4	21 hours ago
6.Playground_DataFrames.ipynb		colab setup fixes	2 months ago
Spark NLP Training-Public-April 2020.pdf		sldies updated	3 months 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
    - [The most widely used NLP library in industry.](#)
    - 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.)

# Introducing Spark NLP

## Summary

PyPI link

Total downloads

Total downloads - 30 days

Total downloads - 7 days

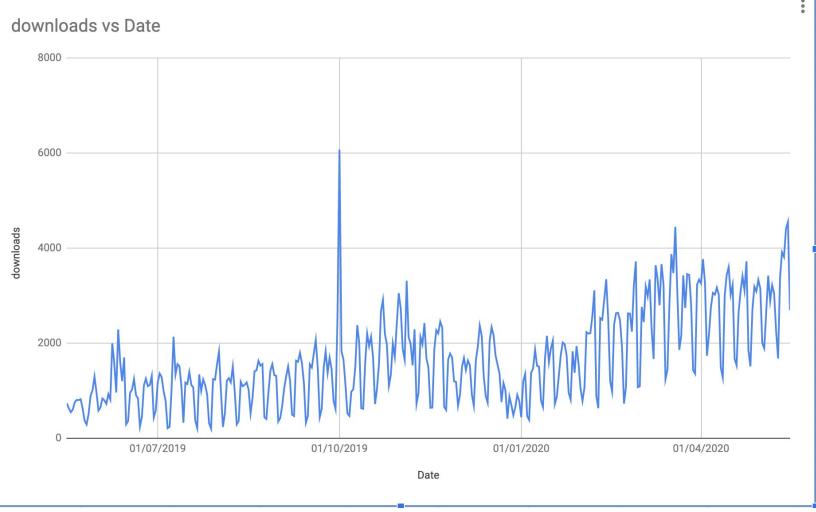
Daily ~ 4K  
Monthly > 100K

<https://pypi.org/project/spark-nlp>

975,632

125,646

37,986



- 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
  - The most widely used NLP library in industry.
  - 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

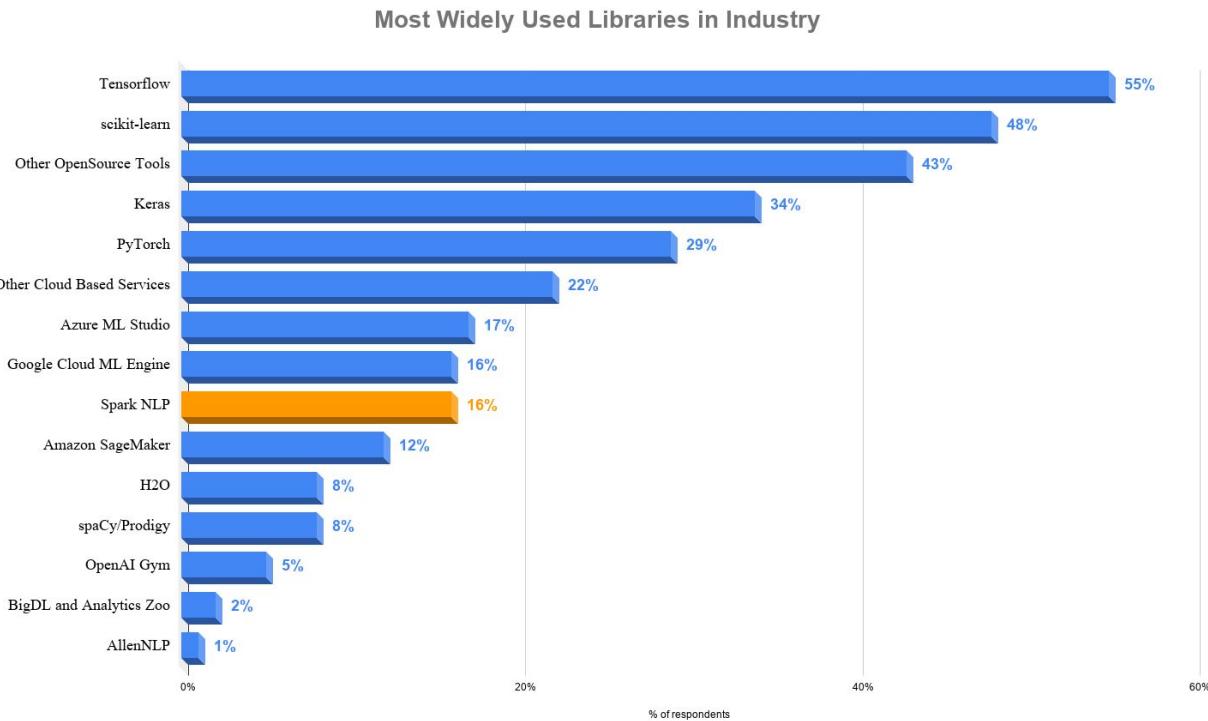


STANFORD  
UNIVERSITY

# 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



# Spark NLP Modules (Enterprise and Public)

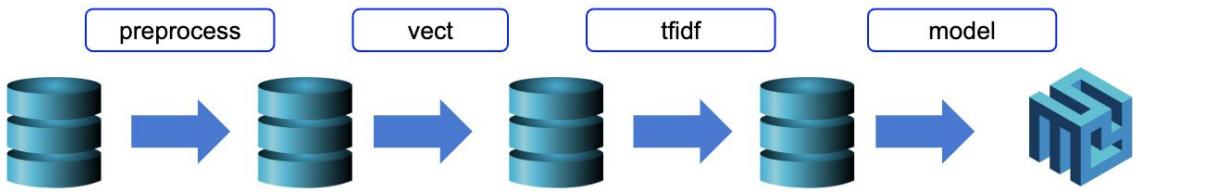
Clinical Entity Recognition	Clinical Entity Linking	Assertion Status	De-Identification						
40 units: DOSAGE of Insulin glargine DRUG at night FREQUENCY	Suspect diabetes: SNOMED-CT: 473127005 Lisinopril 10 MG RxNorm: 316151 Pyponatremia ICD-10: E87.1	Fever and sore throat → PRESENT No stomach pain → ABSENT Father with Alzheimer → FAMILY	Ora [NAME] a 25 [AGE] yo cashier [PROFESSION] from Morocco [LOCATION]						
Algorithms		Content							
<b>Extract Knowledge</b> <ul style="list-style-type: none"> <li>Entity Linker</li> <li>Entity Disambiguator</li> <li>Document Classifier</li> <li>Contextual Parser</li> </ul>	<b>De-Identity Text</b> <ul style="list-style-type: none"> <li>Structured Data</li> <li>Unstructured Text</li> <li>Obfuscator</li> <li>Generalizer</li> </ul>	<b>Medical Transformers</b>  JSL-BERT-Clinical BioBERT GloVe-Med GloVe-ICD-O	<b>Linked Medical Terminologies</b>  SNOMED-CT CPT ICD-10-CM RxNorm ICD-10-PCS ICD-O						
<b>Split Text</b> <ul style="list-style-type: none"> <li>Sentence Detector</li> <li>Deep Sentence Detector</li> <li>Tokenizer</li> <li>nGram Generator</li> </ul>	<b>Clean Medical Text</b> <ul style="list-style-type: none"> <li>Spell Checking</li> <li>Spell Correction</li> <li>Normalizer</li> <li>Stopword Cleaner</li> </ul>	<b>50+ Pretrained Models</b> <table border="1"> <tr> <td><b>Clinical:</b> Signs, Symptoms, Treatments, Procedures, Tests, Labs</td> <td><b>Anatomy:</b> Organ, Subdivision, Cell, Structure</td> </tr> <tr> <td><b>Biological:</b> Organism, Tissue, Gene, Chemical</td> <td><b>Demographics:</b> Age, Gender, Vital Signs, Smoking Indicators</td> </tr> <tr> <td><b>Drugs:</b> Name, Dosage, Strength, Route, Duration, Frequency</td> <td><b>Sensitive Data:</b> Patient Name, Address, Dates, Providers, Identifiers</td> </tr> </table>		<b>Clinical:</b> Signs, Symptoms, Treatments, Procedures, Tests, Labs	<b>Anatomy:</b> Organ, Subdivision, Cell, Structure	<b>Biological:</b> Organism, Tissue, Gene, Chemical	<b>Demographics:</b> Age, Gender, Vital Signs, Smoking Indicators	<b>Drugs:</b> Name, Dosage, Strength, Route, Duration, Frequency	<b>Sensitive Data:</b> Patient Name, Address, Dates, Providers, Identifiers
<b>Clinical:</b> Signs, Symptoms, Treatments, Procedures, Tests, Labs	<b>Anatomy:</b> Organ, Subdivision, Cell, Structure								
<b>Biological:</b> Organism, Tissue, Gene, Chemical	<b>Demographics:</b> Age, Gender, Vital Signs, Smoking Indicators								
<b>Drugs:</b> Name, Dosage, Strength, Route, Duration, Frequency	<b>Sensitive Data:</b> Patient Name, Address, Dates, Providers, Identifiers								
<b>Clinical Grammar</b> <ul style="list-style-type: none"> <li>Stemmer</li> <li>Lemmatizer</li> <li>Part of Speech Tagger</li> <li>Dependency Parser</li> </ul>	<b>Find in Text</b> <ul style="list-style-type: none"> <li>Text Matcher</li> <li>Regex Matcher</li> <li>Date Matcher</li> <li>Chunker</li> </ul>								

