1 line of Python code for 4000+ state-of-the-art NLP models in 200+ languages with the nlu library

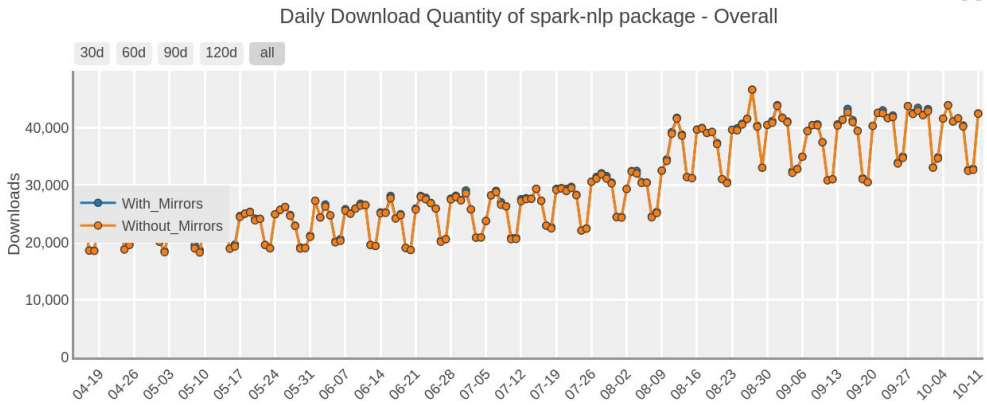
New York - NLP Meetup Feb 2022

pip install nlu

Christian Kasim Loan
Senior Data Scientist
christian@johnsnowlabs.com

Introducing Spark NLP

Total downloads	14,558,265
Total downloads - 30 days	1,371,781
Total downloads - 7 days	318,804

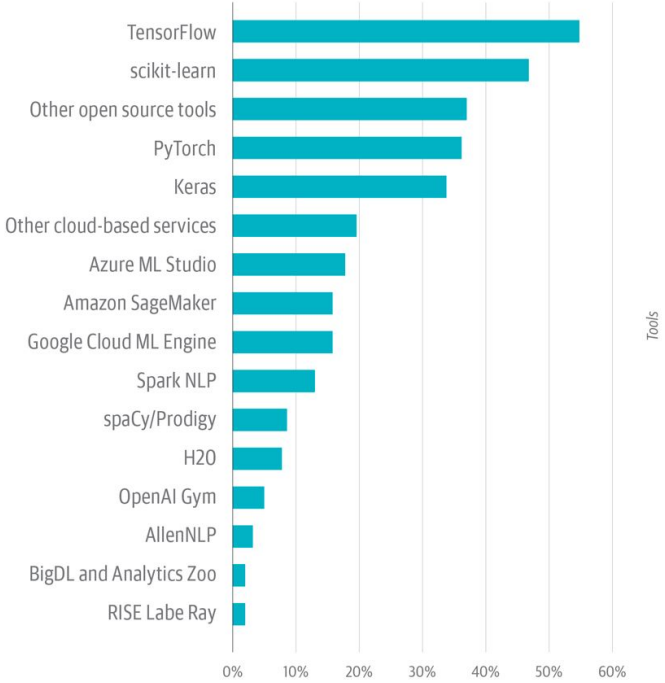


- Spark NLP is an open-source natural language processing library, built on top of Apache Spark and Spark ML. (initial release: Oct 2017)
  - A single unified solution for all your NLP needs
  - Take advantage of transfer learning and implementing the latest and greatest **SOTA** algorithms and models in NLP research
  - The most widely used NLP library in industry (5 yrs in a row)
  - Delivering a mission-critical, enterprise grade NLP library (used by multiple Fortune 500)
  - Full-time development team (30 new releases in 2021, 26 new releases in 2020, 30 new releases in 2019.)

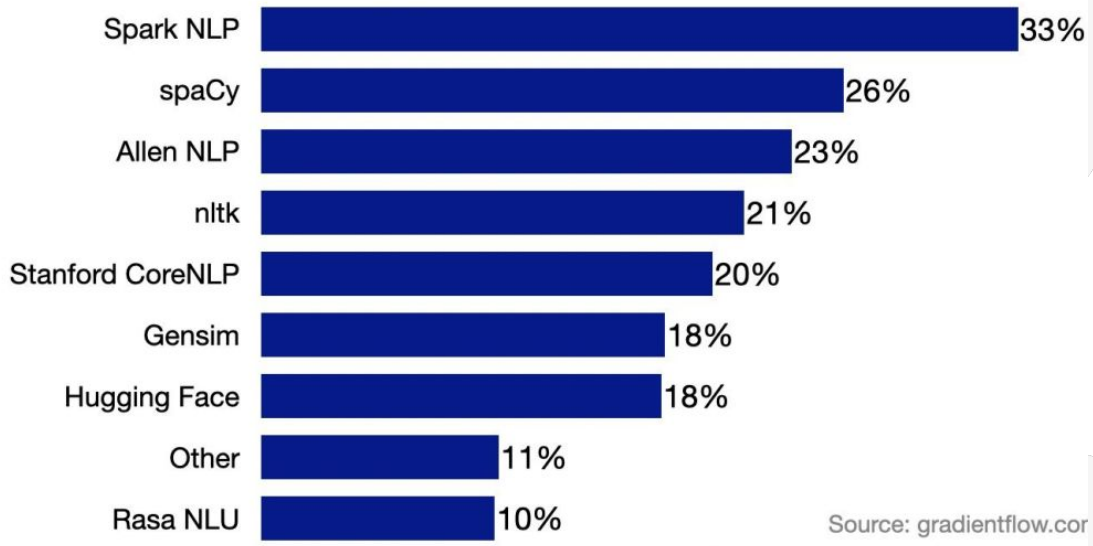
https://medium.com/spark-nlp/introduction-to-spark-nlp-foundations-and-basic-components-part-i-c83b7629ed59

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

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

Recognized by the Technology Experts

NLU & Spark NLP

- A single **unified** library for all your **NLP/NLU** needs
- 4000+ Models,
- 200+ Languages
- 1 Line of code
- 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
Question Answering	**Yes**	Yes	**No**	**No**	Yes
Text Style Transfer	**Yes**	No	**No**	**No**	Yes
Finance models	**Yes**	No	**No**	**No**	Yes
200+ Languages supported	**Yes**	Yes	**No**	**No**	Yes
Summarize Test	**Yes**	Yes	**No**	**No**	Yes
Text Generation (GPT2, T5)	**Yes**	Yes	**No**	**No**	Yes

Powerful NLU 1 Liners you learn today applicable in 200+ Languages

1. **Spell Checking**
2. **Binary Sentiment Classification**
3. **Multi Class Emotion Classification**
4. **Parts of Speech (POS)**
5. **Named Entity Recognition (NER)**
6. **Unsupervised Keyword Extraction (YAKE!)**
7. **Question Answering**
8. **Text Generation with GPT2**
9. **Translation between 200+ languages**
10. **Train a Multilingual Classifier for 100+ languages from just 1 input language**
11. **Visualize Embeddings** and leverage **Dozens of visualizations** with the NLU+**Streamlit** Data Science GUI Cockpit web app with **0 lines of code**
12. **How to use any of the 4000 + models in 1 line of code**

What is NLU?

- All of the 4000+ Spark NLP models in 1 line of code
- Recognize text in Images, PDFs, DOCX files in 1 line of code via NLU powered by Spark OCR
- Train models in 1 line of code
- Visualize with Streamlit or in Jupyter Notebook
- Automagically generates Spark NLP pipelines based on your request (Dependency Resolution)
- Works on Pandas/Spark/Modin Dataframes and returns same type of Dataframe

How does it work?

model= nlu.**load**(model)

- **Returns a nlu pipeline object**

model.**predict**(data)

- Returns a pandas DF

How does it work?

```
model = nlu.load('emotion')
```

- **Returns a nlu pipeline object**

```
model.predict('I love NLU!')
```

- Returns a pandas DF

EMOTION DETECTION

```
nlu.load('emotion').predict('I love NLU!')
```

sentence_embeddings	category_sentence	category_surprise	category_sadness	category_joy	category_fear	sentence	category	id
[0.027570432052016258, -0.052647676318883896, ...]	0	0.012899903	0.0015578865	0.9760173	0.0095249	I love NLU!	joy	1

NLU WORKS DIRECTLY ON TYPICAL PYTHON DATASETS

Strings

```
import nlu
nlu.load('sentiment').predict('This is just one string')
```

Lists

```
import nlu
nlu.load('sentiment').predict(['This is an arrray', ' Of strings!'])
```

Pandas data frame

```
import nlu
import pandas as pd
data = {"text": ['This day sucks', 'I love this day', 'I dont like Sami']}
text_df = pd.DataFrame(data)
nlu.load('sentiment').predict(text_df)
```

Pandas series

```
import nlu
import pandas as pd
data = {"text": ['This day sucks', 'I love this day', 'I dont like Sami']}
text_df = pd.DataFrame(data)
nlu.load('sentiment').predict(text_df['text'])
```

Spark
Data Frame

Ray
Data Frame

Dask
Data Frame

NAMED ENTITY RECOGNITION

```
nlu.load('ner').predict('Angela Merkel from Germany and the American Donald Trump dont share many opinions')
```

embeddings	ner_tag	entities
[[-0.563759982585907, 0.26958999037742615, 0.3...	PER	Angela Merkel
[[-0.563759982585907, 0.26958999037742615, 0.3...	LOC	Germany
[[-0.563759982585907, 0.26958999037742615, 0.3...	MISC	American
[[-0.563759982585907, 0.26958999037742615, 0.3...	PER	Donald Trump

