NLP SUMMIT

Presented by John Snow LABS

Spark NLP for Data Scientists

October 13-14, 2020

Veysel Kocaman

Lead Data Scientist

John Snow Labs

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	- Keyword Extraction (YAKE) - 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

[How to set up Google Colab]

BOOKMARK:

https://nlp.johnsnowlabs.com/models

https://nlp.johnsnowlabs.com/docs/en/quickstart

spark-nlp.slack.com

Open in Colab

```
import os

# Install java
! apt-get update -qq
! 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
! pip install --ignore-installed -q spark-nlp==2.6.3-rc1
```

master · spark-nlp-workshop / tutorials / Certification_Trainings / Public / · Go to file · Add file

vkocaman notebooks updated · 78e1d1b · 19 hours ago · History

..		
data	conf dataset added	5 days ago
databricks_notebooks	classifierdl notebook on databricks updated for debugging	20 days ago
utils	conll eval and ner log parser scripts added	3 months ago
1.SparkNLP_Basics.ipynb	notebooks updated	19 hours ago
2.Text_Preprocessing_with_SparkNLP_Annotators_Tr...	notebooks updated	19 hours ago
3.SparkNLP_Pretrained_Models.ipynb	removed outdated notebooks	last month
4.1_NerDL_Graph.ipynb	graph folder fro NerDL	26 days ago
4.NERDL_Training.ipynb	new updates	6 days ago
5.1_Text_classification_examples_in_SparkML_Spark...	updated for Spark 2.3	2 months ago
5.Text_Classification_with_ClassifierDL.ipynb	bert models for cassifier added	5 days ago
6.Playground_DataFrames.ipynb	updated for Spark 2.3	2 months ago
7.Context_Spell_Checker.ipynb	new examples for spell cheker	2 months ago
8.Keyword_Extraction_YAKE.ipynb	YAKE moved to public and RE updated with new models	last month
9.SentenceDetectorDL.ipynb	SDDL added	12 days ago
Spark NLP Training - Public - July 2020.pdf	july slides added	3 months ago
Spark NLP Training-Public-April 2020.pdf	sldies updated	6 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

CIO Review ARTIFICIAL INTELLIGENCE SOLUTION PROVIDER OF THE YEAR - 2018

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

AI PLATFORM OF THE YEAR

CIO APPLICATIONS 2019

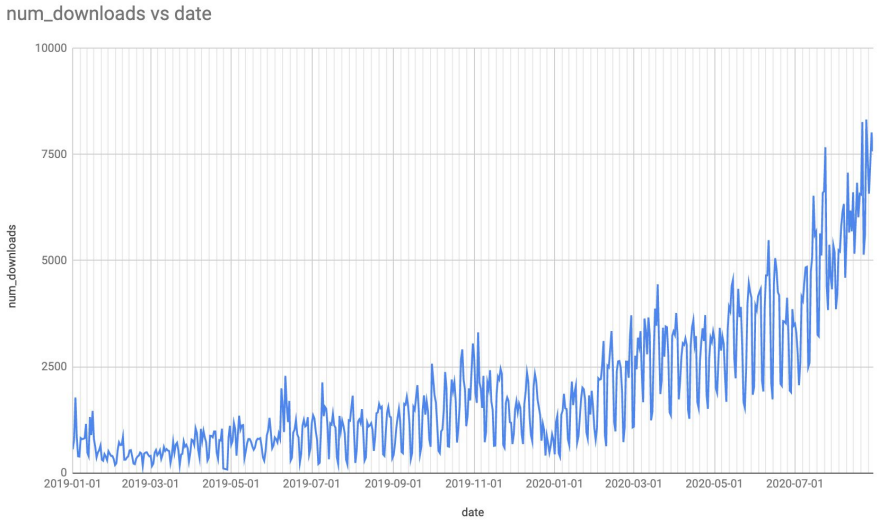
"John Snow Labs wows in both proven customer success and verifiable state-of-the-art technology – making it a natural winner of the highly competitive 2019
AI Platform of the Year Award."

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

Introducing Spark NLP

Daily ~ 10K
Monthly ~ 250K

PyPI link	https://pypi.org/project/spark-nlp
Total downloads	1,471,785
Total downloads - 30 days	245,960
Total downloads - 7 days	76,762



- 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 (3 yrs in a row)
  - 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.)

https://medium.com/spark-nlp/introduction-to-spark-nlp-foundations-and-basic-components-part-i-c83b7629ed59

Spark NLP

- 73 total releases
- Release every two weeks for the past 3 years
- A single unified library for all your NLP/NLU need
- Active community on Slack and GitHub

NLP Feature	Spark NLP	spaCy	NLTK	CoreNLP	Hugging Face
Tokenization	**Yes**	Yes	Yes	Yes	Yes
Sentence segmentation	**Yes**	Yes	Yes	Yes	**No**
Steeming	**Yes**	Yes	Yes	Yes	**No**
Lemmatization	**Yes**	Yes	Yes	Yes	**No**
POS tagging	**Yes**	Yes	Yes	Yes	**No**
Entity recognition	**Yes**	Yes	Yes	Yes	Yes
Dep parser	**Yes**	Yes	Yes	Yes	**No**
Text matcher	**Yes**	Yes	**No**	**No**	**No**
Date matcher	**Yes**	**No**	**No**	**No**	**No**
Sentiment detector	**Yes**	**No**	Yes	Yes	Yes
Text classification	**Yes**	Yes	Yes	**No**	Yes
Spell checker	**Yes**	**No**	**No**	**No**	**No**
Language detector	**Yes**	**No**	**No**	**No**	**No**
Keyword extraction	**Yes**	**No**	**No**	**No**	**No**
Pretrained models	**Yes**	Yes	Yes	Yes	Yes
Trainable models	**Yes**	Yes	Yes	Yes	Yes

TRUSTED BY


Microsoft


Google


amazon


intel


IBM


databricks


verizon


indeed


CapitalOne


viacom


McKESSON


MERCK


Roche


selectdata


UiPath
Reboot Work.


ASCO
CANCER·LINQ
DISCOVERY


cnrs

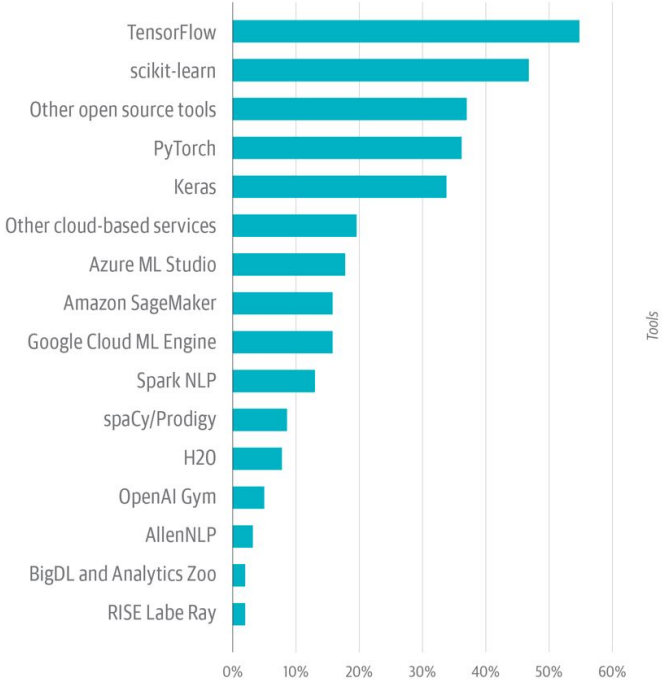

Imperial College
London


Georgia
Tech

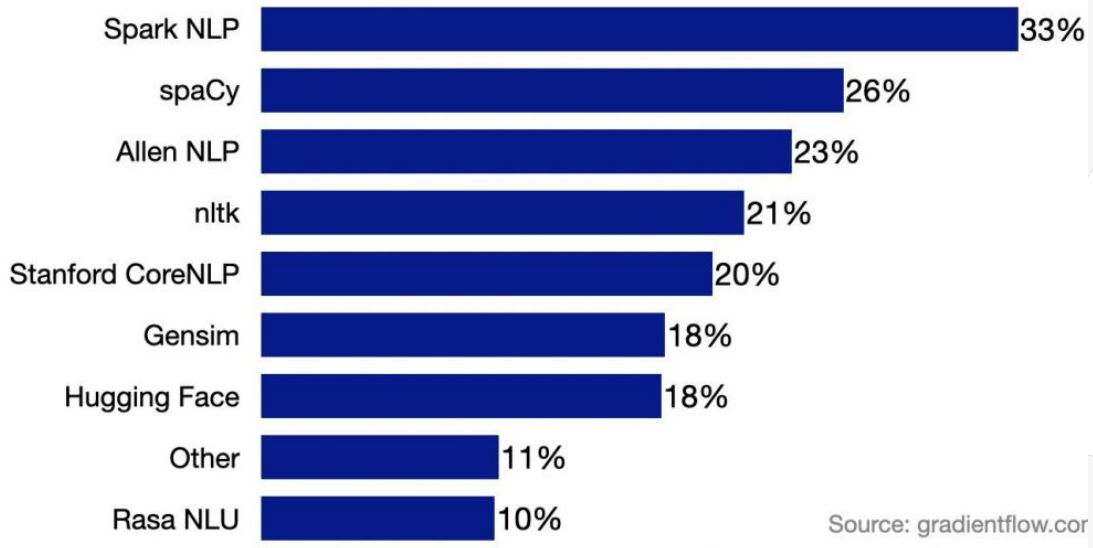

STANFORD
UNIVERSITY

Spark NLP in Industry

Which of the following AI tools do you use?