Trainable & Tunable	Scalable to a Cluster	Fast Inference	Hardware Optimized	Community
			 	

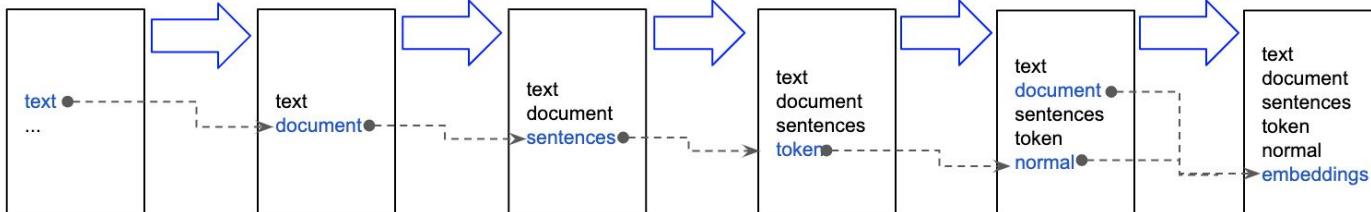
Entity Recognition	Information Extraction	Sentiment Analysis	Document Classification
I love [NAME] PERSON	They met [DATE] >> 29-04-2020		
Algorithms		Content	
<b>Split Text</b> <ul style="list-style-type: none"> <li>Sentence Detector</li> <li>Deep Sentence Detector</li> <li>Tokenizer</li> <li>nGram Generator</li> </ul>	<b>Clean Text</b> <ul style="list-style-type: none"> <li>Spell Checking</li> <li>Spell Correction</li> <li>Normalizer</li> <li>Stopword Cleaner</li> </ul>	<b>Transformers</b>  GloVe ELMO BERT ALBERT XLNet	<b>Languages</b>  Bulgarian Czech Dutch English French German Greek Hungarian Italian Finnish Norwegian Polish Portuguese Spanish Romanian Russian Swedish Turkish Ukrainian
<b>Understand Grammar</b> <ul style="list-style-type: none"> <li>Stemmer</li> <li>Lemmatizer</li> <li>Part of Speech Tagger</li> <li>Dependency Parser</li> </ul>	<b>Find in Text</b> <ul style="list-style-type: none"> <li>Text Matcher</li> <li>Regex Matcher</li> <li>Date Matcher</li> <li>Chunker</li> </ul>	<b>Models</b>  <b>90+</b> Pretrained	<b>Pipelines</b>  <b>70+</b> Pretrained
<b>Trainable &amp; Tunable</b>  	<b>Scalable to a Cluster</b>  	<b>Fast Inference</b>  	<b>Hardware Optimized</b>   
<b>Community</b>  			

# Introducing Spark NLP

## Pipeline of annotators



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



```
from pyspark.ml import Pipeline
document_assembler = 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=[document_assembler,
    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 <b>ing</b>	- <b>ing</b>	study
niñ <b>as</b>	- <b>as</b>	niñ
niñ <b>ez</b>	- <b>ez</b>	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

# Context Spell Checker

The Spell Checker can leverage the context of words for ranking different correction sequences. Let's take a look at some examples,

```
# check for the different occurrences of the word "siter"
example1 = ["I will call my siter.", \
            "Due to bad weather, we had to move to a different siter.", \
            "We travelled to three siter in the summer."]
beautify(lp.annotate(example1))
```

```
['I will call my sister .\n',
 'Due to bad weather , we had to move to a different site .\n',
 'We travelled to three sites in the summer .\n']
```

```
# check for the different occurrences of the word "ueather"
example2 = ["During the summer we have the best ueather.", \
            "I have a black ueather jacket, so nice.", \
            "I introduce you to my sister, she is called ueather."]
beautify(lp.annotate(example2))
```