CALCULATING EMBEDDINGS

```
#watch out for your RAM, this could kill your machine

nlu.load('bert elmo albert xlnet use glove').predict('Get all of them at once! Watch your RAM tough!')
```

token	glove_embeddings	albert_embeddings	xlnet_embeddings	bert_embeddings	elmo_embeddings	use_embeddings	id
Get	[0.1443299949169159, 0.4395099878311157, 0.583...]	[-0.41224443912506104, -0.4611411392688751, 1...]	[-0.003953204490244389, -1.5821468830108643, -...]	[-0.7420049905776978, -0.8647691011428833, 0.1...]	[0.04002974182367325, -0.43536433577537537, -0...]	[[-0.0019260947592556477, 0.009215019643306732...]	1
all	[-0.2182299941778183, 0.6919900178909302, 0.70...]	[1.1014549732208252, -0.43204769492149353, -0...]	[0.31148090958595276, -1.0986182689666748, 0.3...]	[-0.8933112025260925, 0.44822725653648376, -0...]	[0.17885173857212067, 0.045830272138118744, -0...]	[[-0.0019260947592556477, 0.009215019643306732...]	1
of	[-0.15289999544620514, -0.24278999865055084, 0...]	[1.1535910367965698, 0.28440719842910767, 0.60...]	[-1.403516411781311, 0.3108177185058594, -0.32...]	[-0.5550722479820251, 0.2702311873435974, 0.04...]	[0.24783466756343842, -0.248960942029953, 0.02...]	[[-0.0019260947592556477, 0.009215019643306732...]	1
them	[-0.10130999982357025, 0.10941000282764435, 0...]	[0.5475010871887207, 0.8660883903503418, 2.817...]	[-0.7559828758239746, -0.4712887704372406, -1...]	[-0.2922026813030243, -0.1301671266555786, -0...]	[-0.24157099425792694, -0.8055092692375183, -0...]	[[-0.0019260947592556477, 0.009215019643306732...]	1
at	[0.17659999430179596, 0.0938510000705719, 0.24...]	[-0.5005946159362793, -0.4600788354873657, 0.5...]	[0.04092511534690857, -1.0951932668685913, -1...]	[-0.5613634586334229, -0.00903533399105072, -0...]	[-0.11999595910310745, 0.012994140386581421, -...]	[[-0.0019260947592556477, 0.009215019643306732...]	1
once	[-0.23837999999523163, 0.221670001745224, 0.35...]	[-0.39100387692451477, -0.8297092914581299, 2...]	[-0.46001458168029785, -1.2062749862670898, 0...]	[0.2988640069961548, 0.3360409140586853, -0.37...]	[0.6701997518539429, 1.1368376016616821, 0.244...]	[[-0.0019260947592556477, 0.009215019643306732...]	1
!	[0.38471999764442444, 0.49351000785827637, 0.4...]	[0.007945209741592407, -0.27733859419822693, 0...]	[-1.5816600322723389, -0.992130696773529, -0.1...]	[0.7550013065338135, -0.5257778167724609, -0.4...]	[-1.3351283073425293, 0.6296550035476685, -1.4...]	[[-0.0019260947592556477, 0.009215019643306732...]	1
Watch	[-0.38264000415802, -0.08968199789524078, 0.02...]	[-0.10218311846256256, -0.4334276020526886, 0...]	[-1.3921688795089722, 0.6997514963150024, -0.8...]	[-0.24852752685546875, 1.222611427307129, -0.1...]	[0.04002974182367325, -0.43536433577537537, -0...]	[[-0.0019260947592556477, 0.009215019643306732...]	1
your	[-0.5718399882316589, 0.046348001807928085, 0...]	[-0.4086211323738098, 1.0755341053009033, 1.78...]	[-0.8588163256645203, -2.3702170848846436, 0.0...]	[-0.035358428955078125, 0.7711482048034668, 0...]	[0.17885173857212067, 0.045830272138118744, -0...]	[[-0.0019260947592556477, 0.009215019643306732...]	1
RAM	[-1.875599980354309, -0.40814998745918274, -0...]	[-0.09772858023643494, 0.3632940351963043, -0...]	[1.1277621984481812, -1.689896583557129, -0.19...]	[0.4528151750564575, -0.36768051981925964, -0...]	[0.24783466756343842, -0.248960942029953, 0.02...]	[[-0.0019260947592556477, 0.009215019643306732...]	1
tough	[-0.5099300146102905, -0.1428000032901764, 0.5...]	[-0.22261293232440948, 0.21325691044330597, 0...]	[-1.3547197580337524, 0.43423181772232056, -1...]	[0.460737079381942 75, 0.05694812536239624, 0.5...]	[-0.24157099425792694, -0.8055092692375183, -0...]	[[-0.0019260947592556477, 0.009215019643306732...]	1
!	[0.38471999764442444, 0.49351000785827637, 0.4...]	[0.21658605337142944, -0.0493735149502754 2, 0...]	[-1.5816600322723389, -0.992130696773529, -0.1...]	[0.6830563545227051, -0.5751053094863892, -0.6...]	[-0.11999595910310745, 0.012994140386581421, -...]	[[-0.0019260947592556477, 0.009215019643306732...]	1

NLU : Apache License 2.0

```
# Multiple binary sentiment classifiers trained on various datasets
nlu.load('classify.sentiment').predict('I love NLU and Python WebDev Conf 2021!')
nlu.load('classify.sentiment.imdb').predict('The Matrix was a pretty good movie')
nlu.load('classify.sentiment.twitter').predict('@elonmusk Tesla stock price is too high imo')

# Translate between 200 languages
nlu.load('en.translate_to.zh').predict('NLU can translate between 200 languages!')

# Spellchecking
nlu.load('spell').predict('I liek to live dangertus!')

# Extract Named Entities
nlu.load('ner').predict('Donald Trump and John Biden dont share many oppinions')

# Unsupervised Keyword Extraction
nlu.load('yake').predict('Weights extract keywords withouth requiring weights!')

# Over 50+ classifiers on various problems
nlu.load('classify.emotion').predict('He was suprised by the diversity of NLU')
nlu.load('classify.spam').predict('Hello you are the heir to a 100 Million fortune!')
nlu.load('classify.fakenews').predict('Unicorns landed on mars!')
nlu.load('classify.sarcasm').predict('love the teachers who give exams the day after halloween')
nlu.load('en.classify.question').predict('How expensive is the Watch?')
nlu.load('en.classify.toxic').predict('You are to stupid')
nlu.load('classify.cyberbullying').predict('Women belong in the kitchen!') #sorry

# Get BERTology and Transformer Embeddings for Sentences and Words
nlu.load('bert').predict('BERTolgy Word embeddings!')
nlu.load('bert elmo albert glove').predict('Multiple BERTolgy Word embeddings!')
nlu.load('embed_sentence.bert ').predict('BERTolgy Sentence embeddings!')

# Text cleaning and Pre-Processing
nlu.load('lemmatize').predict('Get me the lemmatized version of a string')
nlu.load('normalize').predict('Get me the lemmatized version of a string')
nlu.load('clean').predict('Get me the lemmatized version of a string')

# Grammatical Parts of Speech
nlu.load('pos').predict('Extract Parts of Speech')
```

- 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

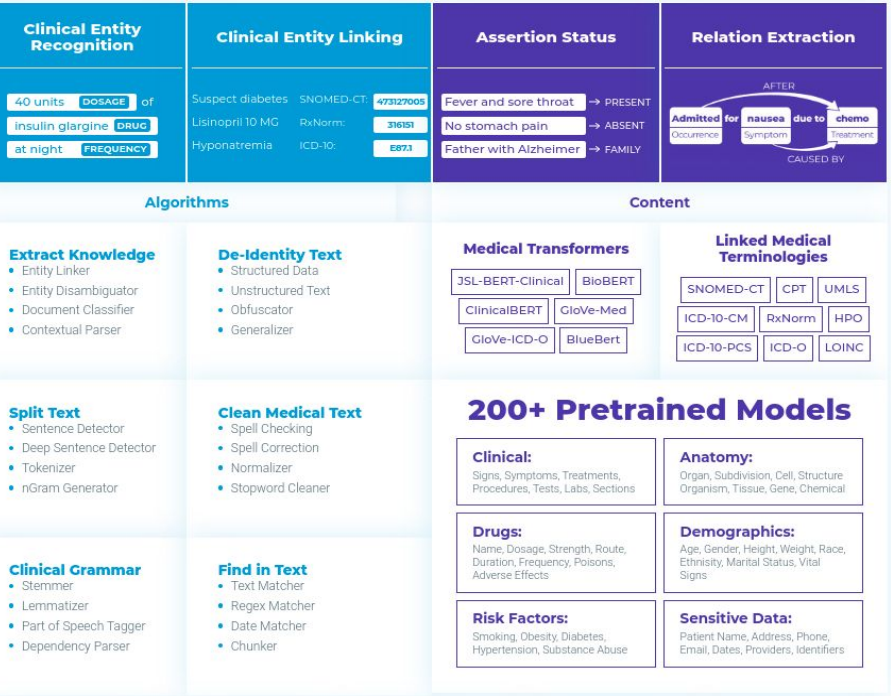
- BERT Embeddings
- ELMO Embeddings
- ALBERT Embeddings
- XLNet Embeddings
- Universal Sentence Encoder
- BERT Sentence Embeddings
- Sentence Embeddings
- Chunk Embeddings
- Unsupervised keywords extraction
- Language Detection & Identification
- Multi-class Text Classification
- Multi-label Text Classification
- 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!