Which NLP libraries does your organization use?



NLP Industry Survey by Gradient Flow,
an independent data science research & insights company, September 2020

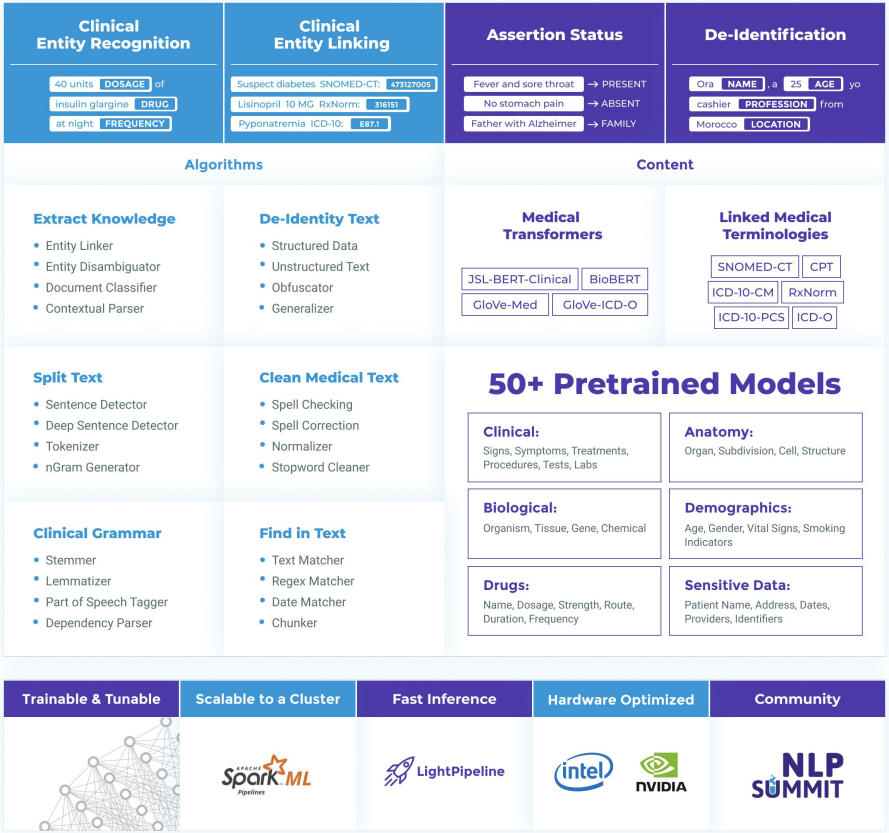
OFFICIALLY SUPPORTED RUNTIMES

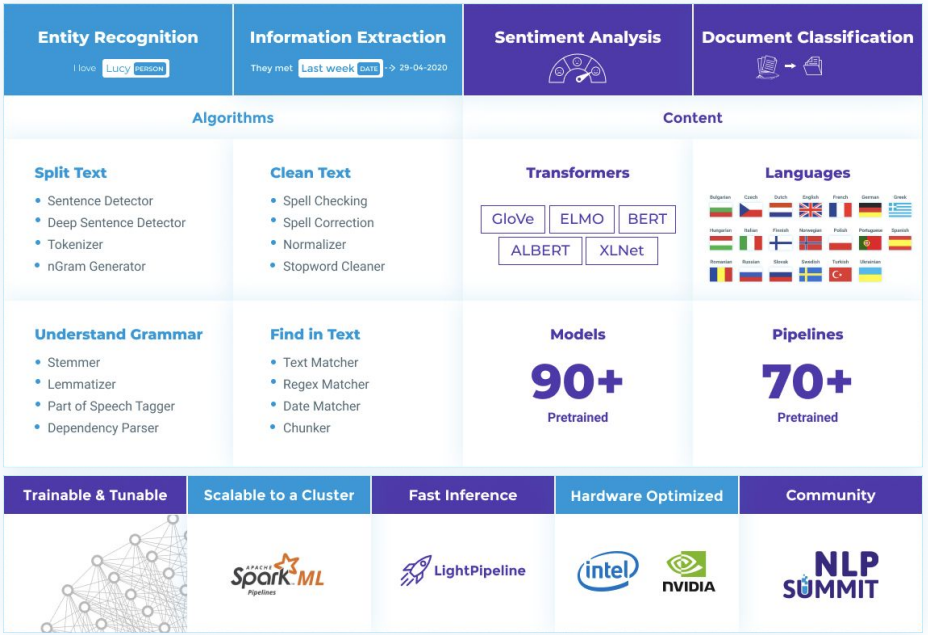
Spark NLP: Apache License 2.0

```
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)
```

- Tokenization
- Sentence Detector
- Stop Words Removal
- Normalizer
- Stemmer
- Lemmatizer
- NGrams
- Regex Matching
- Text Matching
- Chunking
- Date Matcher
- **Part-of-speech** tagging
- **Dependency** parsing
- **Sentiment** Detection (ML models)
- **Spell Checker** (ML and DL models)
- Word **Embeddings**
- **BERT** Embeddings
- **ELMO** Embeddings
- **ALBERT** Embeddings
- **XLNet** Embeddings
- **Universal Sentence Encoder**
- **BERT** Sentence Embeddings
- Sentence Embeddings
- Chunk Embeddings
- Unsupervised **keywords extraction**
- **Language Detection** & Identification
- **Multi-class** Text Classification
- **Multi-label** Text Classification
- Multi-class **Sentiment Analysis**
- **Named entity recognition**
- Easy **TensorFlow** integration
- Full integration with Spark ML functions
- **+250 pre-trained models** in 46 languages!
- **+90 pre-trained pipelines** in 13 languages!

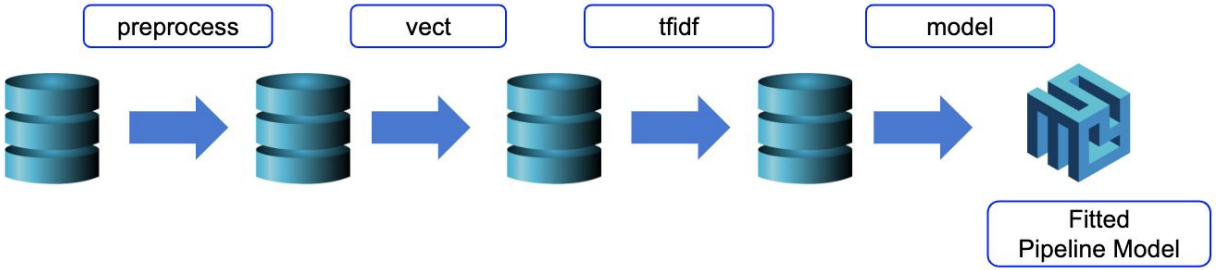
Spark NLP Modules (Enterprise and Public)



