John Snow LABS

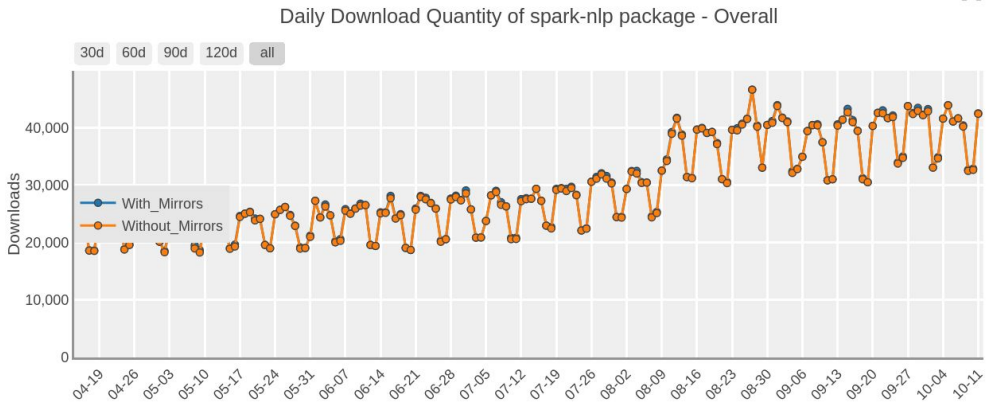
Automated Text Generation & Data-Augmentation for Medicine, Finance, Law, and E-Commerce

September 7. 2022

Christian Kasim Loan
Lead Data Scientist, ML and Big Data Engineer
christian@johnsnowlabs.com

Introducing Spark NLP

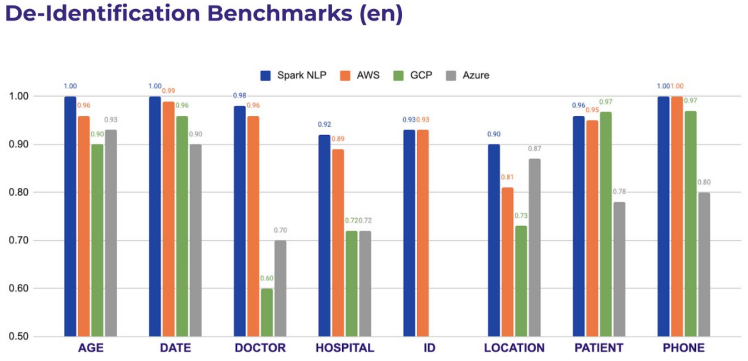
Total downloads	14,558,265
Total downloads - 30 days	1,371,781
Total downloads - 7 days	318,804

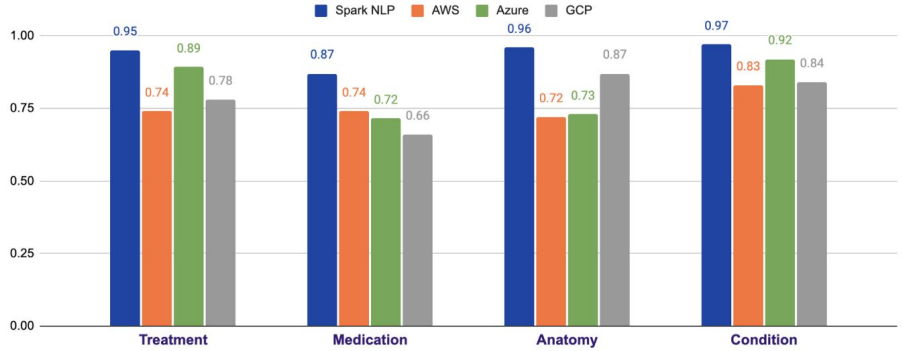


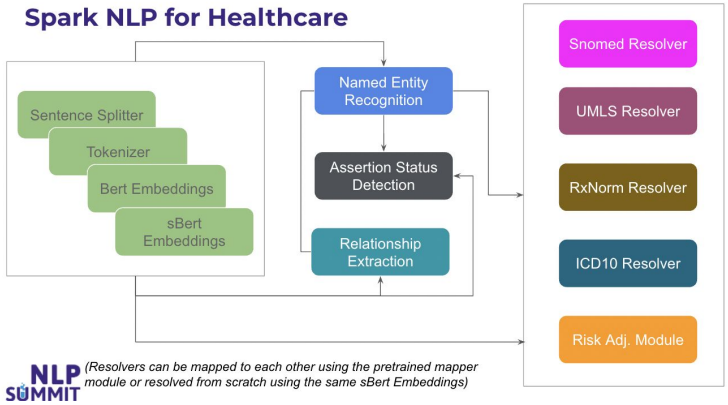
- Spark NLP is an open-source natural language processing library, built on top of Apache Spark and Spark ML. (initial release: Oct 2017)
  - A single unified solution for all your NLP needs
  - Take advantage of transfer learning and implementing the latest and greatest **SOTA** algorithms and models 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)
  - The most scalable, accurate and fastest library in NLP history