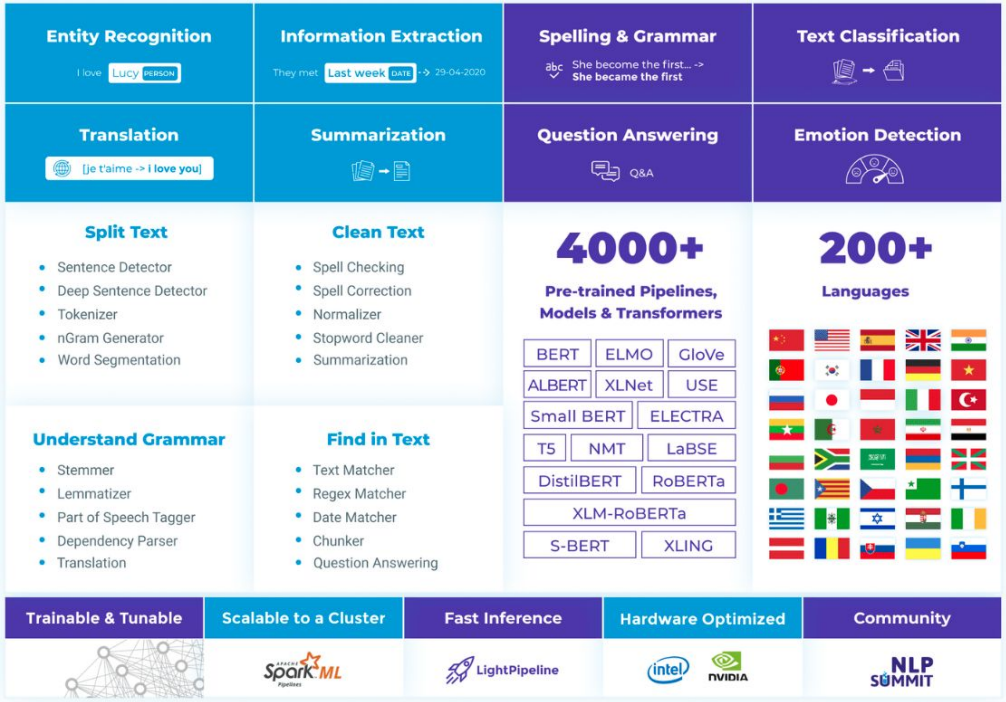
Coding Time

Tutorial Notebook

John Snow LABS

Spark NLP Modules



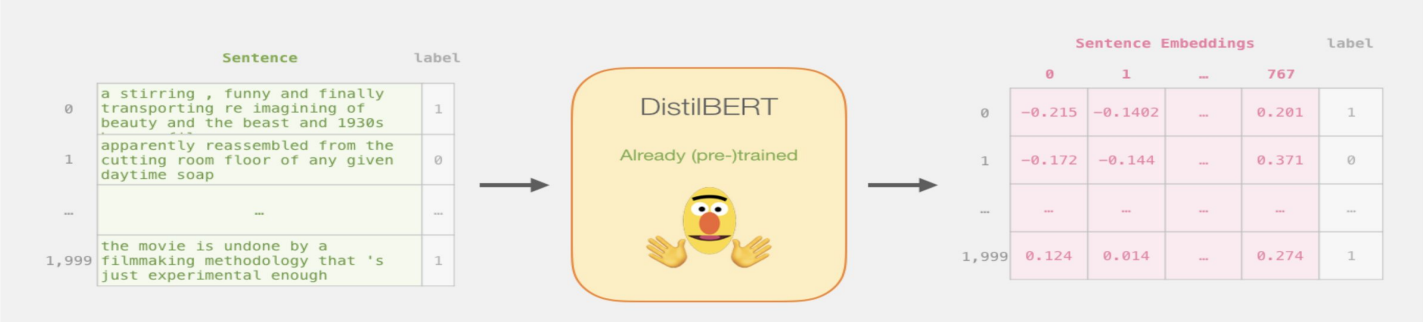


Spark NLP Modules (Enterprise and Public)

Text Classification with Word & Sentence Embeddings & Transformers

Glove
ELMO
BERT
Electra
Longformer
Universal Sentence Encoder
Albert
XLNet
RoBerta
DistilBert
XlmRoBerta
Bert Sentence Embeddings
Word2Vec
Bert Token Classifier
DistilBert Token Classifier
Albert Token Classifier
RoBerta Token Classifier
RoBerta Token Classifier
XlmRoBerta Token Classifier
XLNet Token Classifier
LongFormer Token Classifier
Bert Sequence Classifier
DistilBert Sequence Classifier
DistilBert Sequence Classifier
Albert Sequence Classifier
RoBerta Sequence Classifier
XlmRoBerta Sequence Classifier
XLNet Sequence Classifier
Longformer Sequence Classifier
RoBerta Sentence Embeddings
XlmRoBerta Sentence Embeddings

- Non Transformer Embedding
- Transformer Embedding
- Transformer Sequence Classifier
- Transformer Token Classifier



Spark NLP

- ClassifierDL
- SentimentDL
- MultiClassifierDL
- Sequence Classifier
- Token Classifier

Word & Sentence Embeddings

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 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 pre training method BERT. Researchers introduced two parameter-reduction techniques in ALBERT to lower memory consumption and increase training speed.

USE (Universal Sentence Encoder) is a Transformer-based model for encoding sentences into embedding vectors that specifically target transfer learning to other NLP tasks and outperforms previous word-embedding models on various NLP tasks

Word & Sentence Embeddings

DistilBert is a compressed version of BERT, which leverages knowledge distillation during the pre-training phase and shows that it is possible to reduce the size of a BERT model by 40%, while retaining 97% of its language understanding capabilities and being 60% faster. It introduces a triple loss combining language modeling, distillation and cosine-distance losses. The smaller, faster and lighter model is cheaper to pre-train and we demonstrate its capabilities for on-device computations in a proof-of-concept experiment and a comparative on-device

RoBerta is a optimized version of BERT, which has improved hyperparameters and training data size. The findings show that the original BERT was significantly undertrained and with (1) Longer training, bigger batches, (2) removing next sentence prediction objective, (3) training on longer sequences and (4) dynamically changing the masking pattern applied to the training data every previous BERT can be outperformed and new state of the art can be achieved

XlmRoBerta is a multilingual BERT model which significantly outperforms multilingual BERT on a variety of cross-lingual benchmark and large accuracy gains on various multi-lingual benchmarks for 88 languages that appear in the Wiki-100 corpus

Longformer is a Transformer-based model which improves on processing long sequences by introduces a windowed attention mechanism and a linearly scaling self-attention mechanism which scales linearly with sequence length and easily processes documents with thousands or more tokens.

Train Transformer Models via Huggingface or TfHub and scale with Spark NLP

TensorFlow Hub

TF Hub to Spark NLP

Spark NLP	TF Hub Notebooks	Colab
BertEmbeddings	TF Hub in Spark NLP - BERT	Open in Colab
BertSentenceEmbeddings	TF Hub in Spark NLP - BERT Sentence	Open in Colab
AlbertEmbeddings	TF Hub in Spark NLP - ALBERT	Open in Colab

HuggingFace to Spark NLP

Spark NLP	HuggingFace Notebooks	Colab
BertEmbeddings	HuggingFace in Spark NLP - BERT	Open in Colab
BertSentenceEmbeddings	HuggingFace in Spark NLP - BERT Sentence	Open in Colab
DistilBertEmbeddings	HuggingFace in Spark NLP - DistilBERT	Open in Colab
RoBertaEmbeddings	HuggingFace in Spark NLP - RoBERTa	Open in Colab
XlmRoBertaEmbeddings	HuggingFace in Spark NLP - XLM-RoBERTa	Open in Colab
AlbertEmbeddings	HuggingFace in Spark NLP - ALBERT	Open in Colab
XlnetEmbeddings	HuggingFace in Spark NLP - XLNet	Open in Colab
LongformerEmbeddings	HuggingFace in Spark NLP - Longformer	Open in Colab
BertForTokenClassification	HuggingFace in Spark NLP - BertForTokenClassification	Open in Colab
DistilBertForTokenClassification	HuggingFace in Spark NLP - DistilBertForTokenClassification	Open in Colab
AlbertForTokenClassification	HuggingFace in Spark NLP - AlbertForTokenClassification	Open in Colab
RoBertaForTokenClassification	HuggingFace in Spark NLP - RoBertaForTokenClassification	Open in Colab
XlmRoBertaForTokenClassification	HuggingFace in Spark NLP - XlmRoBertaForTokenClassification	Open in Colab
BertForSequenceClassification	HuggingFace in Spark NLP - BertForSequenceClassification	Open in Colab
DistilBertForSequenceClassification	HuggingFace in Spark NLP - DistilBertForSequenceClassification	Open in Colab
AlbertForSequenceClassification	HuggingFace in Spark NLP - AlbertForSequenceClassification	Open in Colab
RoBertaForSequenceClassification	HuggingFace in Spark NLP - RoBertaForSequenceClassification	Open in Colab
XlmRoBertaForSequenceClassification	HuggingFace in Spark NLP - XlmRoBertaForSequenceClassification	Open in Colab
XlnetForSequenceClassification	HuggingFace in Spark NLP - XlnetForSequenceClassification	Open in Colab

https://nlp.johnsnowlabs.com/docs/en/transformers#example-notebooks

100+ Languages supported by **Language-agnostic BERT Sentence Embedding (LABSE)** and **XLM-RoBERTa**

Train in **1 Language**, predict in **100+ different languages**

ISO	NAME	ISO	NAME	ISO	NAME
af	AFRIKAANS	ht	HAITIAN_CREOLE	pt	PORTUGUESE
am	AMHARIC	hu	HUNGARIAN	ro	ROMANIAN
ar	ARABIC	hy	ARMENIAN	ru	RUSSIAN
as	ASSAMESE	id	INDONESIAN	rw	KINYARWANDA
az	AZERBAIJANI	ig	IGBO	si	SINHALESE
be	BELARUSIAN	is	ICELANDIC	sk	SLOVAK
bg	BULGARIAN	it	ITALIAN	sl	SLOVENIAN
bn	BENGALI	ja	Japanese	sm	SAMOAN
bo	TIBETAN	jv	JAVANESE	sn	SHONA
bs	BOSNIAN	ka	GEORGIAN	so	SOMALI
ca	CATALAN	kk	KAZAKH	sq	ALBANIAN
ceb	CEBUANO	km	KHMER	sr	SERBIAN
co	CORSICAN	kn	KANNADA	st	SESOTHO
cs	CZECH	ko	KOREAN	su	SUNDANESE
cy	WELSH	ku	KURDISH	sv	SWEDISH
da	DANISH	ky	KYRGYZ	sw	SWAHILI
de	GERMAN	la	LATIN	ta	TAMIL
el	GREEK	lb	LUXEMBOURGISH	te	TELUGU
en	ENGLISH	lo	LAOTHIAN	tg	TAJIK
eo	ESPERANTO	lt	LITHUANIAN	th	THAI
es	SPANISH	lv	LATVIAN	tk	TURKMEN
et	ESTONIAN	mg	MALAGASY	tl	TAGALOG
eu	BASQUE	mi	MAORI	tr	TURKISH
fa	PERSIAN	mk	MACEDONIAN	tt	TATAR
fi	FINNISH	ml	MALAYALAM	ug	UIGHUR
fr	FRENCH	mn	MONGOLIAN	uk	UKRAINIAN
fy	FRISIAN	mr	MARATHI	ur	URDU
ga	IRISH	ms	MALAY	uz	UZBEK
gd	SCOTS_GAELIC	mt	MALTESE	vi	VIETNAMESE
gl	GALICIAN	my	BURMESE	wo	WOLOF
gu	GUJARATI	ne	NEPALI	xh	XHOSA
ha	HAUSA	nl	DUTCH	yi	YIDDISH
haw	HAWAIIAN	no	NORWEGIAN	yo	YORUBA
he	HEBREW	ny	NYANJA	zh	Chinese
hi	HINDI	or	ORIYA	zu	ZULU
hmn	HMONG	pa	PUNJABI		
hr	CROATIAN	pl	POLISH		

```
# Binary Class Classifier, 2 classes
nlu.load('xx.embed_sentence.labse train.sentiment').fit(train_df).predict(test_df)

# Multi Class Classifier, N classes
nlu.load('xx.embed_sentence.labse train.classifier').fit(train_df).predict(test_df)

# Multi Class Classifier with multiple labels example (i.e. Hashtags)
# N classes, where one row can be assigned up to N labels
nlu.load('xx.embed_sentence.labse train.multi_classifier').fit(train_df).predict(test_df)
```

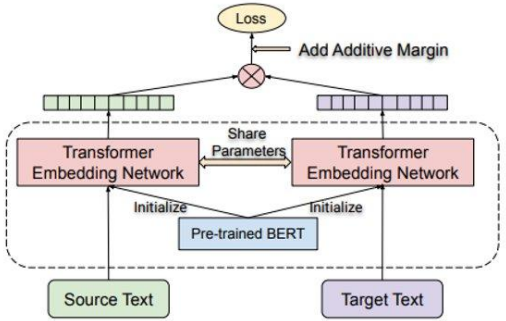


Figure 1: Dual encoder model with BERT based encoding modules.