```
['During the summer we have the best weather .\n',
 'I have a black leather jacket , so nice .\n',
 'I introduce you to my sister , she is called Heather .\n']
```

Notice that in the first example, 'siter' is indeed a valid English word,

<https://www.merriam-webster.com/dictionary/siter>

# NORMALIZATION

Remove or replace undesirable characters or regular expressions:

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

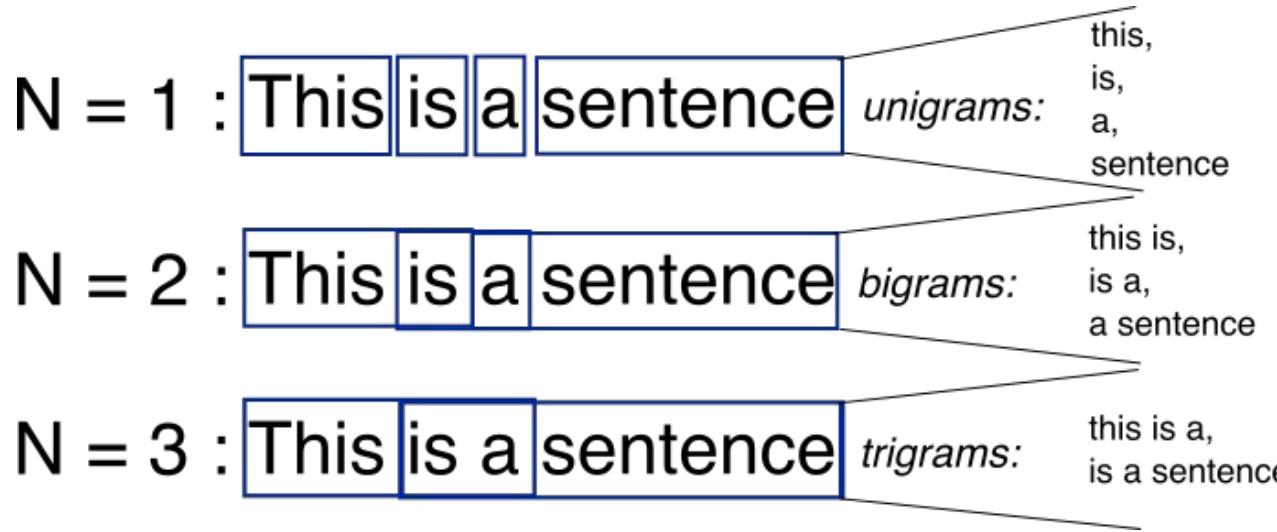
Spark NLP also comes with a Slang normalizer:

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

**Normalized tweet**

@USER, are you coming to MidCorner this Sunday?

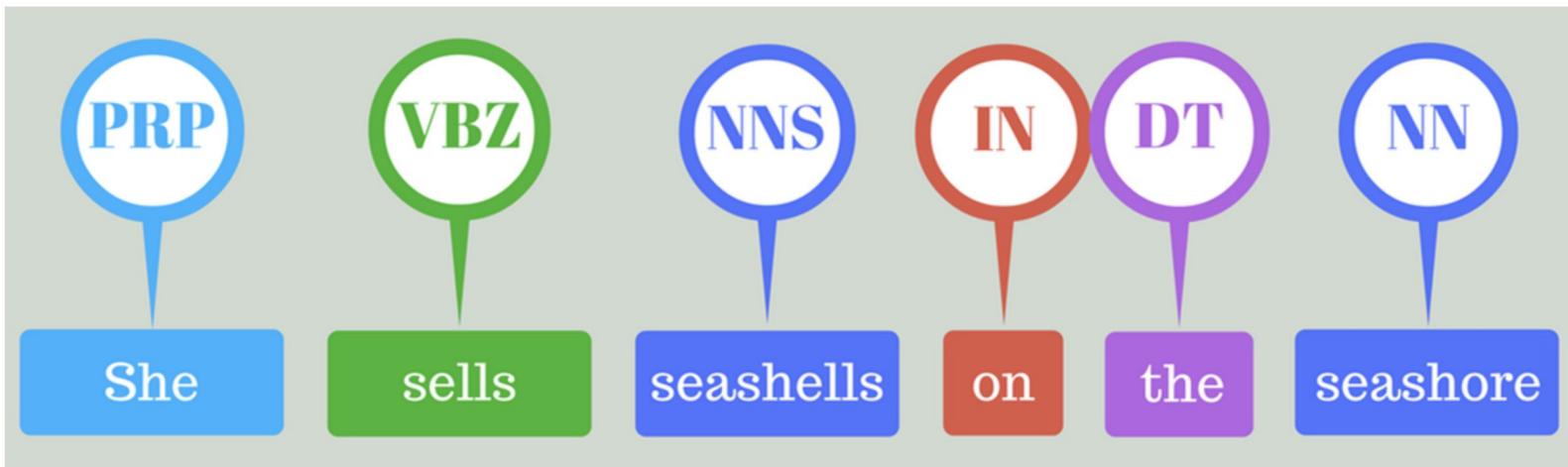
# N-gram Tokenization



- Kind of tokenizers which split words or sentences into several tokens
- Each token has certain number of characters
- Number of character depends on the type of ngram tokenizer
- Unigram, bigram, trigram, etc.

# PART OF SPEECH TAGGING

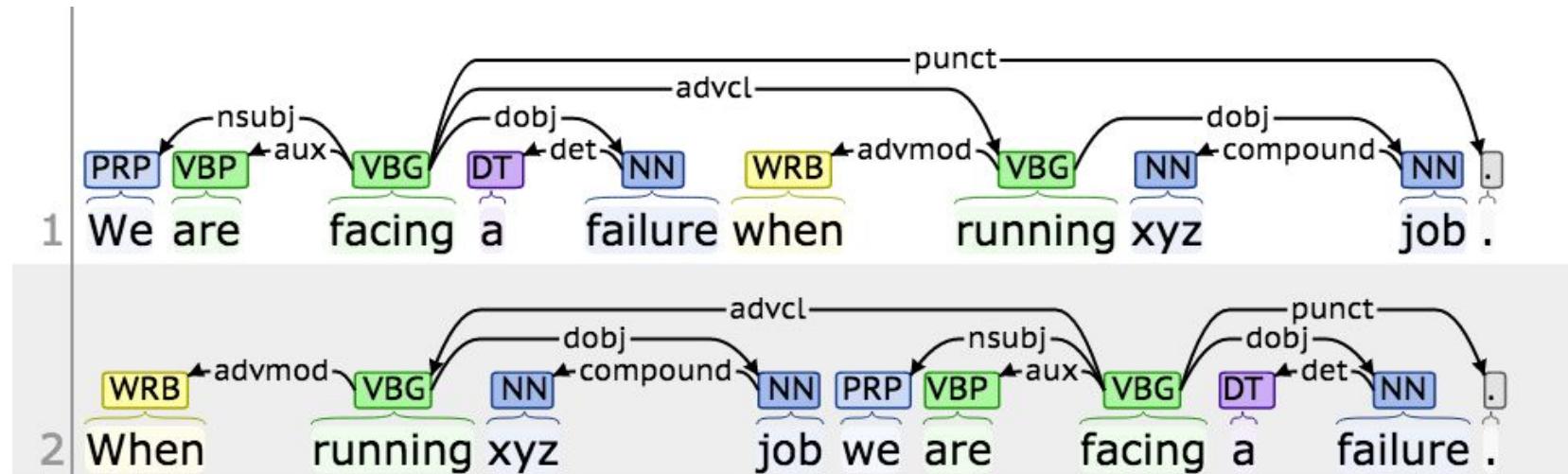
Often useful for recognizing named entities or word relationships.



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.

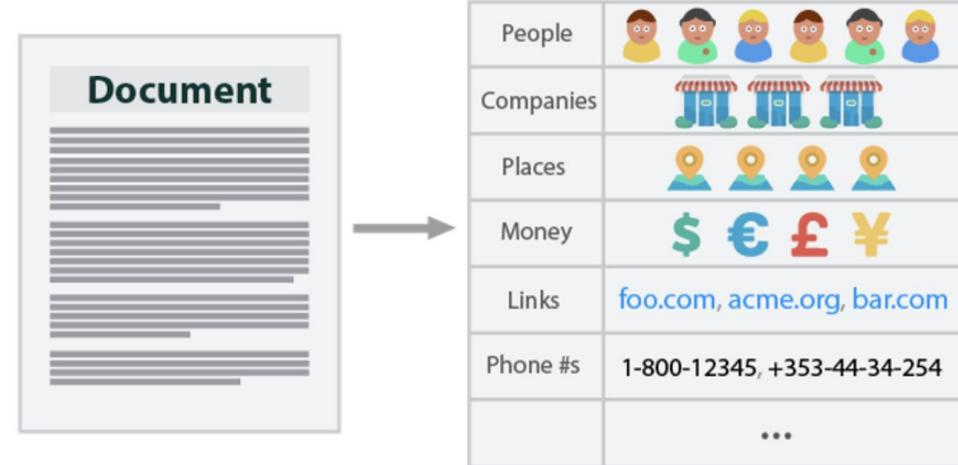
# DEPENDENCY PARSING

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.

# Word & Sentence Embeddings

Glove  
(100, 200, 300)

ELMO  
(512, 1024)

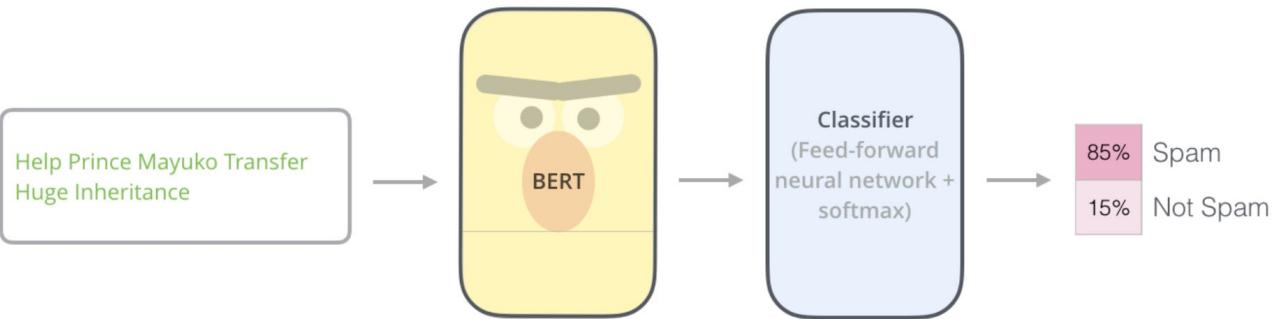
BERT  
(768d)

Albert  
(768, 1024, 2048, 4096)

XLNet  
(768, 1024)

Universal Sentence Encoders  
(512)

Input  
Features



Output  
Prediction



# Word & Sentence Embeddings

albert\_base = [https://tfhub.dev/google/albert\\_base/3](https://tfhub.dev/google/albert_base/3) |  
768-embed-dim, 12-layer, 12-heads, 12M parameters

albert\_large = [https://tfhub.dev/google/albert\\_large/3](https://tfhub.dev/google/albert_large/3) |  
1024-embed-dim, 24-layer, 16-heads, 18M parameters

albert\_xlarge = [https://tfhub.dev/google/albert\\_xlarge/3](https://tfhub.dev/google/albert_xlarge/3) |  
2048-embed-dim, 24-layer, 32-heads, 60M parameters

albert\_xxlarge = [https://tfhub.dev/google/albert\\_xxlarge/3](https://tfhub.dev/google/albert_xxlarge/3) |  
4096-embed-dim, 12-layer, 64-heads, 235M parameters

XLNet-Large =  
[https://storage.googleapis.com/xlnet/released\\_models/cased\\_L-24\\_H-1024\\_A-16.zip](https://storage.googleapis.com/xlnet/released_models/cased_L-24_H-1024_A-16.zip) | 24-layer, 1024-hidden, 16-heads

XLNet-Base =  
[https://storage.googleapis.com/xlnet/released\\_models/cased\\_L-12\\_H-768\\_A-12.zip](https://storage.googleapis.com/xlnet/released_models/cased_L-12_H-768_A-12.zip) | 12-layer, 768-hidden, 12-heads.

<https://nlp.johnsnowlabs.com/api/>

**BERT** is a bi-directional transformer for pre-training over a lot of unlabeled textual data to learn a language representation that can be used to fine-tune for specific machine learning tasks. While BERT outperformed the NLP state-of-the-art on several challenging tasks, its performance improvement could be attributed to the bidirectional transformer, novel pre-training tasks of Masked Language Model and Next Structure Prediction along with a lot of data and Google's compute power.

**XLNet** is a large bidirectional transformer that uses improved training methodology, larger data and more computational power to achieve better than BERT prediction metrics on 20 language tasks.

To improve the training, XLNet introduces permutation language modeling, where all tokens are predicted but in random order. This is in contrast to BERT's masked language model where only the masked (15%) tokens are predicted.