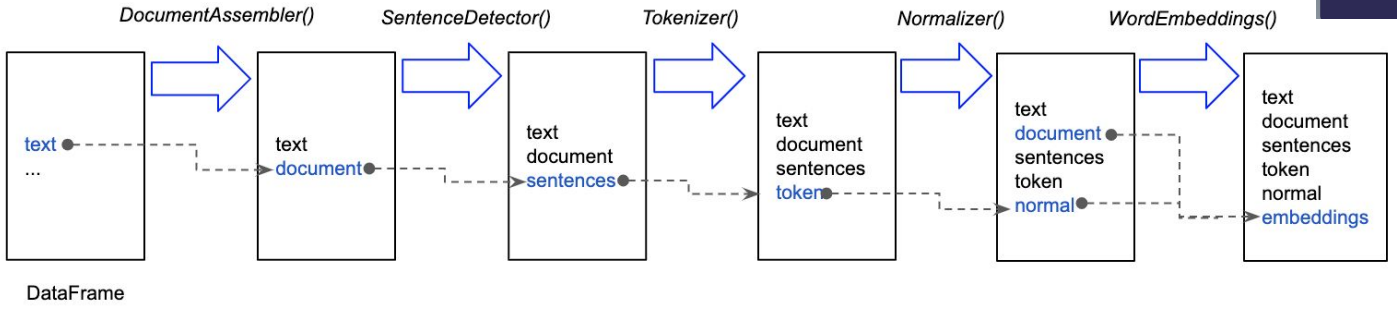


Introducing Spark NLP

Pipeline of annotators




```
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)
```



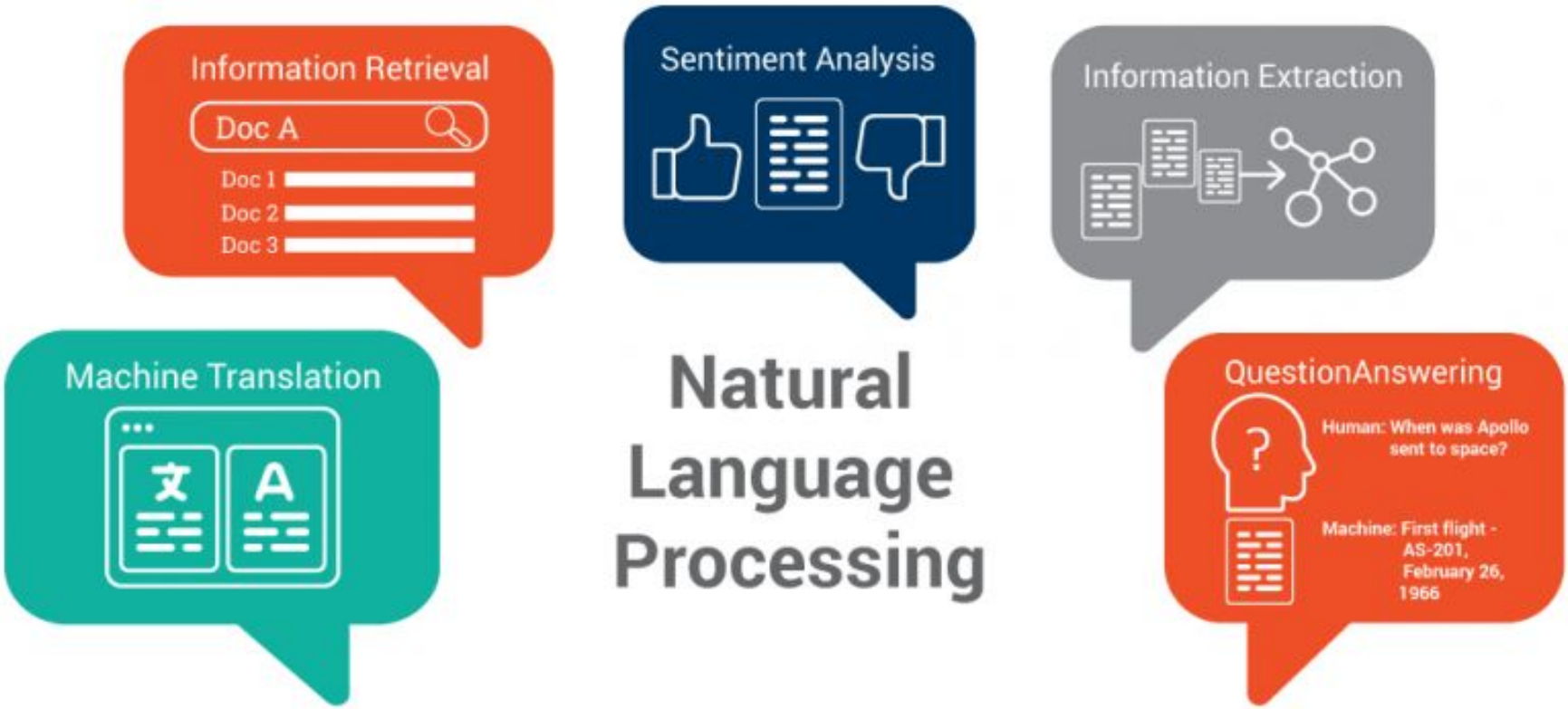
Introducing Spark NLP

Spark is like a locomotive racing a bicycle. The bike will win if the load is light, it is quicker to accelerate and more agile, but with a heavy load the locomotive might take a while to get up to speed, but it's going to be faster in the end.

Faster inference

```
from sparknlp.base import LightPipeline
LightPipeline(someTrainedPipeline).annotate(someStringOrArray)
```

LightPipelines are Spark ML pipelines converted into a single machine but multithreaded task, becoming more than 10x times faster for smaller amounts of data (small is relative, but 50k sentences is roughly a good maximum).


Information Retrieval
Doc A
Doc 1
Doc 2
Doc 3
Sentiment Analysis
Information Extraction
Machine Translation
文
A
Natural Language Processing
QuestionAnswering
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
studying	-ing	study
niñas	-as	niñ
niñez	-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

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

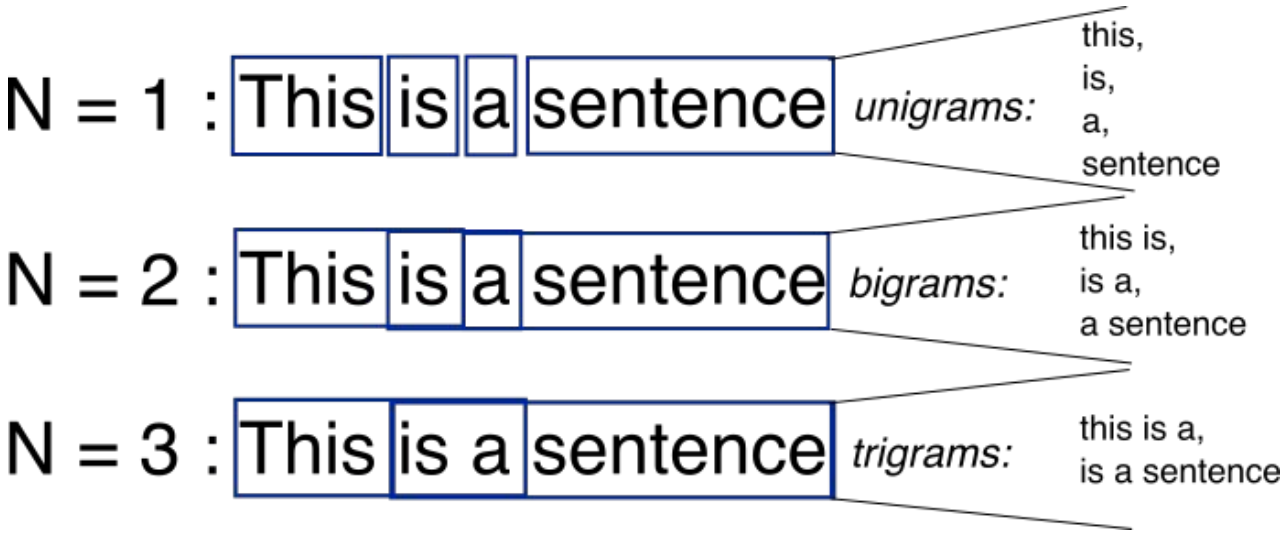
NORMALIZATION

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:

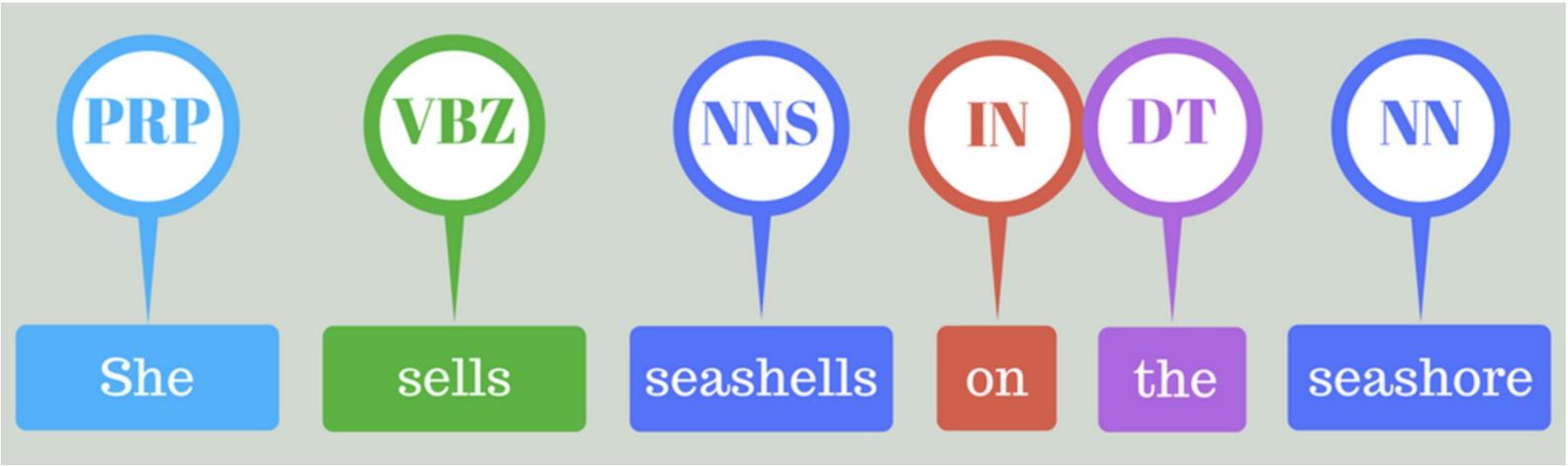
N-gram Tokenization



- 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

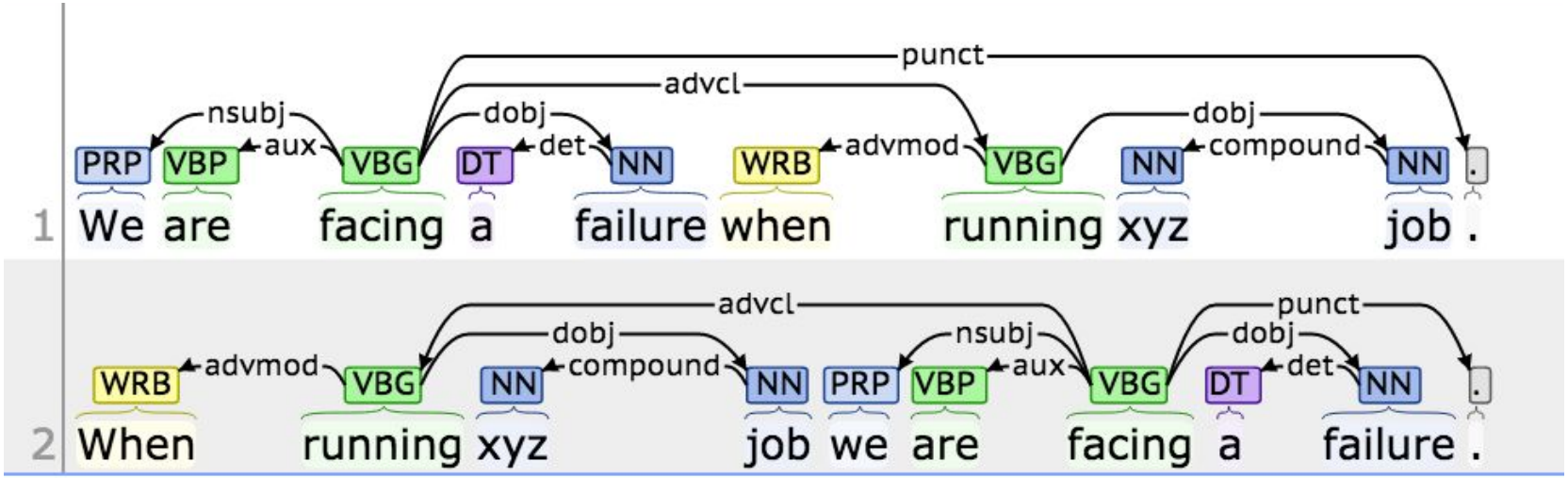
Often useful for recognizing named entities or word relationships.



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.

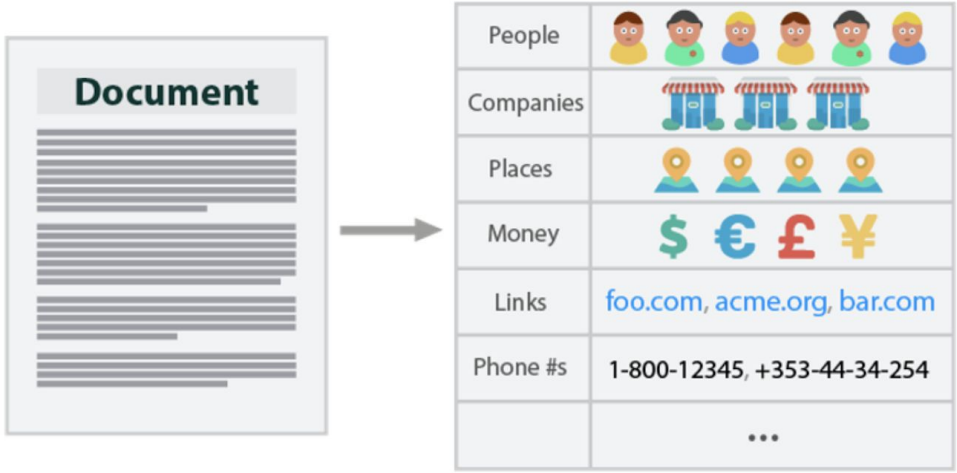
DEPENDENCY PARSING

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



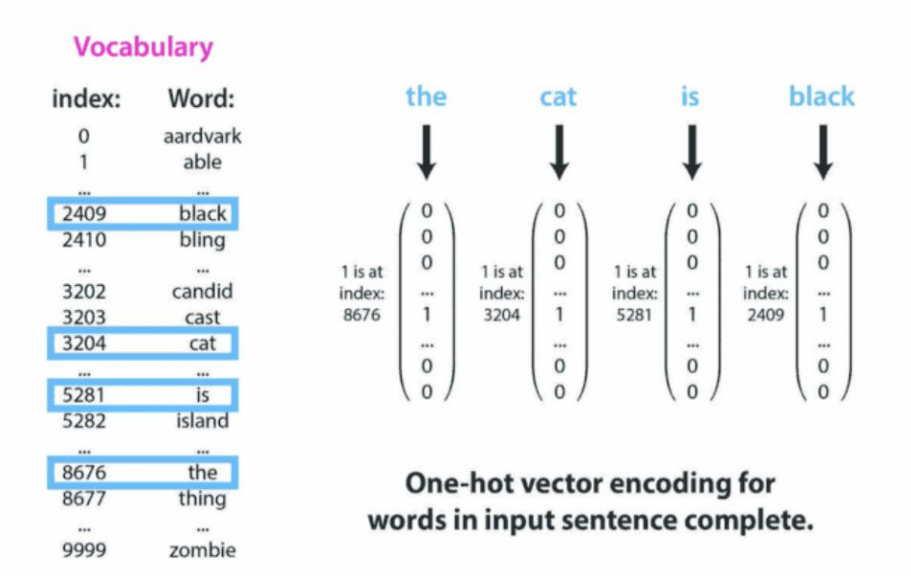
Named Entity Recognition (NER)

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.