Microsoft

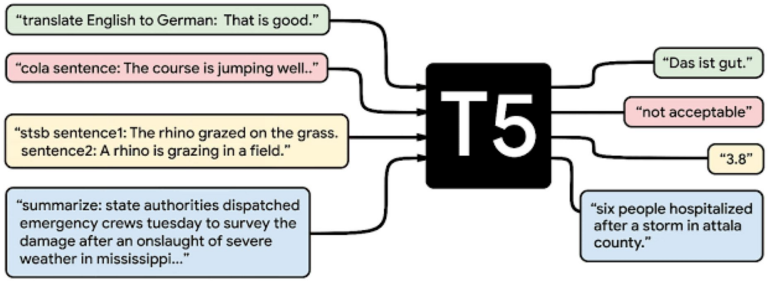
Translate between **200+ Languages**

With Marian: Fast Neural Machine Translation in C++

MARIANNMT

Fast Neural Machine Translation in C++


```
# Use ISO standards for the languages
nlu.load('<start_language>.translate_to.<target_language>')

#Translate Turkish to English:
nlu.load('tr.translate_to.en')

#Translate English to French:
nlu.load('en.translate_to.fr')

#Translate French to Hebrew
nlu.load('fr.translate_to.he')`

#Translate English to German
nlu.load('en.translate_to.de')`
```




```
# Closed book Question Answering
nlu.load('en.t5').predict('what is the capital of Germany?') # >>> Berlin
# Open Book Question answering
nlu.load('en.t5').predict('Who is president of Nigeria?') # >>> Muhammadu Buhari


# Open book Question Answering
context   = 'Peters last week was terrible! He had an accident and broke his leg while skiing!'
question1  = 'Why was peters week so bad?'
question2  = 'How did peter broke his leg?'
nlu.load('answer_question').predict(question1 + context) # >>> broke his leg
nlu.load('answer_question').predict(question2 + context) # >>> skiing



# Big T5 model for Summarization, Sentiment, Text Similarity and other SQUAD/GLUE tasks
pipe = nlu.load('t5')
pipe['t5'].settask('summarize')
pipe.predict(long_text)
```

[Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer]

1. Text summarization
2. Question answering
3. Translation
4. Sentiment analysis
5. Natural Language inference
6. Coreference resolution
7. Sentence Completion
8. Word sense disambiguation

Google

Every T5 Task with explanation:

Task Name	Explanation
1.CoLA	Classify if a sentence is grammatically correct
2.RTE	Classify whether a statement can be deducted from a sentence
3.MNLI	Classify for a hypothesis and premise whether they contradict or contradict each other or neither of both (3 class).
4.MRPC	Classify whether a pair of sentences is a re-phrasing of each other (semantically equivalent)
5.QNLI	Classify whether the answer to a question can be deducted from an answer candidate.
6.QQP	Classify whether a pair of questions is a re-phrasing of each other (semantically equivalent)
7.SST2	Classify the sentiment of a sentence as positive or negative
8.STSB	Classify the sentiment of a sentence on a scale from 1 to 5 (21 Sentiment classes)
9.CB	Classify for a premise and a hypothesis whether they contradict each other or not (binary).
10.COPA	Classify for a question, premise, and 2 choices which choice the correct choice is (binary).
11.MultiRc	Classify for a question, a paragraph of text, and an answer candidate, if the answer is correct (binary),
12.WiC	Classify for a pair of sentences and a disambigous word if the word has the same meaning in both sentences.
13.WSC/DPR	Predict for an ambiguous pronoun in a sentence what it is referring to.
14.Summarization	Summarize text into a shorter representation.
15.SQuAD	Answer a question for a given context.
16.WMT1.	Translate English to German
17.WMT2.	Translate English to French
18.WMT3.	Translate English to Romanian

GPT2 - Conditional Text Generation

- **GPT2** is very capable of generating text, but introduces new engineering challenges, so called **Prompt Engineering**
- The outputs of GPT2 depend on the text sequence we feed it in the beginning, the so called "**Prompt".** Choosing the right prompt for your problem is the biggest challenge
- **Tunable Parameters:**
  - **Min Output Length** : Minimum length of generated sequence
  - **Max Output Length** : Maximum length of generated sequence
  - **Repetition Penalty:** How much to penalize new tokens, based on their existing frequency in the text so far. Decreases the model's likelihood to repeat the same line verbatim
  - **Temperature** : Controls **Randomness**, lower values result in less random text generation. As the temperature approaches zero, the model will become deterministic and repetitive.
  - **Top P** : Controls diversity via nucleus sampling : 0.5 means half of all likelihood - weighted options are considered
  - Do Sampling
  - **Top K**: The number of highest probability vocabulary tokens to keep for top-k-filtering
  - **Number of Repeated N Grams :** How often a N Gram may be repeated. Setting it to N, all N Grams of size N can only occur once.

Open AI GPT2

GPT2 - Good vs Bad Prompts

```
37] # Bad prompting, the input text we condition GPT2 yields bad output, it does not understand the pattern we want from the original input
gpt2_pipe.predict("Suggest me a good Sci-Fi movie")
```

1 entry Filter

index	document	generated
0	Suggest me a good Sci-Fi movie	generate Suggest me a good Sci-Fi movie. I'm not sure if I'm going to be able to do this, but I'm sure I'll be able. (I'm sure you're going to want to do it.) So, I'm not going to do that. . , .

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

```
# Good prompting. help GPT2 out and by giving it a few samples in the prompt we condition it on
gpt2_pipe.predict("""Generate a top 10 movie list: \n
1.The Matrix \n
2.Terminator \n
3. """)
```

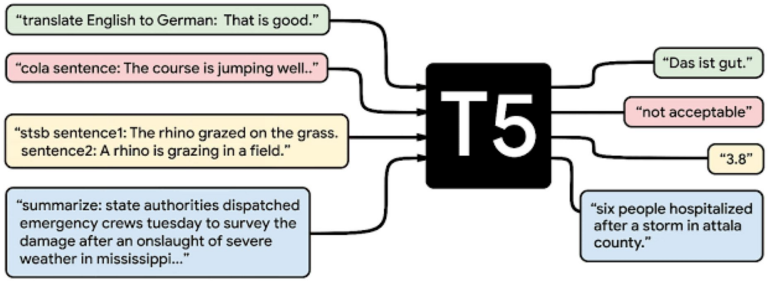
1 entry Filter

index	document	generated
0	Generate a top 10 movie list: 1.The Matrix 2.Terminator 3.	generate Generate a top 10 movie list: 1.The Matrix 2.Terminator 3.The Hunger Games 4.The Dark Knight Rises 5.The Lord of the Rings 6.The Hobbit 7.The Last Crusade 8.The Lion King 9.The Lego Movie 10.The LEGO Movie 11.The King of the Hill 12.The Jungle Book 13.The Legend of Zelda 14.The Little Mermaid 15.The Princess Bride 16.The Pirates of the Caribbean 17.The Twilight Saga 18

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

Open AI GPT2




```
# Closed book Question Answering
nlu.load('en.t5').predict('what is the capital of Germany?') # >>> Berlin
# Open Book Question answering
nlu.load('en.t5').predict('Who is president of Nigeria?') # >>> Muhammadu Buhari


# Open book Question Answering
context   = 'Peters last week was terrible! He had an accident and broke his leg while skiing!'
question1  = 'Why was peters week so bad?'
question2  = 'How did peter broke his leg?'
nlu.load('answer_question').predict(question1 + context) # >>> broke his leg
nlu.load('answer_question').predict(question2 + context) # >>> skiing



# Big T5 model for Summarization, Sentiment, Text Similarity and other SQUAD/GLUE tasks
pipe = nlu.load('t5')
pipe['t5'].settask('summarize')
pipe.predict(long_text)
```

[Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer]

1. Text summarization
2. Question answering
3. Translation
4. Generate SQL from natural language text
5. Text style transfer
6. Sentiment analysis
7. Natural Language inference
8. Coreference resolution
9. Sentence Completion
10. Word sense disambiguation

Google

Every T5 Task with explanation:

Task Name	Explanation
1.CoLA	Classify if a sentence is grammatically correct
2.RTE	Classify whether a statement can be deducted from a sentence
3.MNLI	Classify for a hypothesis and premise whether they contradict or contradict each other or neither of both (3 class).
4.MRPC	Classify whether a pair of sentences is a re-phrasing of each other (semantically equivalent)
5.QNLI	Classify whether the answer to a question can be deducted from an answer candidate.
6.QQP	Classify whether a pair of questions is a re-phrasing of each other (semantically equivalent)
7.SST2	Classify the sentiment of a sentence as positive or negative
8.STSB	Classify the sentiment of a sentence on a scale from 1 to 5 (21 Sentiment classes)
9.CB	Classify for a premise and a hypothesis whether they contradict each other or not (binary).
10.COPA	Classify for a question, premise, and 2 choices which choice the correct choice is (binary).
11.MultiRc	Classify for a question, a paragraph of text, and an answer candidate, if the answer is correct (binary),
12.WiC	Classify for a pair of sentences and a disambigous word if the word has the same meaning in both sentences.
13.WSC/DPR	Predict for an ambiguous pronoun in a sentence what it is referring to.
14.Summarization	Summarize text into a shorter representation.
15.SQuAD	Answer a question for a given context.
16.WMT1.	Translate English to German
17.WMT2.	Translate English to French
18.WMT3.	Translate English to Romanian

100+ Languages supported by **Language-agnostic BERT Sentence Embedding (LABSE)** and **XLM-RoBERTa**