**Albert** is a Google's new "ALBERT" language model and achieved state-of-the-art results on three popular benchmark tests for natural language understanding (NLU): GLUE, RACE, and SQuAD 2.0. ALBERT is a "lite" version of Google's 2018 NLU pretraining method BERT. Researchers introduced two parameter-reduction techniques in ALBERT to lower memory consumption and increase training speed.

# Coding ...

## 1. Spark NLP Basics

## 2. Text Preprocessing with Spark NLP

## 3. Spark NLP Pretrained Models

(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/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)

[https://github.com/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/Certification\\_Trainings/Public/3.SparkNLP\\_Pretrained\\_Models.ipynb](https://github.com/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/Certification_Trainings/Public/3.SparkNLP_Pretrained_Models.ipynb)

### Spark NLP Basics and Pretrained Pipelines

- Colab Setup
- How to prevent Google Colab from disconnecting?
- Start Spark Session
- Using Pretrained Pipelines
  - Explain Document ML
  - Explain Document DL
  - Recognize Entities DL
  - Clean Stop Words
  - Clean Slang
  - Spell Checker
  - Spell Checker DL
  - Parsing a list of texts
  - Using fullAnnotate to get more details
  - Use pretrained match\_chunk Pipeline for Individual Noun Phrase
  - Extract exact dates from referential date phrases
  - Sentiment Analysis
    - Vivek algo
    - DL version (trained on imdb)
    - DL version (trained on twitter dataset)

### Text Preprocessing with Spark NLP

- Colab Setup
- Annotators and Transformer Concepts
- Create Spark Dataframe
- Transformers
- Document Assembler
- Sentence Detector
- Tokenizer
- Stacking Spark NLP Annotators in Spark ML Pipeline
- Normalizer
- Stopwords Cleaner
- Token Assembler
- Stemmer
- Lemmatizer
- NGram Generator
- TextMatcher
- RegexMatcher
- Text Cleaning with UDF
- Finisher
- LightPipeline

### Spark NLP Pretrained Models

- Colab Setup
- LemmatizerModel
- PerceptronModel (POS - Part of speech tags)
- Chunker
- Dependency Parser
- SpellChecker
  - Norvig Spell Checker
  - Context SpellChecker
- Language Detector
- Embeddings
  - Word Embeddings (Glove)
  - Using your own Word embeddings in Spark NLP
- Elmo Embeddings
- Bert Embeddings
- Albert Embeddings
- XlnetEmbeddings
- Chunk Embeddings
- UniversalSentenceEncoder
- Loading Models from local
- Getting Sentence Embeddings from word embeddings
  - Cosine similarity between two embeddings (sentence similarity)
- NERDL Model
  - Public NER (CoNLL 2003)
  - NerDL OntoNotes 100D
  - NER with Bert (Onto)
  - Getting the NER chunks with NER Converter
- Highlight the entities



# Spark NLP for Data Scientists

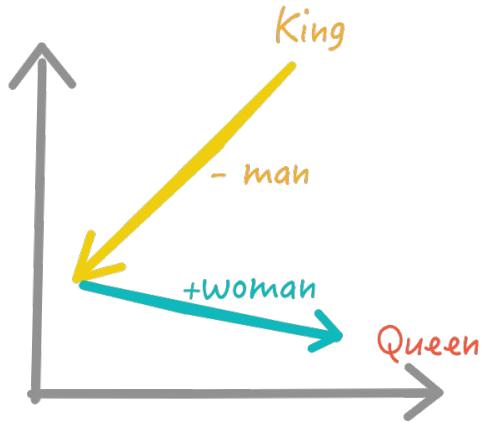
Session start at 14:45

Veysel Kocaman

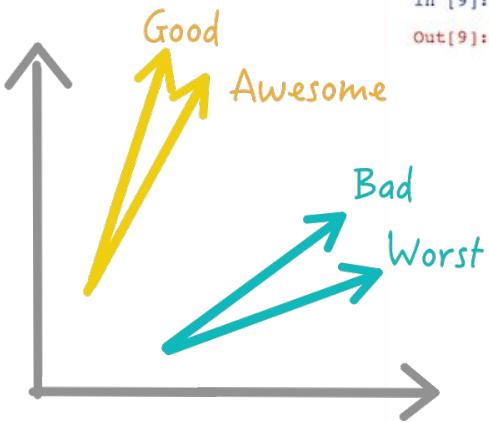
Sr. Data Scientist

[veysel@johnsnowlabs.com](mailto:veysel@johnsnowlabs.com)

# Word & Sentence Embeddings



a) Learns Analogy



b) Similar Words have same angles