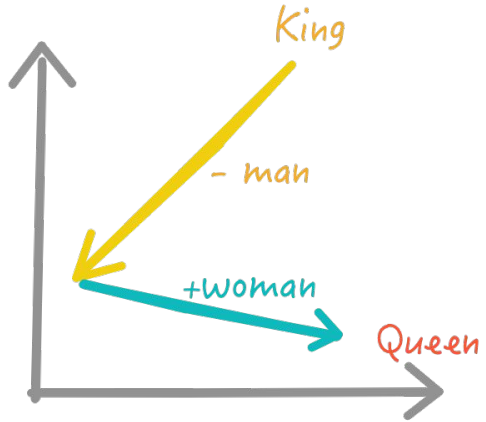
Word & Sentence Embeddings



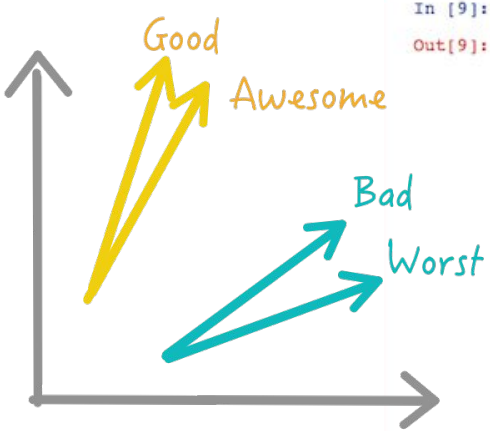

```
In [9]: doc[3].vector
Out[9]: array([ 0.037103 , -0.31259  , -0.17857  ,  0.30001  ,  0.078154 ,
                0.17958  ,  0.12048  , -0.11879  , -0.20601  ,  1.2849   ,
               -0.20409  ,  0.80613  ,  0.34344  , -0.19191  , -0.084511 ,
                0.17339  ,  0.042483 ,  2.0282   , -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  ,
```

- 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.