Train in **1 Language**, predict in **100+ different languages**

ISO	NAME	ISO	NAME	ISO	NAME
af	AFRIKAANS	ht	HAITIAN_CREOLE	pt	PORTUGUESE
am	AMHARIC	hu	HUNGARIAN	ro	ROMANIAN
ar	ARABIC	hy	ARMENIAN	ru	RUSSIAN
as	ASSAMESE	id	INDONESIAN	rw	KINYARWANDA
az	AZERBAIJANI	ig	IGBO	si	SINHALESE
be	BELARUSIAN	is	ICELANDIC	sk	SLOVAK
bg	BULGARIAN	it	ITALIAN	sl	SLOVENIAN
bn	BENGALI	ja	Japanese	sm	SAMOAN
bo	TIBETAN	jv	JAVANESE	sn	SHONA
bs	BOSNIAN	ka	GEORGIAN	so	SOMALI
ca	CATALAN	kk	KAZAKH	sq	ALBANIAN
ceb	CEBUANO	km	KHMER	sr	SERBIAN
co	CORSICAN	kn	KANNADA	st	SESOTHO
cs	CZECH	ko	KOREAN	su	SUNDANESE
cy	WELSH	ku	KURDISH	sv	SWEDISH
da	DANISH	ky	KYRGYZ	sw	SWAHILI
de	GERMAN	la	LATIN	ta	TAMIL
el	GREEK	lb	LUXEMBOURGISH	te	TELUGU
en	ENGLISH	lo	LAOTHIAN	tg	TAJIK
eo	ESPERANTO	lt	LITHUANIAN	th	THAI
es	SPANISH	lv	LATVIAN	tk	TURKMEN
et	ESTONIAN	mg	MALAGASY	tl	TAGALOG
eu	BASQUE	mi	MAORI	tr	TURKISH
fa	PERSIAN	mk	MACEDONIAN	tt	TATAR
fi	FINNISH	ml	MALAYALAM	ug	UIGHUR
fr	FRENCH	mn	MONGOLIAN	uk	UKRAINIAN
fy	FRISIAN	mr	MARATHI	ur	URDU
ga	IRISH	ms	MALAY	uz	UZBEK
gd	SCOTS_GAELIC	mt	MALTESE	vi	VIETNAMESE
gl	GALICIAN	my	BURMESE	wo	WOLOF
gu	GUJARATI	ne	NEPALI	xh	XHOSA
ha	HAUSA	nl	DUTCH	yi	YIDDISH
haw	HAWAIIAN	no	NORWEGIAN	yo	YORUBA
he	HEBREW	ny	NYANJA	zh	Chinese
hi	HINDI	or	ORIYA	zu	ZULU
hmn	HMONG	pa	PUNJABI		
hr	CROATIAN	pl	POLISH		

```
# Binary Class Classifier, 2 classes
nlu.load('xx.embed_sentence.labse train.sentiment').fit(train_df).predict(test_df)

# Multi Class Classifier, N classes
nlu.load('xx.embed_sentence.labse train.classifier').fit(train_df).predict(test_df)

# Multi Class Classifier with multiple labels example (i.e. Hashtags)
# N classes, where one row can be assigned up to N labels
nlu.load('xx.embed_sentence.labse train.multi_classifier').fit(train_df).predict(test_df)
```

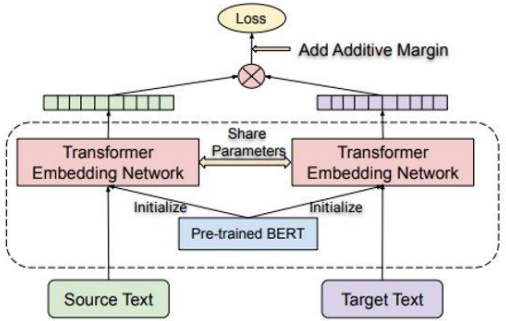
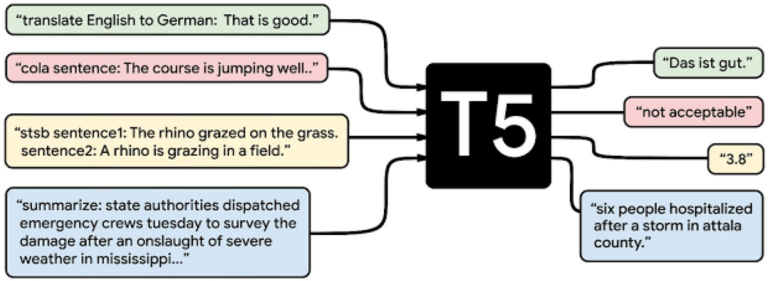


Figure 1: Dual encoder model with BERT based encoding modules.




```
# Closed book Question Answering
nlu.load('en.t5').predict('what is the capital of Germany?') # >>> Berlin
# Open Book Question answering
nlu.load('en.t5').predict('Who is president of Nigeria?') # >>> Muhammadu Buhari


# Open book Question Answering
context   = 'Peters last week was terrible! He had an accident and broke his leg while skiing!'
question1  = 'Why was peters week so bad?'
question2  = 'How did peter broke his leg?'
nlu.load('answer_question').predict(question1 + context) # >>> broke his leg
nlu.load('answer_question').predict(question2 + context) # >>> skiing



# Big T5 model for Summarization, Sentiment, Text Similarity and other SQUAD/GLUE tasks
pipe = nlu.load('t5')
pipe['t5'].settask('summarize')
pipe.predict(long_text)
```

[Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer]

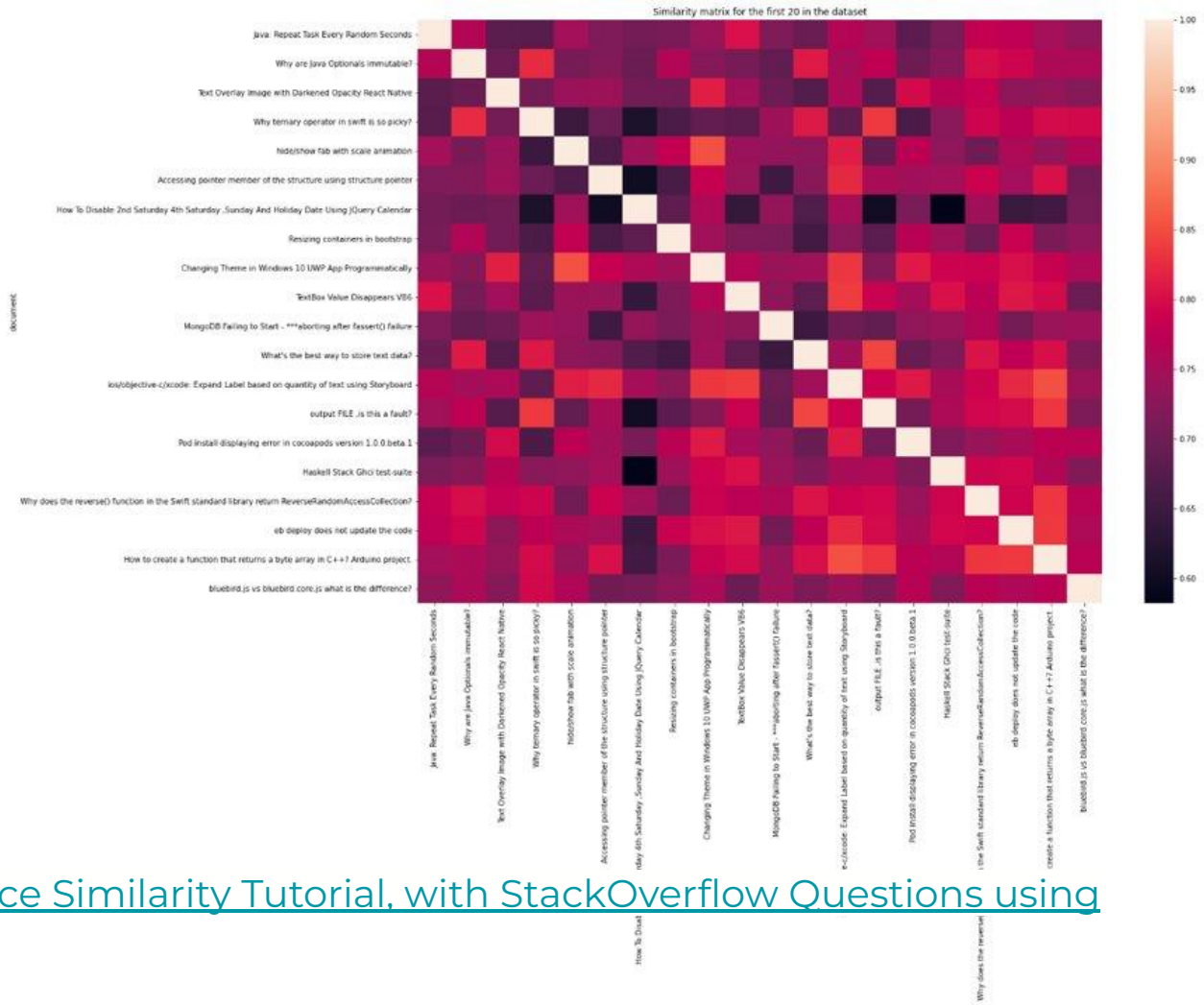
1. Text summarization
2. Question answering
3. Translation
4. Generate SQL from natural language text
5. Text style transfer
6. Sentiment analysis
7. Natural Language inference
8. Coreference resolution
9. Sentence Completion
10. Word sense disambiguation

Google

Every T5 Task with explanation:

Task Name	Explanation
1.CoLA	Classify if a sentence is grammatically correct
2.RTE	Classify whether a statement can be deducted from a sentence
3.MNLI	Classify for a hypothesis and premise whether they contradict or contradict each other or neither of both (3 class).
4.MRPC	Classify whether a pair of sentences is a re-phrasing of each other (semantically equivalent)
5.QNLI	Classify whether the answer to a question can be deducted from an answer candidate.
6.QQP	Classify whether a pair of questions is a re-phrasing of each other (semantically equivalent)
7.SST2	Classify the sentiment of a sentence as positive or negative
8.STSB	Classify the sentiment of a sentence on a scale from 1 to 5 (21 Sentiment classes)
9.CB	Classify for a premise and a hypothesis whether they contradict each other or not (binary).
10.COPA	Classify for a question, premise, and 2 choices which choice the correct choice is (binary).
11.MultiRc	Classify for a question, a paragraph of text, and an answer candidate, if the answer is correct (binary),
12.WiC	Classify for a pair of sentences and a disambigous word if the word has the same meaning in both sentences.
13.WSC/DPR	Predict for an ambiguous pronoun in a sentence what it is referring to.
14.Summarization	Summarize text into a shorter representation.
15.SQuAD	Answer a question for a given context.
16.WMT1.	Translate English to German
17.WMT2.	Translate English to French
18.WMT3.	Translate English to Romanian

Sentence Similarity With BERTology Embeds or T5



[In Depth and Easy Sentence Similarity Tutorial, with StackOverflow Questions using BERTology embeddings]

t-SNE Visualizations with NLU

[1 line of Python code for BERT, ALBERT, ELMO, ELECTRA, XLNET, GLOVE, Part of Speech with NLU and t-SNE]