```
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

## Vocabulary

index: Word:

0 aardvark  
1 able  
...

2409 black  
2410 bling  
...

3202 candid

3203 cast

3204 cat

...

5281 is

5282 island

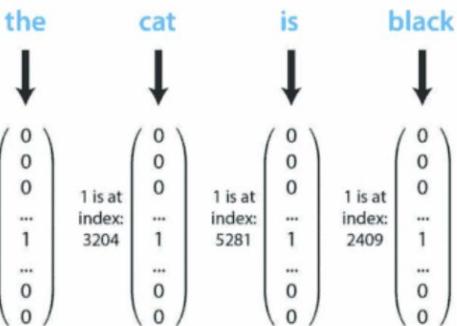
...

8676 the

8677 thing

...

9999 zombie



One-hot vector encoding for words in input sentence complete.

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 ,
 0.00211152,  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 ])
```

- Words that are used in similar contexts will be given similar representations. That is, words that are used in similar ways will be placed close together within the high-dimensional semantic space—these points will cluster together, and their distance to each other will be low.

# 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](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>

image.png

In [ ]:

#### 0. Colab Setup

```
import os

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

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

# Install Spark NLP
!pip install --ignore-installed -q spaCy
openjdk version "1.8.0_242"
OpenJDK Runtime Environment (build 1.8.0_242-b14)
OpenJDK 64-Bit Server VM (build 25.242-b14, mixed mode, sharing)
```

English - Models

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



# Spark NLP for Data Scientists

Session start at 13:30

Veysel Kocaman

Sr. Data Scientist

[veysel@johnsnowlabs.com](mailto:veysel@johnsnowlabs.com)

# Part - II

- ❖ Named Entity Recognition (NER) in Spark NLP

## CoNLL 2003 (English)

The CoNLL 2003 NER task consists of newswire text from the Reuters RCV1 corpus tagged with four different entity types (PER, LOC, ORG, MISC). Models are evaluated based on span-based F1 on the test set. \* used both the train and development splits for training.

Model	F1	Paper / Source	Code
CNN Large + fine-tune (Baevski et al., 2019)	93.5	Cloze-driven Pretraining of Self-attention Networks	
RNN-CRF+Flair	93.47	Improved Differentiable Architecture Search for Language Modeling and Named Entity Recognition	
LSTM-CRF+ELMo+BERT+Flair	93.38	Neural Architectures for Nested NER through Linearization	Official
Flair embeddings (Akbik et al., 2018)*	93.09	Contextual String Embeddings for Sequence Labeling	Flair framework
BERT Large (Devlin et al., 2018)	92.8	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding	
CVT + Multi-Task (Clark et al., 2018)	92.61	Semi-Supervised Sequence Modeling with Cross-View Training	Official
BERT Base (Devlin et al., 2018)	92.4	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding	
BILSTM-CRF+ELMo (Peters et al., 2018)	92.22	Deep contextualized word representations	AllenNLP Project AllenNLP GitHub
Peters et al. (2017) *	91.93	Semi-supervised sequence tagging with bidirectional language models	
CRF + AutoEncoder (Wu et al., 2018)	91.87	Evaluating the Utility of Hand-crafted Features in Sequence Labelling	Official
Bi-LSTM-CRF + Lexical Features (Ghadhar and Langlais 2018)	91.73	Robust Lexical Features for Improved Neural Network Named-Entity Recognition	Official
BILSTM-CRF + IntNet (Xin et al., 2018)	91.64	Learning Better Internal Structure of Words for Sequence Labeling	
Chiu and Nichols (2016) *	91.62	Named entity recognition with bidirectional LSTM-CNNs	

# NER-DL in Spark NLP

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

The best NER score in production

93.3 %  
Test Set



Bert



NerDLApproach

# NER Systems

Feature-engineered machine learning systems	Dict	SP	DU	EN	GE
Carreras et al. (2002) binary AdaBoost classifiers	Yes	81.39	77.05	-	-
Malouf (2002) - Maximum Entropy (ME) + features	Yes	73.66	68.08	-	-