Word & Sentence Embeddings



a) Learns Analogy



b) Similar Words have same angles

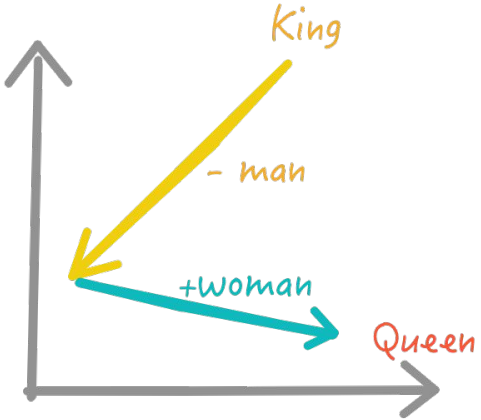
```
In [9]: doc[3].vector
Out[9]: array([ 0.037103 , -0.31259  , -0.17857  ,  0.30001  ,  0.078154 ,
                0.17958  ,  0.12048  , -0.11879  , -0.20601  ,  1.2849   ,
               -0.20409  ,  0.80613  ,  0.34344  , -0.19191  , -0.084511 ,
                0.17339  ,  0.042483 ,  2.0282   , -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.

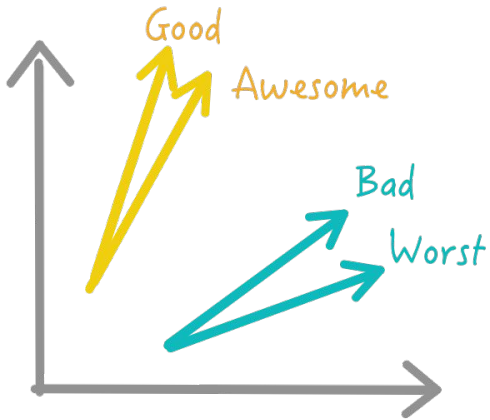
Word & Sentence Embeddings

Glove (100, 200, 300)	ELMO (512, 1024)	BERT (768d)	Universal Sentence Encoders (512)

Albert (768, 1024, 2048, 4096)	XLNet (768, 1024)	Electra (768)	Bert Sentence Embeddings (768)



a) Learns Analogy

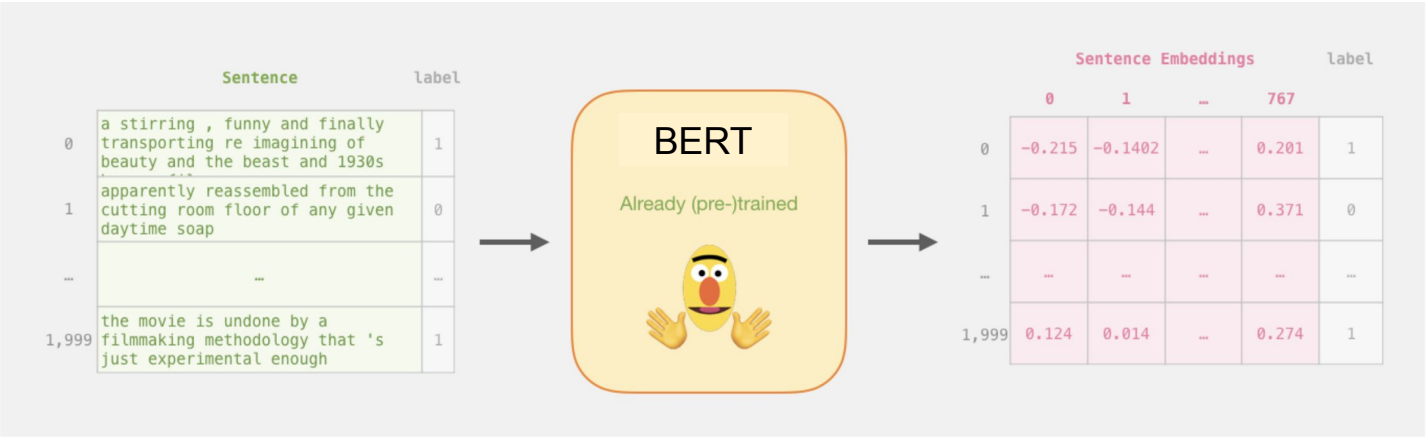


b) Similar Words have same angles

- 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.
- Elmo and Bert-family embeddings are context-aware.

Text Classification with Word & Sentence Embeddings

Glove (100, 200, 300)	ELMO (512, 1024)	BERT (768d)	Universal Sentence Encoders (512)
Albert (768, 1024, 2048, 4096)	XLNet (768, 1024)	Electra (768)	Bert Sentence Embeddings (768)



Model	Name
WordEmbeddingsModel (GloVe)	`glove_100d`
BertEmbeddings	`electra_small_uncased`
BertEmbeddings	`electra_base_uncased`
BertEmbeddings	`electra_large_uncased`
BertEmbeddings	`bert_base_uncased`
BertEmbeddings	`bert_base_cased`
BertEmbeddings	`bert_large_uncased`
BertEmbeddings	`bert_large_cased`
BertEmbeddings	`biobert_pubmed_base_cased`
BertEmbeddings	`biobert_pubmed_large_cased`
BertEmbeddings	`biobert_pmc_base_cased`
BertEmbeddings	`biobert_pubmed_pmc_base_cased`
BertEmbeddings	`biobert_clinical_base_cased`
BertEmbeddings	`biobert_discharge_base_cased`
BertEmbeddings	`covidbert_large_uncased`
BertEmbeddings	`small_bert_L2_128`
BertEmbeddings	`small_bert_L4_128`
BertEmbeddings	`small_bert_L6_128`
BertEmbeddings	`small_bert_L8_128`
BertEmbeddings	`small_bert_L10_128`
BertEmbeddings	`small_bert_L12_128`
BertEmbeddings	`small_bert_L2_256`
BertEmbeddings	`small_bert_L4_256`
BertEmbeddings	`small_bert_L6_256`
BertEmbeddings	`small_bert_L8_256`
BertEmbeddings	`small_bert_L10_256`
BertEmbeddings	`small_bert_L12_256`
BertEmbeddings	`small_bert_L2_512`
BertEmbeddings	`small_bert_L4_512`
BertEmbeddings	`small_bert_L6_512`
BertEmbeddings	`small_bert_L8_512`
BertEmbeddings	`small_bert_L10_512`
BertEmbeddings	`small_bert_L12_512`
BertEmbeddings	`small_bert_L2_768`
BertEmbeddings	`small_bert_L4_768`
BertEmbeddings	`small_bert_L6_768`
BertEmbeddings	`small_bert_L8_768`
BertEmbeddings	`small_bert_L10_768`
BertEmbeddings	`small_bert_L12_768`
BertEmbeddings	`elmo`
BertEmbeddings	`albert_base_uncased`
BertEmbeddings	`albert_large_uncased`
BertEmbeddings	`albert_xlarge_uncased`
BertEmbeddings	`albert_xxlarge_uncased`
BertEmbeddings	`xlnet_base_cased`
BertEmbeddings	`xlnet_large_cased`
BertEmbeddings	`tfhub_use`
BertEmbeddings	`tfhub_use_lg`
BertSentenceEmbeddings	`sent_small_bert_L6_128`
BertSentenceEmbeddings	`sent_small_bert_L8_128`
BertSentenceEmbeddings	`sent_small_bert_L10_128`
BertSentenceEmbeddings	`sent_small_bert_L12_128`
BertSentenceEmbeddings	`sent_small_bert_L2_256`
BertSentenceEmbeddings	`sent_small_bert_L4_256`
BertSentenceEmbeddings	`sent_small_bert_L6_256`
BertSentenceEmbeddings	`sent_small_bert_L8_256`
BertSentenceEmbeddings	`sent_small_bert_L10_256`
BertSentenceEmbeddings	`sent_small_bert_L12_256`
BertSentenceEmbeddings	`sent_small_bert_L2_512`
BertSentenceEmbeddings	`sent_small_bert_L4_512`
BertSentenceEmbeddings	`sent_small_bert_L6_512`
BertSentenceEmbeddings	`sent_small_bert_L8_512`
BertSentenceEmbeddings	`sent_small_bert_L10_512`