Visualize NER results

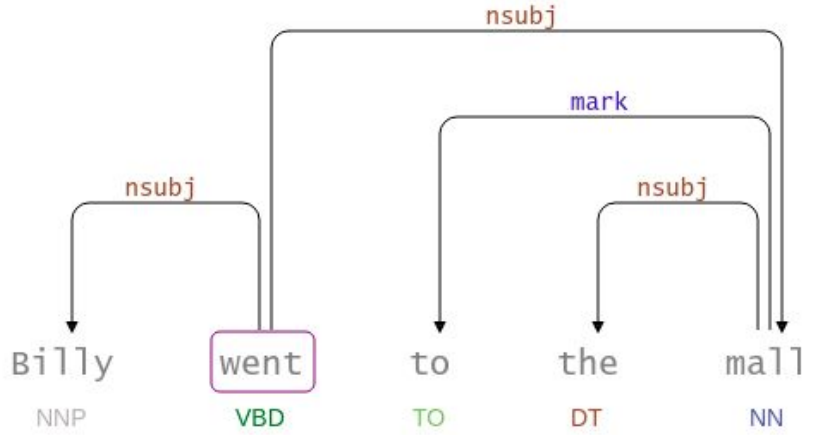
```
nlu.load('ner').viz("Donald Trump from America and Angela Merkel from Germany don't share many oppinions.")
```

```
onto_recognize_entities_sm download started this may take some time.
Approx size to download 160.1 MB
[OK!]
```

Visualize Dependency Trees

```
nlu.load('dep.typed').viz("Billy went to the mall")
```

```
dependency_typed_conllu download started this may take some time.
Approximate size to download 2.3 MB
[OK!]
dependency_conllu download started this may take some time.
Approximate size to download 16.7 MB
[OK!]
pos_anc download started this may take some time.
Approximate size to download 3.9 MB
[OK!]
sentence_detector_dl download started this may take some time.
Approximate size to download 354.6 KB
[OK!]
```



Visualize Assertion results

```
nlu.load('med_ner.clinical assert').viz("The MRI scan showed no signs of cancer in the left lung")
```

```
ner_clinical download started this may take some time.
Approximate size to download 13.9 MB
[OK!]
assertion_dl download started this may take some time.
Approximate size to download 1.3 MB
[OK!]
embeddings_clinical download started this may take some time.
Approximate size to download 1.6 GB
[OK!]
sentence_detector_dl download started this may take some time.
Approximate size to download 354.6 KB
[OK!]
```

Visualize Resolution

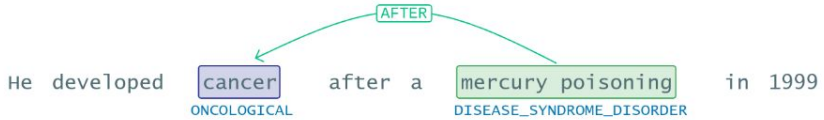
```
nlu.load('med_ner.jsl.wip.clinical resolve_chunk.rxnorm.in').viz("He took 2 pills of Aspirin daily")
```

```
jsl_ner_wip_clinical download started this may take some time.
Approximate size to download 14.5 MB
[OK!]
chunkresolve_rxnorm_in_clinical download started this may take some time.
Approximate size to download 26.9 MB
[OK!]
embeddings_clinical download started this may take some time.
Approximate size to download 1.6 GB
[OK!]
sentence_detector_dl download started this may take some time.
Approximate size to download 354.6 KB
[OK!]
```

Visualize Entity Relationships

```
nlu.load('med_ner.jsl.wip.clinical relation.temporal_events').viz('He developed cancer after a mercury poisoning in 1999 ')
```

```
jsl_ner_wip_clinical download started this may take some time.
Approximate size to download 14.5 MB
[OK!]
redl_temporal_events_biobert download started this may take some time.
Approximate size to download 383.3 MB
[OK!]
embeddings_clinical download started this may take some time.
Approximate size to download 1.6 GB
[OK!]
sentence_detector_dl download started this may take some time.
Approximate size to download 354.6 KB
[OK!]
```

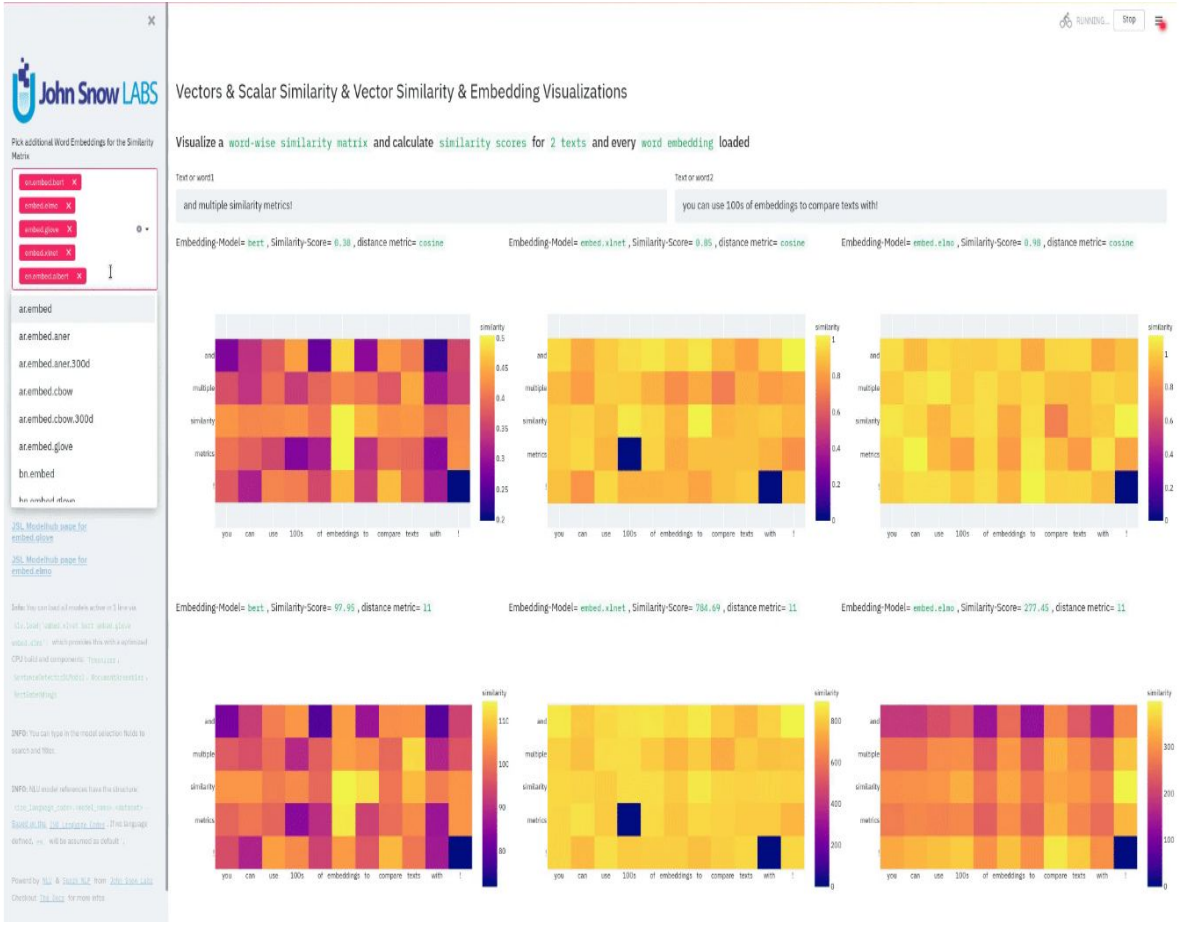


nlu_viz_cheatsheet.py

```python
data = 'I want some pretty visualizations please!'

# Viz detected entities
nlu.load('ner').viz(data)

# Viz labeled dependency tree and POS tags
nlu.load('dep.typed').viz(data)

# Viz asserted statuses
nlu.load('med_ner.clinical assert').viz(data)

# Viz resolved sentences
nlu.load('med_ner.jsl.wip.clinical resolve.icd10cm').viz(data)

# Viz resolved entities
nlu.load('med_ner.jsl.wip.clinical resolve_chunk.rxnorm.in').viz(data)

# Viz extracted relationships between entities
nlu.load('med_ner.jsl.wip.clinical relation.temporal_events').viz(data)
```

Explore 100+ Embeddings via 6 Similarity Metrics with 0 lines of code with NLU & Streamlit

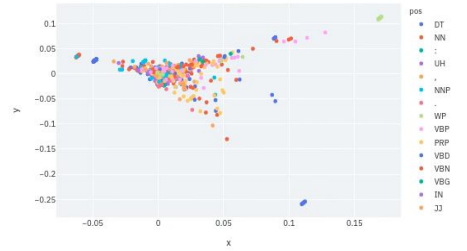
- Compare Multiple similarity Metrics And Embeddings at the same time
- Supported similarities :
  - Cosine
  - Cityblock
  - Euclidean
  - L2
  - L1
  - Manhattan



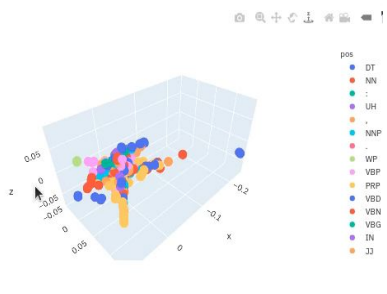
Explore 100+ Embeddings via 10+ Manifold and Matrix Decomposition with 0 lines of code with NLU & Streamlit

Compare multiple dimension reduction
Techniques and
Embeddings at the same time

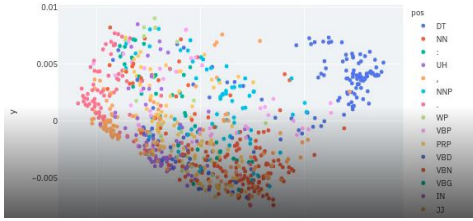
Word-Embeddings = `small_bert_L2_128` , Manifold-Algo = `LLE` for `D=2`



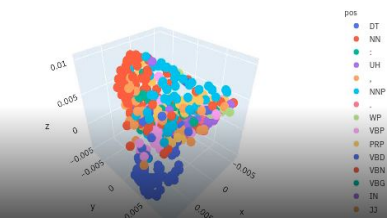
Word-Embeddings = `small_bert_L2_128` , Manifold-Algo = `LLE` for `D=3`