Li et al. (2005) SVM with class weights	Yes	-	-	88.3	-
Passos et al. (2014) CRF	Yes	-	-	90.90	-
Ando and Zhang (2005a) Semi-supervised state of the art	No	-	-	89.31	75.27
Agerri and Rigau (2016)	Yes	<b>84.16</b>	<b>85.04</b>	<b>91.36</b>	<b>76.42</b>
Feature-inferring neural network word models					
Collobert et al. (2011) Vanilla NN +SLL / Conv-CRF	No	-	-	81.47	-
Huang et al. (2015) Bi-LSTM+CRF	No	-	-	84.26	-
Yan et al. (2016) Win-BiLSTM (English), FF (German) (Many fets)	Yes	-	-	88.91	<b>76.12</b>
Collobert et al. (2011) Conv-CRF (SENNNA+Gazetteer)	Yes	-	-	89.59	-
Huang et al. (2015) Bi-LSTM+CRF+ (SENNNA+Gazetteer)	Yes	-	-	<b>90.10</b>	-
Feature-inferring neural network character models					
Gillick et al. (2015) – BTS	No	<b>82.95</b>	<b>82.84</b>	<b>86.50</b>	<b>76.22</b>
Kuru et al. (2016) CharNER	No	82.18	79.36	84.52	70.12
Feature-inferring neural network word + character models					
Yang et al. (2017)	Yes	85.77	<b>85.19</b>	91.26	-
Luo (2015)	Yes	-	-	91.20	-
Chiu and Nichols (2015)	Yes	-	-	<b>91.62</b>	-
Ma and Hovy (2016)	No	-	-	91.21	-
Santos and Guimaraes (2015)	No	82.21	-	-	-
Lample et al. (2016)	No	85.75	81.74	90.94	<b>78.76</b>
Bharadwaj et al. (2016)	Yes	<b>85.81</b>	-	-	-
Dernoncourt et al. (2017)	No	-	-	90.5	-
Feature-inferring neural network word + character + affix models					
Re-implementation of Lample et al. (2016) (100 Epochs)	No	85.34	85.27	90.24	78.44
Yadav et al. (2018)(100 Epochs)	No	86.92	87.50	90.69	78.56
Yadav et al. (2018) (150 Epochs)	No	<b>87.26</b>	<b>87.54</b>	90.86	<b>79.01</b>

## 1. Classical Approaches (rule based)

## 2. ML Approaches

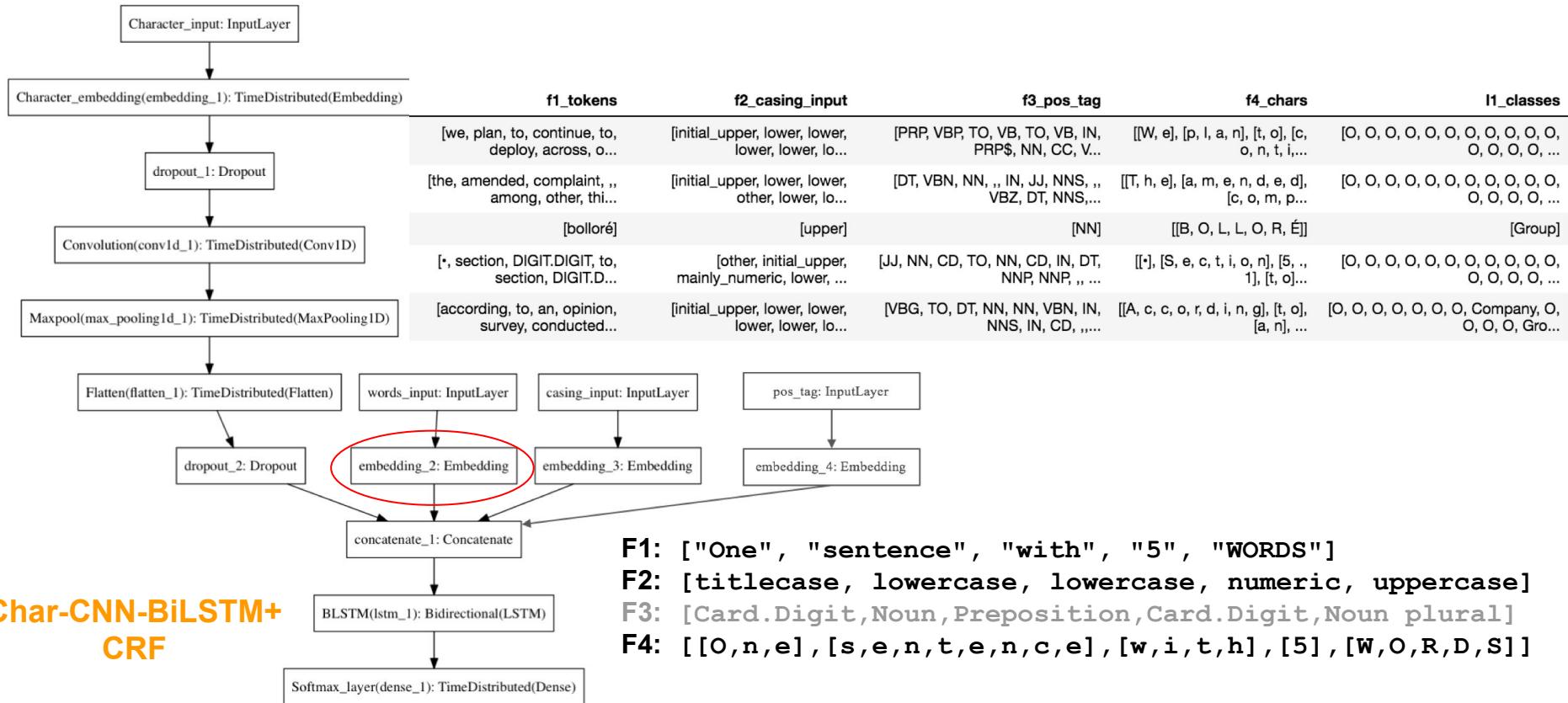
- Multi-class classification
- Conditional Random Field (CRF)

## 3. DL Approaches

- Bidirectional LSTM-CRF
- Bidirectional LSTM-CNNs
- Bidirectional LSTM-CNNS-CRF
- Pre-trained language models  
(Bert, Elmo)

## 4. Hybrid Approaches (DL + ML)

# NER-DL in Spark NLP



# NER-DL in Spark NLP

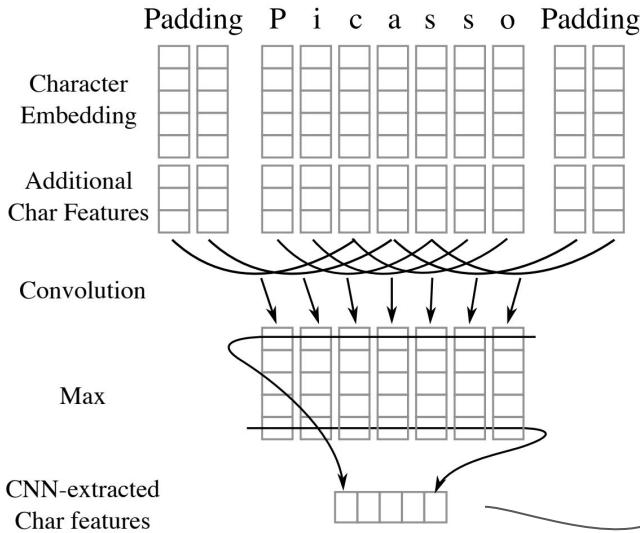


Figure 2: The convolutional neural network extracts character features from each word. The character embedding and (optionally) the character type feature vector are computed through lookup tables. Then, they are concatenated and passed into the CNN.

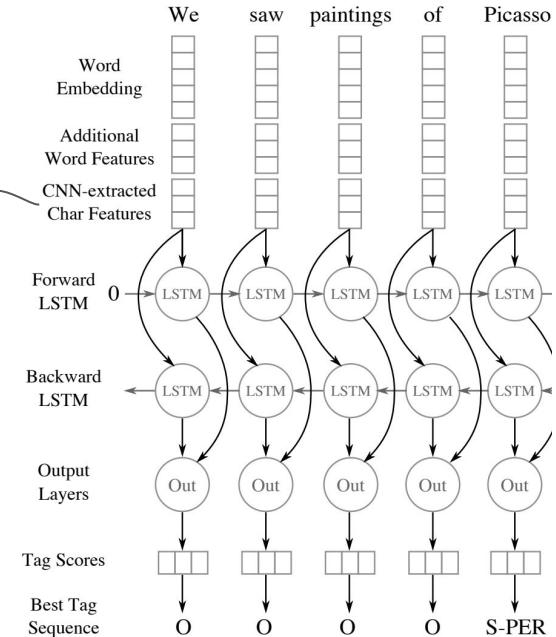


Figure 1: The (unrolled) BLSTM for tagging named entities. Multiple tables look up word-level feature vectors. The CNN (Figure 2) extracts a fixed length feature vector from character-level features. For each word, these vectors are concatenated and fed to the BLSTM network and then to the output layers (Figure 3).

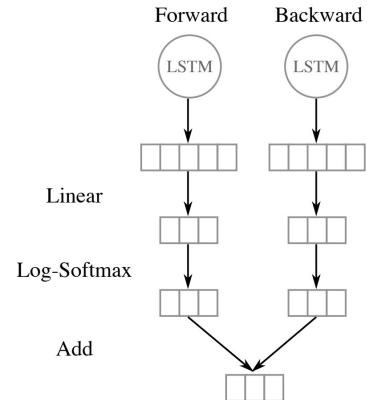


Figure 3: The output layers ("Out" in Figure 1) decode output into a score for each tag category.

**Char-CNN-BiLSTM**

# Clinical Named Entity Recognition

A 28-year-old female with a history of gestational diabetes mellitus diagnosed eight years prior to presentation and subsequent type two diabetes mellitus (T2DM), one prior episode of HTG-induced pancreatitis three years prior to presentation, associated with an acute hepatitis, and obesity with a body mass index (BMI) of 33.5 kg/m<sup>2</sup>, presented with a one-week history of polyuria, polydipsia, poor appetite, and vomiting. Two weeks prior to presentation, she was treated with a five-day course of amoxicillin for a respiratory tract infection. She was on metformin, glipizide, and dapagliflozin for T2DM and atorvastatin and gemfibrozil for HTG. She had been on dapagliflozin for six months at the time of presentation. Physical examination on presentation was significant for dry oral