BertSentenceEmbeddings	`sent_small_bert_L12_512`
BertSentenceEmbeddings	`sent_small_bert_L2_768`
BertSentenceEmbeddings	`sent_small_bert_L4_768`
BertSentenceEmbeddings	`sent_small_bert_L6_768`
BertSentenceEmbeddings	`sent_small_bert_L8_768`
BertSentenceEmbeddings	`sent_small_bert_L10_768`
BertSentenceEmbeddings	`sent_small_bert_L12_768`

Word & Sentence Embeddings

albert_base = 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 | 1024-embed-dim, 24-layer, 16-heads, 18M parameters

albert_xlarge = 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 | 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 | 24-layer, 1024-hidden, 16-heads

XLNet-Base = https://storage.googleapis.com/xlnet/released_models/cased_L-12_H-768_A-12.zip | 12-layer, 768-hidden, 12-heads.

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.

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

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/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

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

Veysel Kocaman
Sr. Data Scientist
veysel@johnsnowlabs.com

Part - II

- Named Entity Recognition (NER) in Spark NLP

Not Secure | nlpprogress.com/eng...

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 (Ghaddar 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

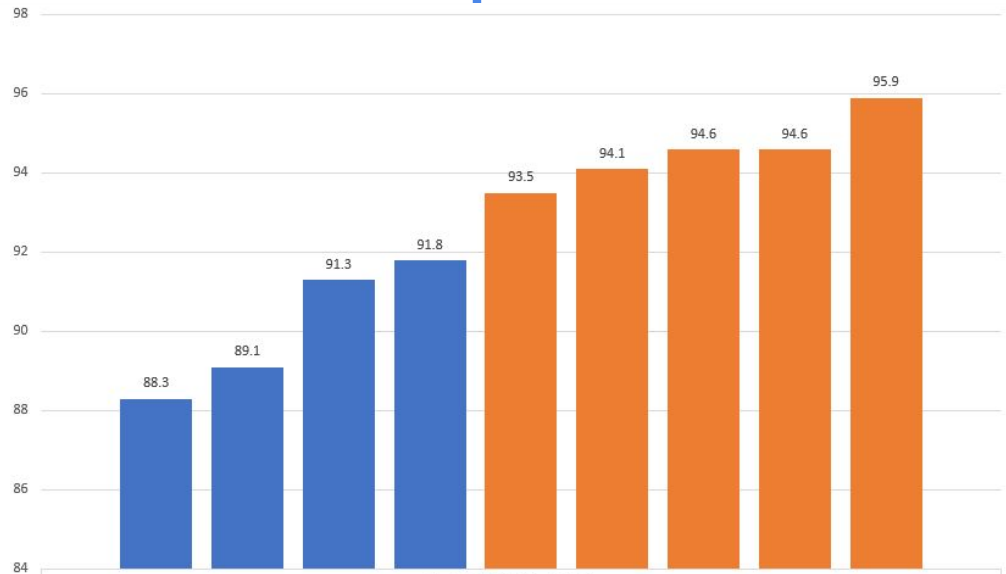
93.3 %
Test Set

Bert

NerDLApproach

John Snow LABS

NER-DL in Spark NLP



	CoNLL 2003 - F1 Dev
spaCy 2.2	88.3
spaCy 2.2 + en_trf_bertbaseuncased_lg	89.1
spaCy 2.2 + en_core_web_md	91.3
spaCy 2.2 + en_core_web_lg	91.8
Spark NLP + GloVe 100d	93.5
Spark NLP + GloVe 300d	94.1
Spark NLP + ELMO	94.6
Spark NLP + BERT Base Cased	94.6
Spark NLP + BERT Large Cased	95.9

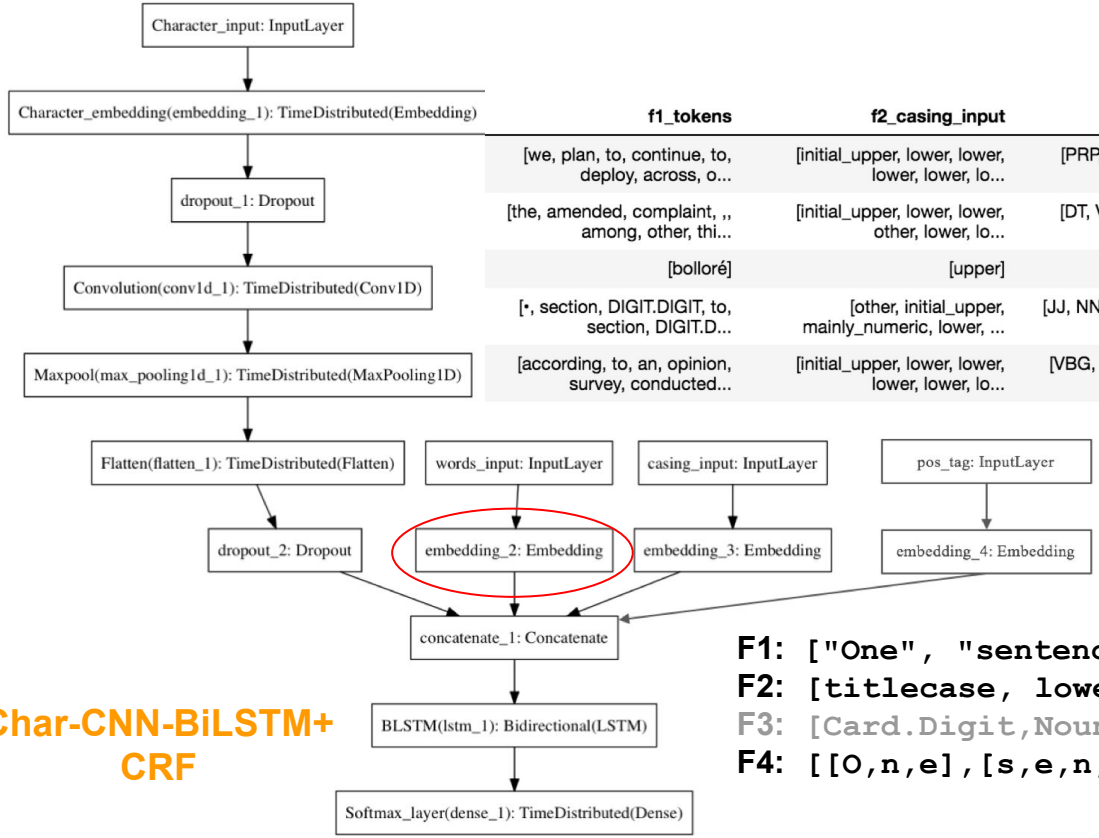
NER Systems

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	**84.16**	**85.04**	**91.36**	**76.42**
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	**76.12**
Collobert et al. (2011) Conv-CRF (SENNA+Gazetteer)	Yes	-	-	89.59	-
Huang et al. (2015) Bi-LSTM+CRF+ (SENNA+Gazetteer)	Yes	-	-	**90.10**	-
Feature-inferring neural network character models					
Gillick et al. (2015) – BTS	No	**82.95**	**82.84**	**86.50**	**76.22**
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	**85.19**	91.26	-
Luo (2015)	Yes	-	-	91.20	-
Chiu and Nichols (2015)	Yes	-	-	**91.62**	-
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	**78.76**
Bharadwaj et al. (2016)	Yes	**85.81**	-	-	-
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	**87.26**	**87.54**	90.86	**79.01**

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)

John Snow LABS

NER-DL in Spark NLP



f1_tokens	f2_casing_input	f3_pos_tag	f4_chars	l1_classes
[we, plan, to, continue, to, deploy, across, o...	[initial_upper, lower, lower, lower, lower, lo...	[PRP, VBP, TO, VB, TO, VB, IN, PRP$, NN, CC, V...	[[W, e], [p, l, a, n], [t, o], [c, o, n, t, i,...	[O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, ...
[the, amended, complaint, ,, among, other, thi...	[initial_upper, lower, lower, other, lower, lo...	[DT, VBN, NN, ,, IN, JJ, NNS, ,, VBZ, DT, NNS,...	[[T, h, e], [a, m, e, n, d, e, d], [c, o, m, p...	[O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, ...
[bolloré]	[upper]	[NN]	[[B, O, L, L, O, R, É]]	[Group]
[•, section, DIGIT.DIGIT, to, section, DIGIT.D...	[other, initial_upper, mainly_numeric, lower, ...	[JJ, NN, CD, TO, NN, CD, IN, DT, NNP, NNP, ,, ...	[[•], [S, e, c, t, i, o, n], [5, ., 1], [t, o]...	[O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, ...
[according, to, an, opinion, survey, conducted...	[initial_upper, lower, lower, lower, lower, lo...	[VBG, TO, DT, NN, NN, VBN, IN, NNS, IN, CD, ,,...	[[A, c, c, o, r, d, i, n, g], [t, o], [a, n], ...	[O, O, O, O, O, O, O, Company, O, O, O, Gro...