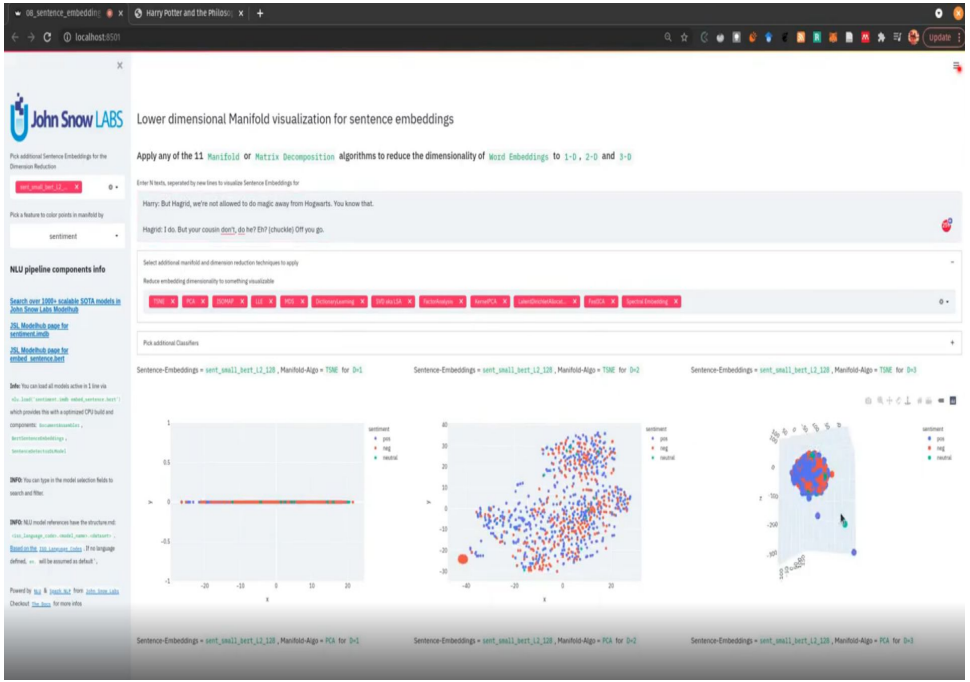


Word-Embeddings = `small_bert_L2_128` , Manifold-Algo = `Spectral Embedding` for `D=2`



Word-Embeddings = `small_bert_L2_128` , Manifold-Algo = `Spectral Embedding` for `D=3`





nlu_streamlit_cheatsheet.py

```python
# Input text data for visualizations
data = 'Billy from Berlin wants some pretty visualizations please!'

# Full UI with self generating Python code snippets for every feature
nlu.load('ner').viz_streamlit(data)

# Classification
nlu.load('sentiment').viz_streamlit_classes(data)

# Named Entity Recognition (NER)
nlu.load('ner').viz_streamlit_ner(data)

# Dependency Tree and Part of Speech (POS) Tags
nlu.load('dep.typed').viz_streamlit_dep_tree(data)

# Token features
nlu.load('stemm pos spell').viz_streamlit_token(data)

# Calculate similarity between two texts based on Word Embeddings
nlu.load('bert').viz_streamlit_word_similarity(['I love  NLU! <3','I also Streamlit! <3'])

# Raw visualizations in Streamlit
nlu.load('<Model>').viz(data, write_to_streamlit=True)

# Predict on any datatype and returns Pandas df
nlu.load('<Model>').predict(data)

# Enable caching
nlu.enable_streamlit_caching()
```

Streamlit Time

https://nlu.johnsnowlabs.com/docs/en/streamlit_viz_examples

pip install nlu==3.4.1rc5

NLU OCR

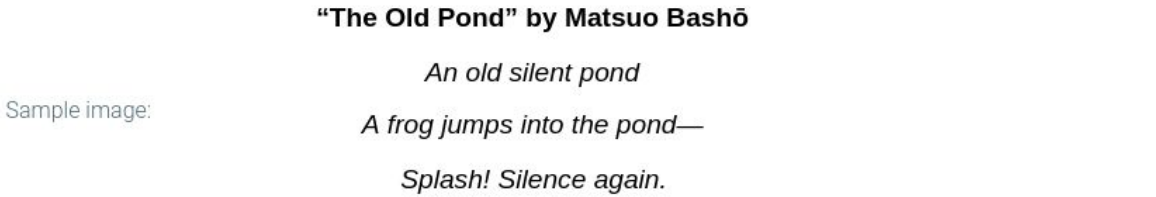
- Extract text from
  - **Images**
  - **PDFs**
  - **DOCX**

NLU Spell	Transformer Class
nlu.load(`img2text`)	**ImageToText**
nlu.load(`pdf2text`)	**PdfToText**
nlu.load(`doc2text`)	**DocToText**

https://nlu.johnsnowlabs.com/docs/en/nlu_for_ocr

Image to Text

Sample image:




```
nlu.load('img2text').predict('path/to/haiku.png')
```

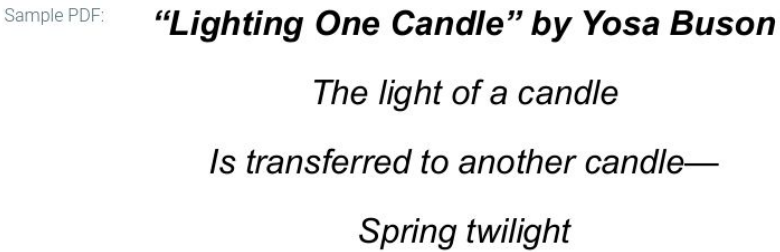
Output of IMG OCR:

text
“The Old Pond” by Matsuo Basho
An old silent pond
A frog jumps into the pond—
Splash! Silence again.

PDF to Text

haiku.pdf

Sample PDF:




```
nlu.load('pdf2text').predict('path/to/haiku.pdf')
```

Output of PDF OCR:

text
“Lighting One Candle” by Yosa Buson
The light of a candle
Is transferred to another candle—
Spring twilight

DOCX to text

Sample DOCX:

“In a Station of the Metro” by Ezra Pound

The apparition of these faces in the crowd;

Petals on a wet, black bough.

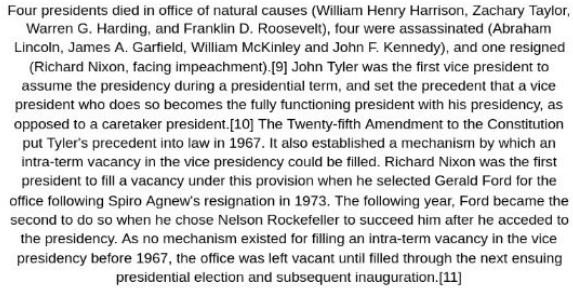
```
nlu.load('doc2text').predict('path/to/haiku.docx')
```

Output of DOCX OCR:

text
“In a Station of the Metro” by Ezra Pound
The apparition of these faces in the crowd;
Petals on a wet, black bough.

Combine OCR and NLP models

Sample image containing named entities **from U.S. Presidents Wikipedia**:




```
nlu.load('img2text ner').predict('path/to/presidents.png')
```

Output of image OCR and NER NLP :

entities_ner	entities_ner_class	entities_ner_confidence
Four	CARDINAL	0.9986
Abraham Lincoln	PERSON	0.705514
John F. Kennedy),	PERSON	0.966533
one	CARDINAL	0.9457
Richard Nixon,	PERSON	0.71895
John Tyler	PERSON	0.9929
first	ORDINAL	0.9811
The Twenty-fifth Amendment	LAW	0.548033
Constitution	LAW	0.9762
Tyler's	CARDINAL	0.5329
1967	DATE	0.8926
Richard Nixon	PERSON	0.99515
first	ORDINAL	0.9588
Gerald Ford	PERSON	0.996