mucosa; significantly, her abdominal examination was benign with no tenderness, guarding, or rigidity. Pertinent laboratory findings on admission were: serum glucose 111 mg/dL, bicarbonate 18 mmol/L, anion gap 20, creatinine 0.4 mg/dL, triglycerides 508 mg/dL, total cholesterol 122 mg/dL, glycated hemoglobin (HbA1c) 10%, and venous pH 7.27. Serum lipase was normal at 43 U/L. Serum acetone levels could not be assessed as blood samples kept hemolyzing due to significant lipemia. The patient was initially admitted for starvation ketosis, as she reported poor oral intake for three days prior to admission. However, serum chemistry obtained six hours after presentation revealed her glucose was 186 mg/dL, the anion gap was still elevated at 21, serum bicarbonate was 16 mmol/L, triglyceride level peaked at 2050 mg/dL, and lipase was 52 U/L. The β-hydroxybutyrate level was obtained and found to be elevated at 5.29 mmol/L - the original sample was centrifuged and the chylomicron layer removed prior to analysis due to interference from turbidity caused by lipemia again.

## Clinical NER

Color codes: PROBLEM, TREATMENT, TEST,

The patient was prescribed 1 capsule of Advil for 5 days. He was seen by the endocrinology service and she was discharged on 40 units of insulin glargine at night, 12 units of insulin lispro with meals, and metformin 1000 mg two times a day. It was determined that all SGLT2 inhibitors should be discontinued indefinitely from 3 months.

## Posology NER

Color codes: FREQUENCY, DOSAGE, DURATION, DRUG, FORM, STRENGTH,

No findings in urinary system, skin color is normal, brain CT and cranial checks are clear. Swollen fingers and eyes. Extensive stage small cell lung cancer. Chemotherapy with carboplatin and etoposide. Left scapular pain status post CT scan of the thorax.

## Anatomy NER

Color codes: Organ, Organism\_subdivision, Organism\_substance, PathologicalFormation, Anatomical\_system,

A . Record date : 2093-01-13, David Hale, M.D., Name : Hendrickson, Ora MR. # 7194334  
Date : 01/13/93 PCP : Oliveira, 25 years-old, Record date : 2079-11-09. Cocke County  
Baptist Hospital, 0295 Keats Street

Color codes: STREET, DOCTOR, AGE, HOSPITAL, PATIENT, DATE, MEDICALRECORD,

## PHI NER

# 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.

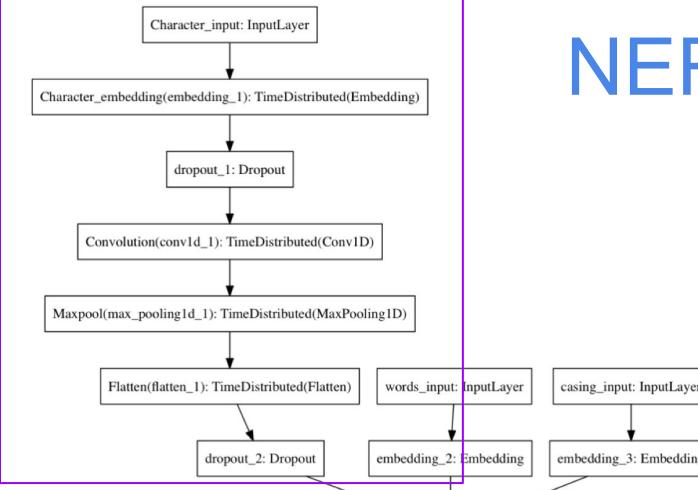
\* CoNLL: Conference on Computational Natural Language Learning

## BIO schema

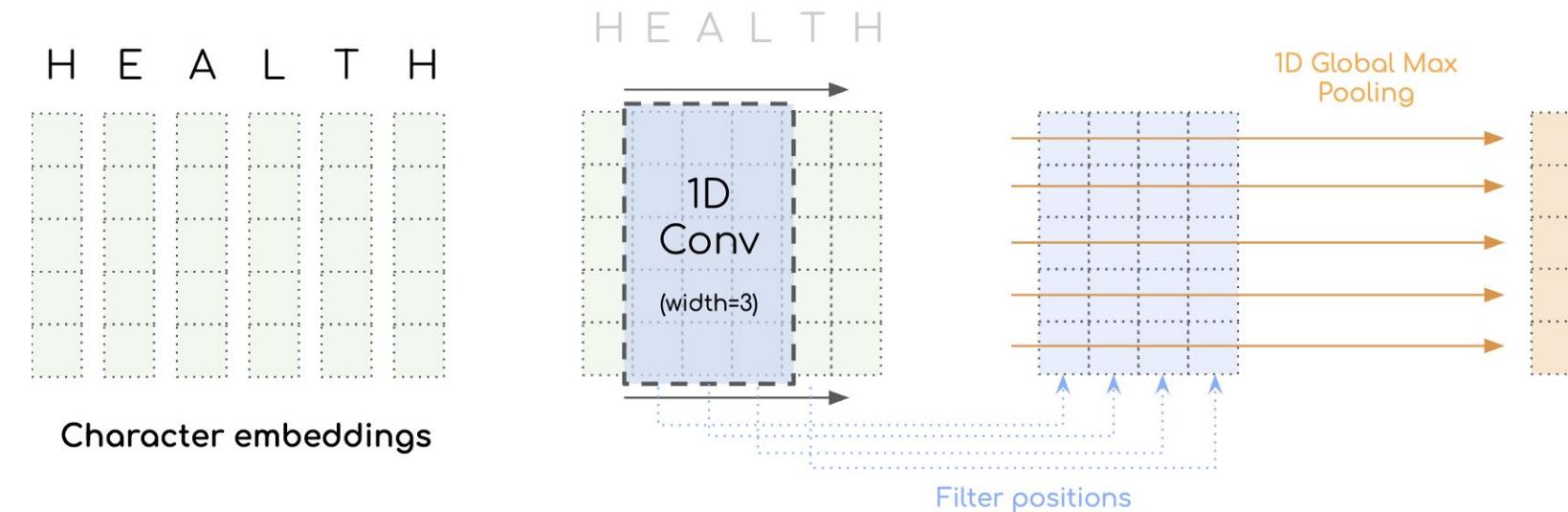
John	B-PER
Smith	I-PER
lives	O
in	O
New	B-LOC
York	I-LOC

John Smith ⇒ PERSON  
New York ⇒ LOCATION

# NER-DL in Spark NLP



Char-CNN process, e.g. on the world “HEALTH”



# NER-DL in Spark NLP

## Char-CNN-BiLSTM

	F1 : Tokens	F2 : Casing	F3 : POS	F4 : Char CNN	Labels
The					O
company					O
XYZ					Company
Private					Company
Limited					Company
works					O
in					O
the					O
health					Activity
sector					Activity
in					O
Europe					Location

# NER-DL in Spark NLP

## Classification

	The	company	XYZ	Private	Limited	works	in	the	health	sector	in	Europe
GROUND TRUTH	O	O	Company	Company	Company	O	O	O	Activity	Activity	O	Location
PREDICTION	O	O	O	Company	Company	O	Group	O	O	Activity	O	Location

Class	O	Company	Location	Activity	Group
O	91969	546	295	1069	251
Company	569	3084	69	43	129
Location	137	48	1677	1	4
Activity	735	28	0	1329	0
Group	98	95	8	0	1185

Recall	Precision	F1
97,7 %	98,4 %	98 %
79,2 %	81,1 %	80,2 %
89,8 %	81,8 %	85,6 %
63,5 %	54,4 %	58,6 %
85,5 %	75,5 %	80,2 %

# 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](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-roundner|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
```