Char-CNN-BiLSTM+ CRF

F1: `["One", "sentence", "with", "5", "WORDS"]`
F2: `[titlecase, lowercase, lowercase, numeric, uppercase]`
F3: `[Card.Digit,Noun,Preposition,Card.Digit,Noun plural]`
F4: `[[O,n,e],[s,e,n,t,e,n,c,e],[w,i,t,h],[5],[W,O,R,D,S]]`

NER-DL in Spark NLP

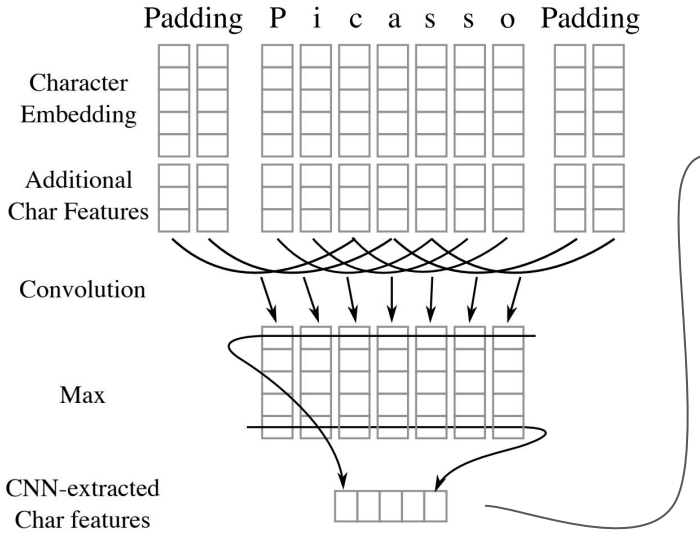


Figure 2: The convolutional neural network extracts character features from each word. The character embedding and (optionally) the character type feature vector are computed through lookup tables. Then, they are concatenated and passed into the CNN.

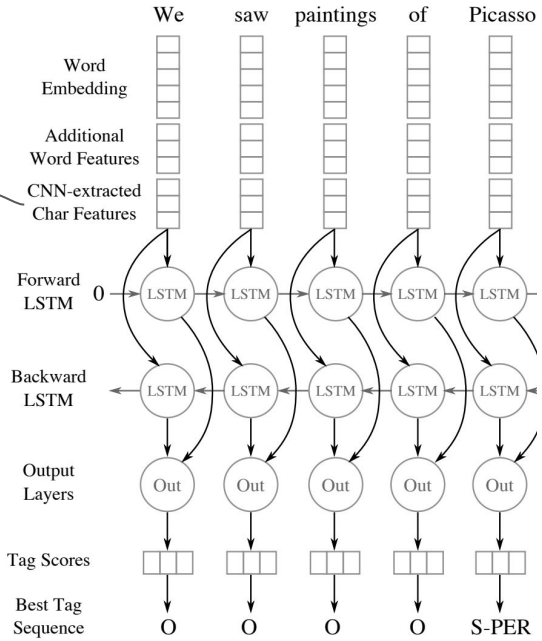


Figure 1: The (unrolled) BLSTM for tagging named entities. Multiple tables look up word-level feature vectors. The CNN (Figure 2) extracts a fixed length feature vector from character-level features. For each word, these vectors are concatenated and fed to the BLSTM network and then to the output layers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

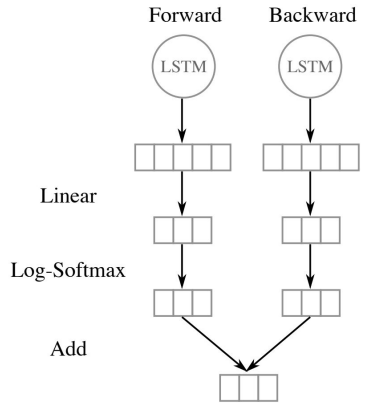


Figure 3: The output layers ("Out" in Figure 1) decode output into a score for each tag category.

Char-CNN-BiLSTM

John Snow LABS

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/m2 , 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 fro 3 months .

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

Posology NER

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 .

Color codes:Organ, Organism_subdivision, Organism_substance, Pathological_formation, Anatomical_system,

Anatomy NER

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

CoNNL2003 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

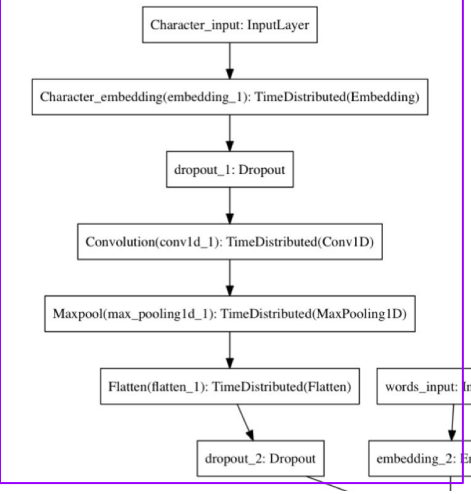
BIO schema

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

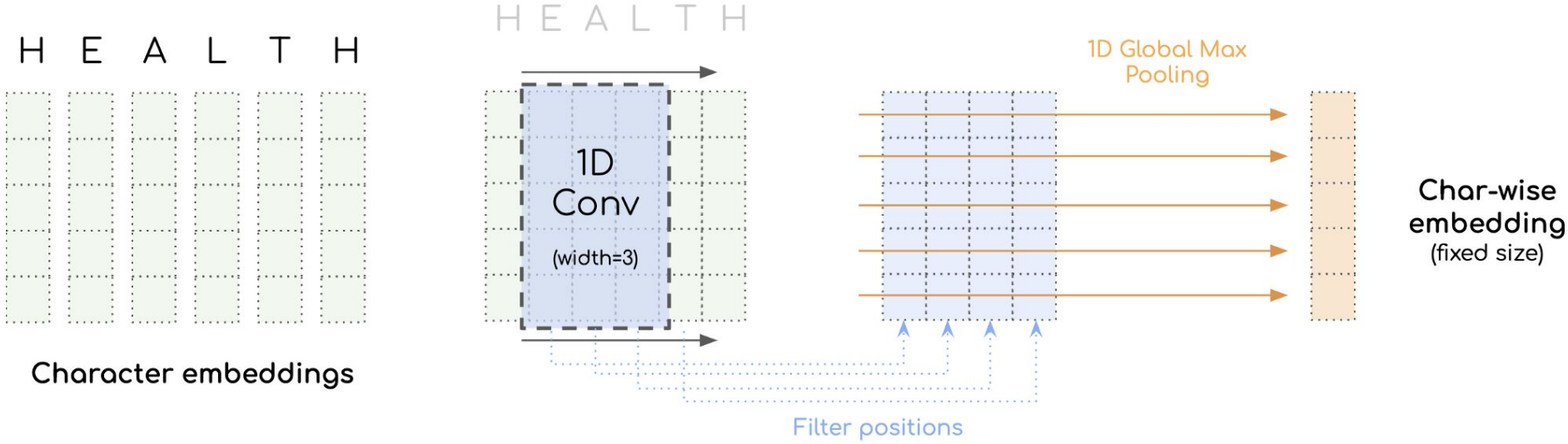
John Smith ⇒ PERSON
New York ⇒ LOCATION

John Snow LABS

NER-DL in Spark NLP

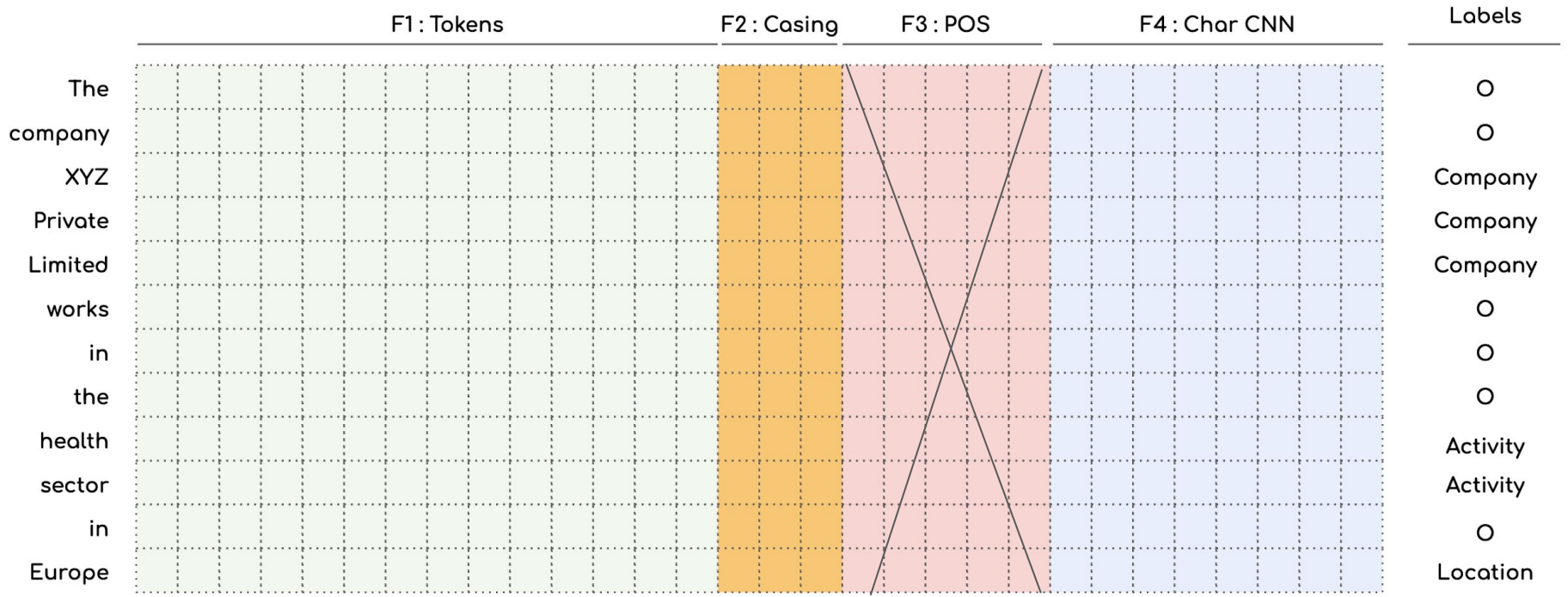


Char-CNN process, e.g. on the world “HEALTH”



NER-DL in Spark NLP

Char-CNN-BiLSTM



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

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

SentimentDL, ClassifierDL, and MultiClassifierDL

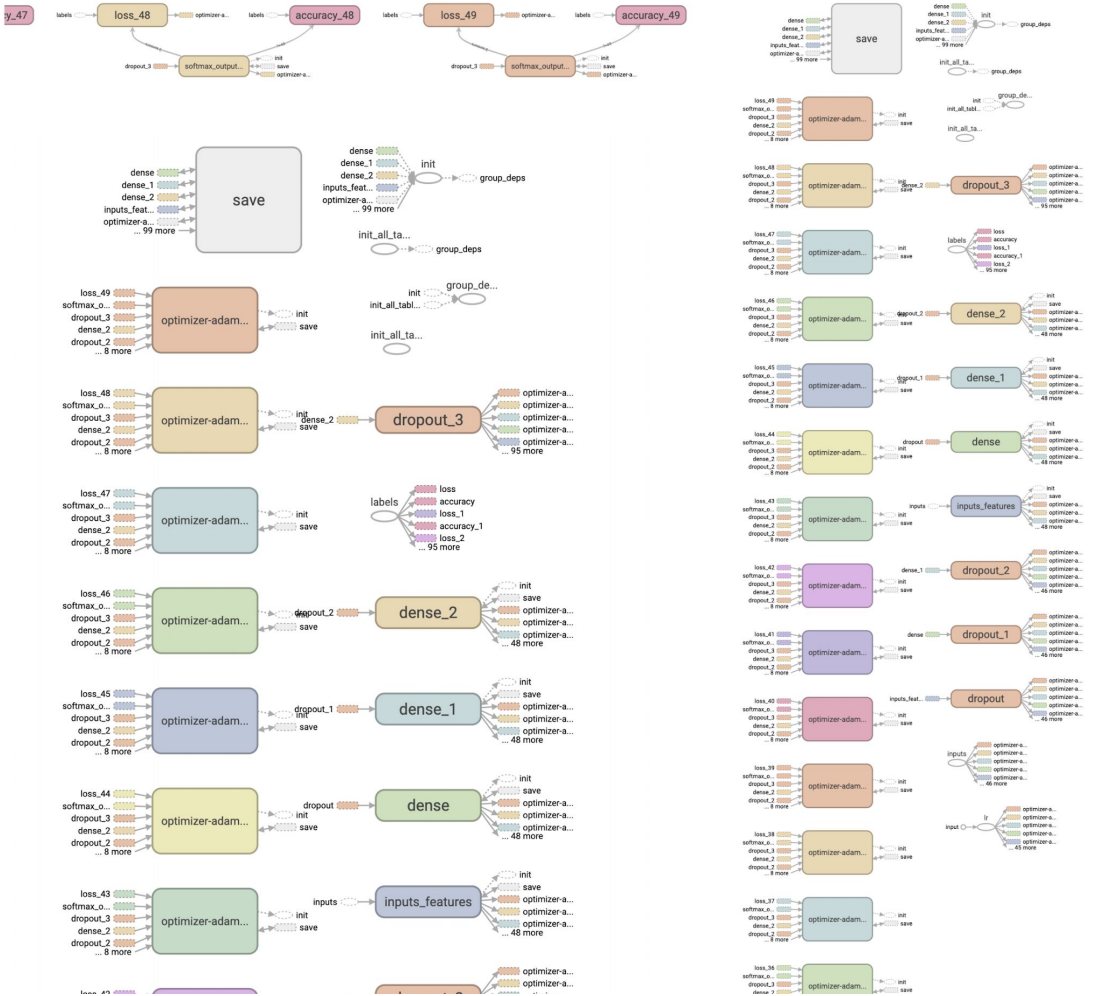
- BERT
- Small BERT
- BioBERT
- CovidBERT
- LaBSE
- ALBERT
- ELECTRA
- XLNet
- ELMO
- Universal Sentence Encoder
- GloVe

- 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)
- ... 100 classes!

- 100 dimensions
- 200 dimensions
- 128 dimensions
- 256 dimensions
- 300 dimensions
- 512 dimensions
- 768 dimensions
- 1024 dimensions

- tfhub_ues
- 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_uncased
- electra_small_uncased
- elmo
- ... 90+ Word & Sentence models

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 sentece detector here if we want to train on and get predictions for each senten
ce

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

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

use_clf_pipeline = Pipeline(
    stages = [
        document,
        use,
        classsifierdl
    ])
```

```
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_a0fddc9e970b.log
-rw-r--r-- 1 root root 976 Apr  7 16:59 ClassifierDLApproach_f222663dfb2c.log
```

In [ ]:
```
!cat ~/annotator_logs/ClassifierDLApproach_a0fddc9e970b.log
```

```
Training started - total epochs: 5 - learning rate: 0.005 - batch size: 64 - training examples: 12
0000
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/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

John Snow LABS


NOW ANNOUNCING
NLP
SUMMIT
Applied Natural
Language Processing
Boston, Oct 27-28 | San Francisco, Nov 17-18