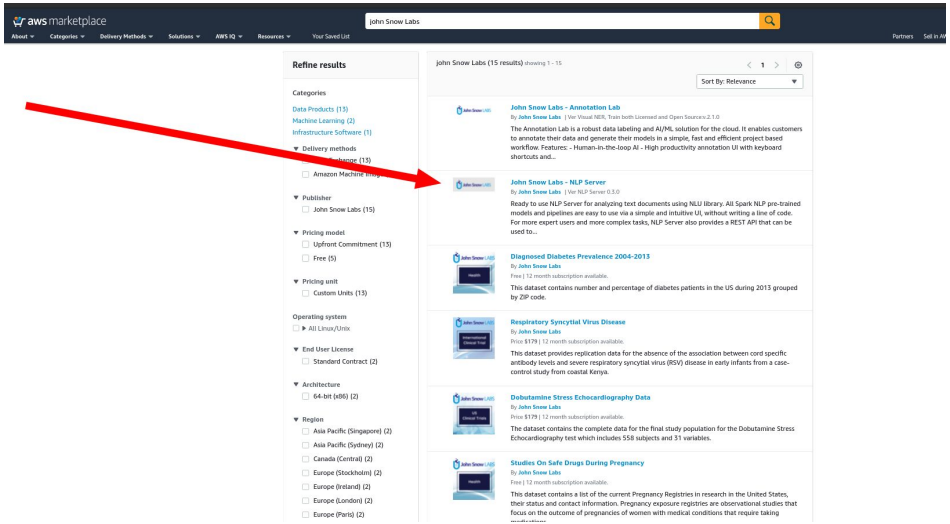
More NLU Ressources

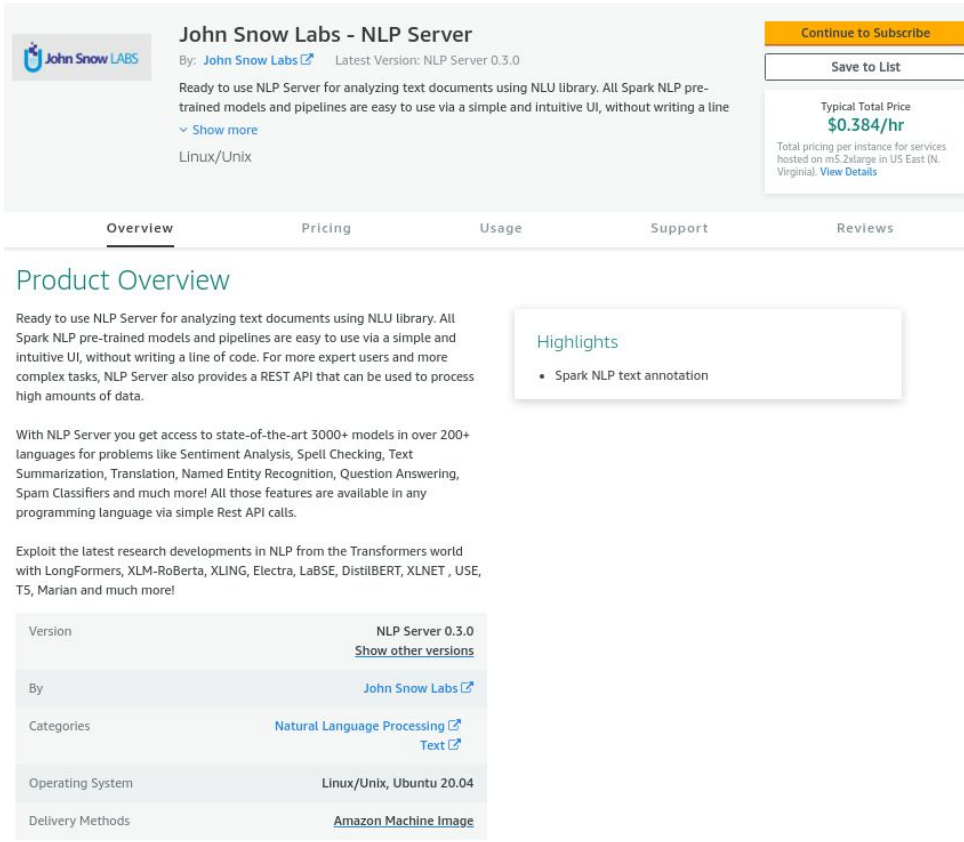
- Join our Slack
- NLU Website
- NLU Githuab
- Many more NLU example tutorials
- Overview of every powerful nlu 1-liner
- Checkout the Modelshub for an overview of all models
- Checkout the NLU Namespace where you can find every model as a tabel
- Intro to NLU article
- Indepth and Easy Sentence Similarity Tutorial, with StackOverflow Questions using BERTology embeddings
- 1 line of Python code for BERT, ALBERT, ELMO, ELECTRA, XLNET, GLOVE, Part of Speech with NLU and t-SNE

Webinars and Videos

- NLU & Streamlit Tutorial
- Crash course of the 50 + Medical Domains and the 200+ Healtchare models in NLU
- Multi Lingual NLU Webinar - Tutorial on Chinese News dataset
- John Snow Labs NLU: Become a Data Science Superhero with One Line of Python code
- Python Web Def Conf - Python's NLU library: 4,000+ Models, 200+ Languages, State of the Art Accuracy, 1 Line of Code
- NYC/DC NLP Meetup with NLU

NLP Server - Powerful private deployable NLP Rest API

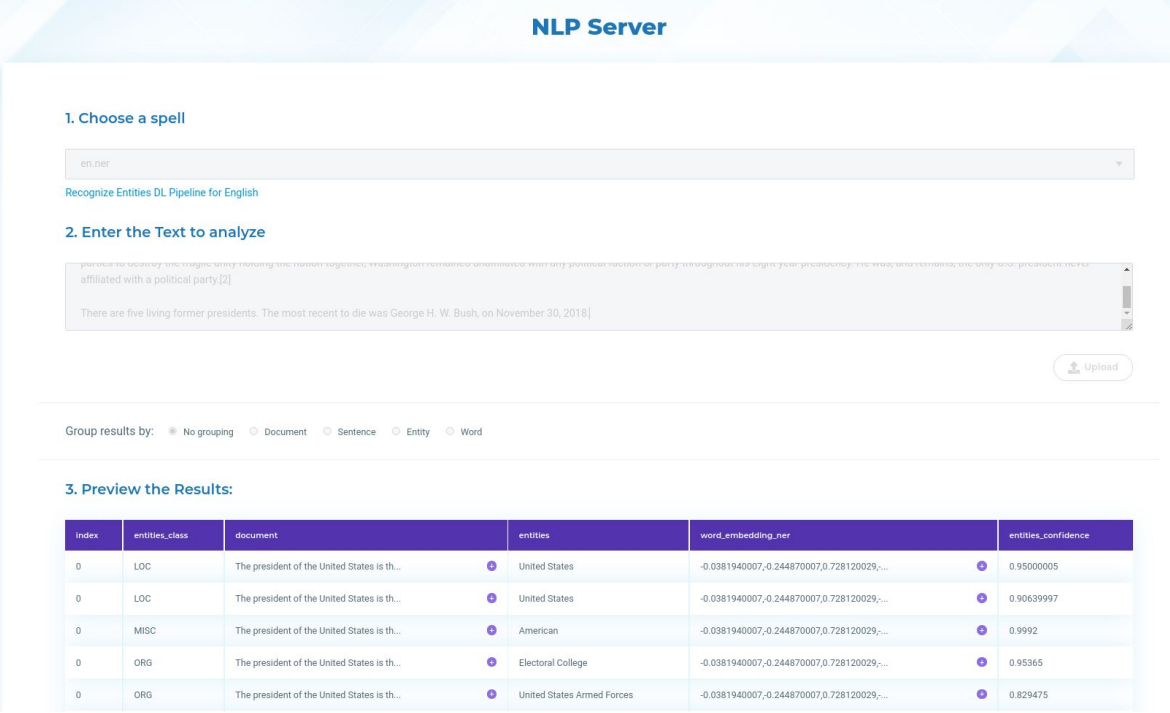




https://nlp.johnsnowlabs.com/docs/en/nlp_server/nlp_server

4000+ Models in 200 languages available as API and GUI with a few clicks in the AWS marketplace.

This enables you to use all of these Models with any programming Language by simple using REST API calls



API Usage in 3 Steps :

- Query - Send Data request
- Poll - Is server done processing?
- Get result - Get result when server done

Step 1 : Post a request

- Post a data processing request
- Returns UUID status

POST /api/results/

Request samples

Payload

Content type
application/json

Copy Expand all Collapse all

```
{
    "spell": "tokenize",
    "data": "I love NLU! <3"
}
```

Response samples

200 404 422

Content type
application/json

Copy Expand all Collapse all

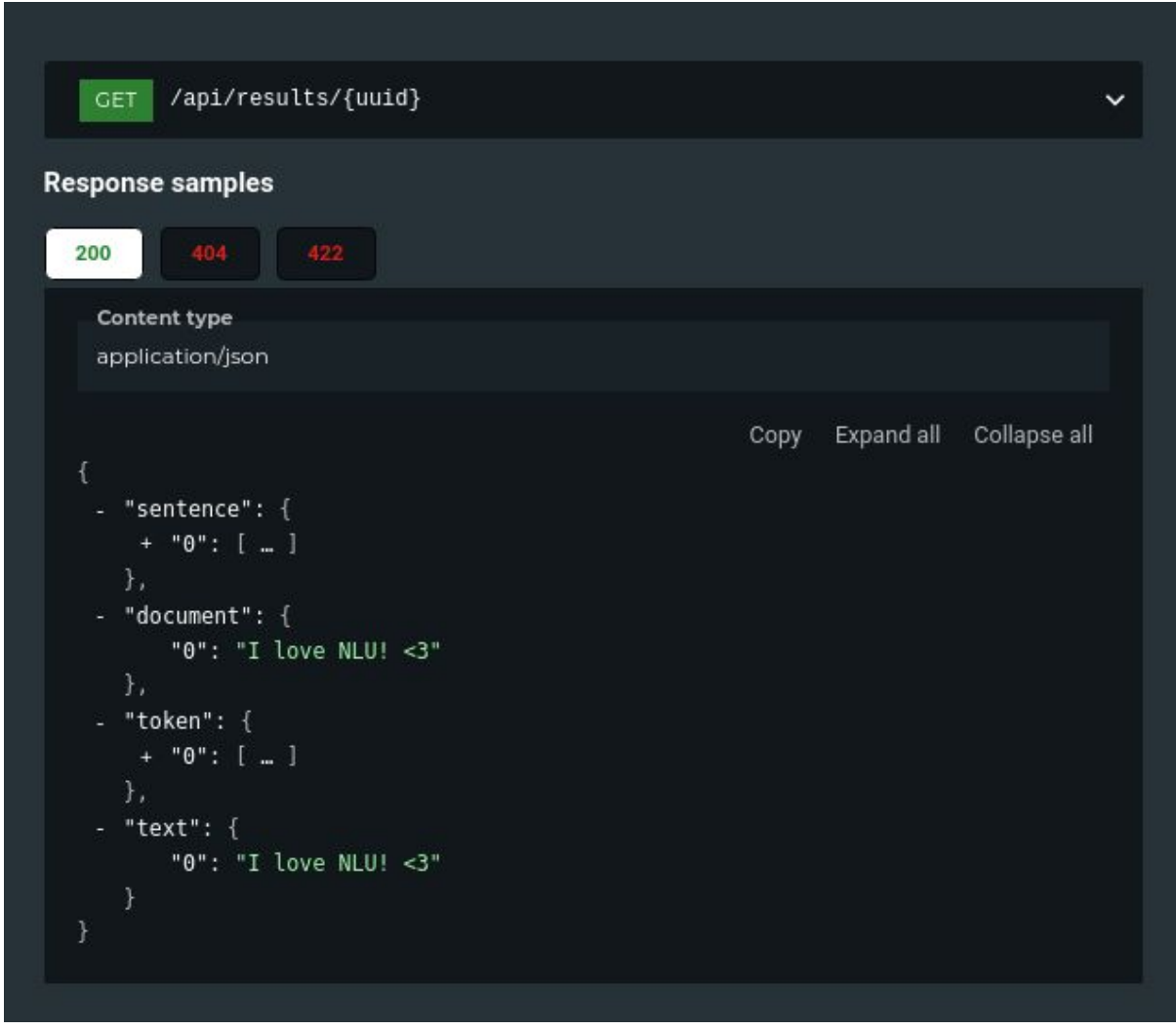
```
{
    "uuid": "njGeaDKl1VoRR7kjf002h"
}
```

Step 2 : Poll for result

- Poll Server for processing status

Step 3 : Get result as JSON

- Get result for UUID
- Contains NLP predictions



All of Spark NLP features behind a simple to use API. deployable in a few clicks on AWS

NLP_server_cheatsheet.py

```
import requests
# Invoke Processing with tokenization spell
r = requests.post(f'http://localhost:5000/api/results',json={"spell": "tokenize","data": "I love NLU!
<3"})
# Use the uuid to get your processed data
uuid = r.json()['uuid']

# Get status of processing
r = requests.get(f'http://localhost:5000/api/results/{uuid}/status').json
>>> {'status': {'code': 'success', 'message': None}}

# Get results
r = requests.get(f'http://localhost:5000/api/results/{uuid}').json()
>>> {'sentence': {'0': ['I love NLU! <3']}, 'document': {'0': 'I love NLU! <3'}, 'token': {'0': ['I',
'love', 'NLU', '!', '<3']}}
```

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

Youtube Tutorial for NLP Server Deployment on AWS

Thank you.


christian@JohnSnowLabs.com


@ckl_it


In/Christian-Kasim-Loan

Medium.com/@Christian.Kasim.Loan