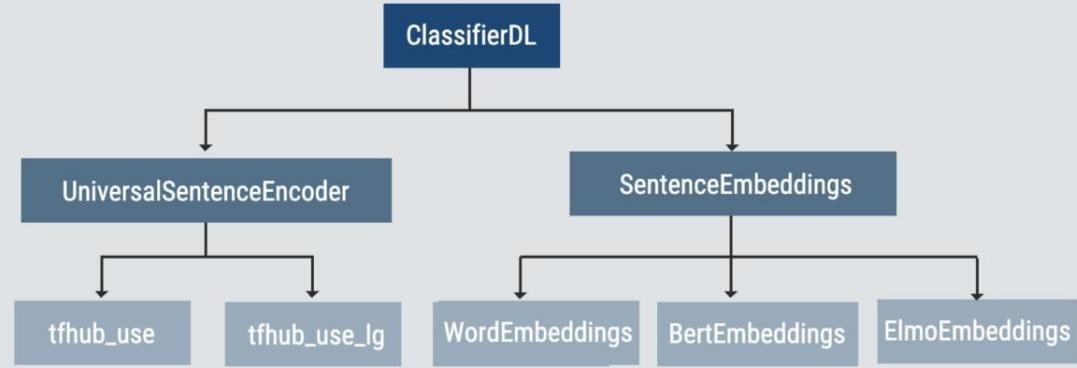
# Part - III

- ❖ 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

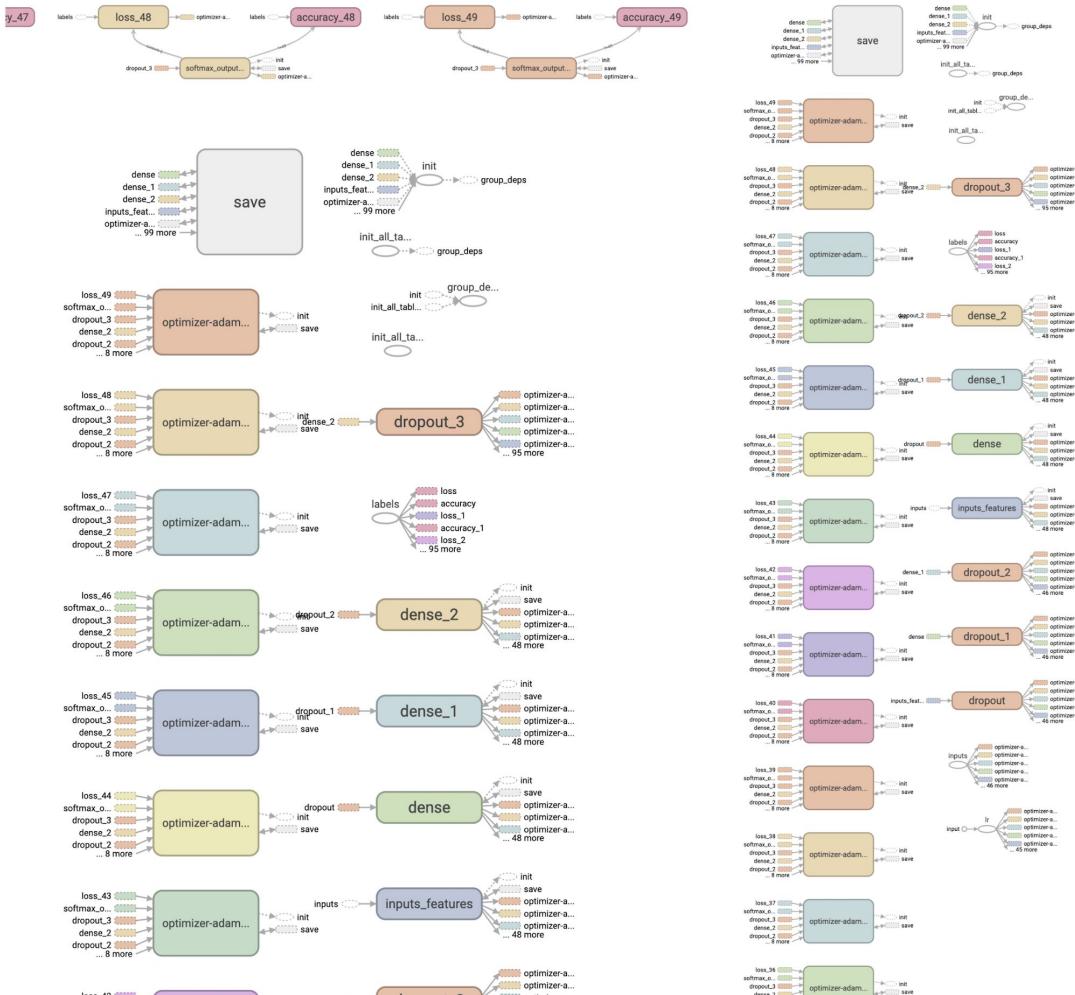


- 2 classes (positive/negative)
- 3 classes (0, 1, 2)
- 4 classes (Sports, Business, etc.)
- 5 classes (1.0, 2.0, 3.0, 4.0, 5.0)
- 50 classes!
- 100 dimensions
- 200 dimensions
- 300 dimensions
- 512 dimensions
- 768 dimensions
- 1024 dimensions
- tfhub\_use
- tfhub\_use\_lg
- glove\_6B\_100
- glove\_6B\_300
- 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](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)

Named Entity Recognition with Bert in Spark NLP

Text Classification in Spark NLP with Bert and Universal Sentence Encoders

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/why-spark-nlp-is-the-most-widely-used-nlp-library-enterprise/>

<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-enterprise/>

<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>



NOW ANNOUNCING

# NLP SUMMIT

Applied Natural  
Language Processing

Boston, Oct 27-28 | San Francisco, Nov 17-18



**John Snow LABS**

**Spark NLP  
for Data Scientists**

**July 28-29, 2020**

**Spark NLP  
for Healthcare Data Scientist**

**August 4-5, 2020**

**Veysel Kocaman**

Sr. Data Scientist

[veysel@johnsnowlabs.com](mailto: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)



NOW ANNOUNCING

# NLP SUMMIT

Applied Natural  
Language Processing

Boston, Oct 27-28 | San Francisco, Nov 17-18



# Spark NLP for Data Scientists

Session start at 13:30

Veysel Kocaman

Sr. Data Scientist

[veysel@johnsnowlabs.com](mailto:veysel@johnsnowlabs.com)