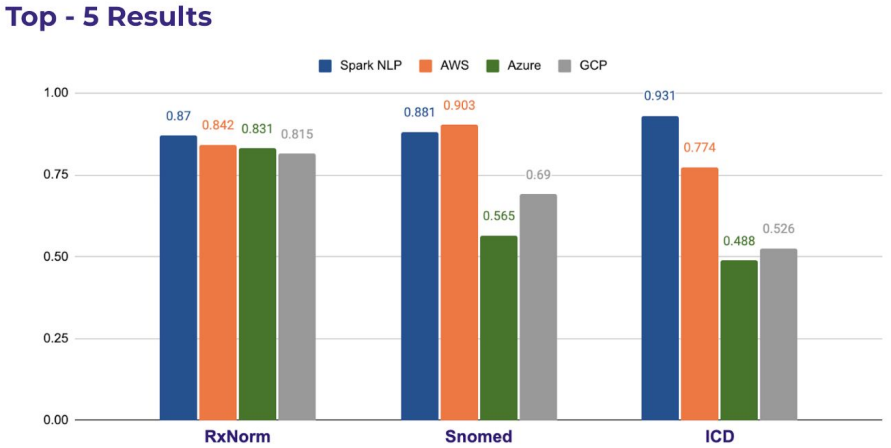
https://medium.com/spark-nlp/introduction-to-spark-nlp-foundations-and-basic-components-part-i-c83b7629ed59

Most Accurate in the Industry





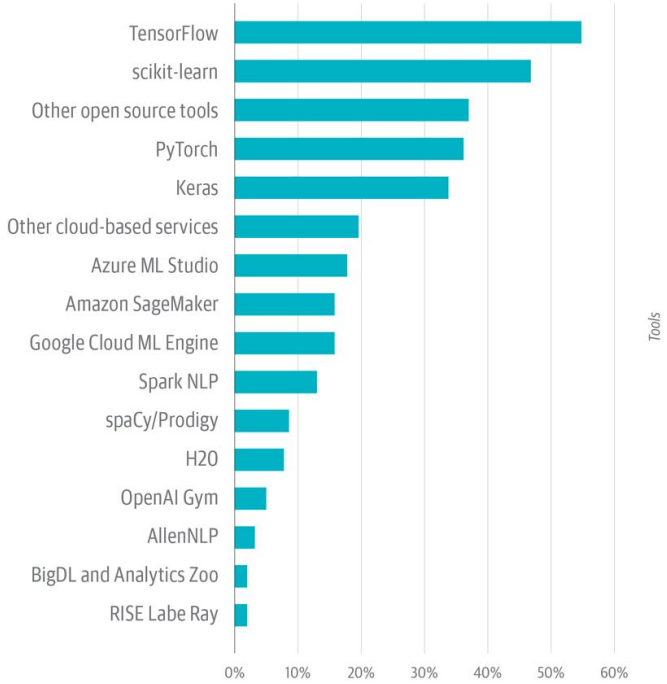




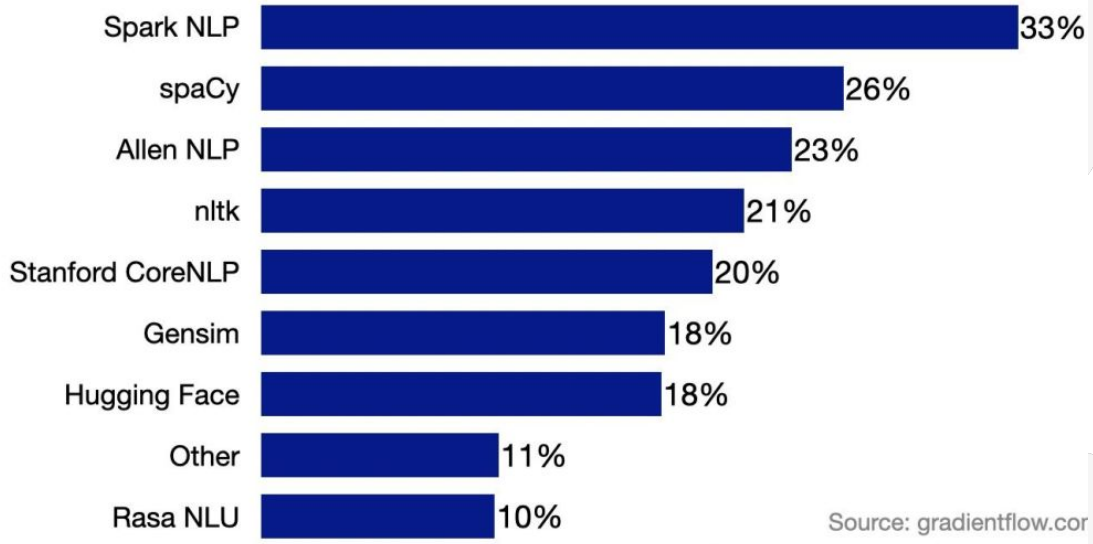
[Spark NLP vs Commercial healthcare NLP Solutions Benchmark]

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 2021

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

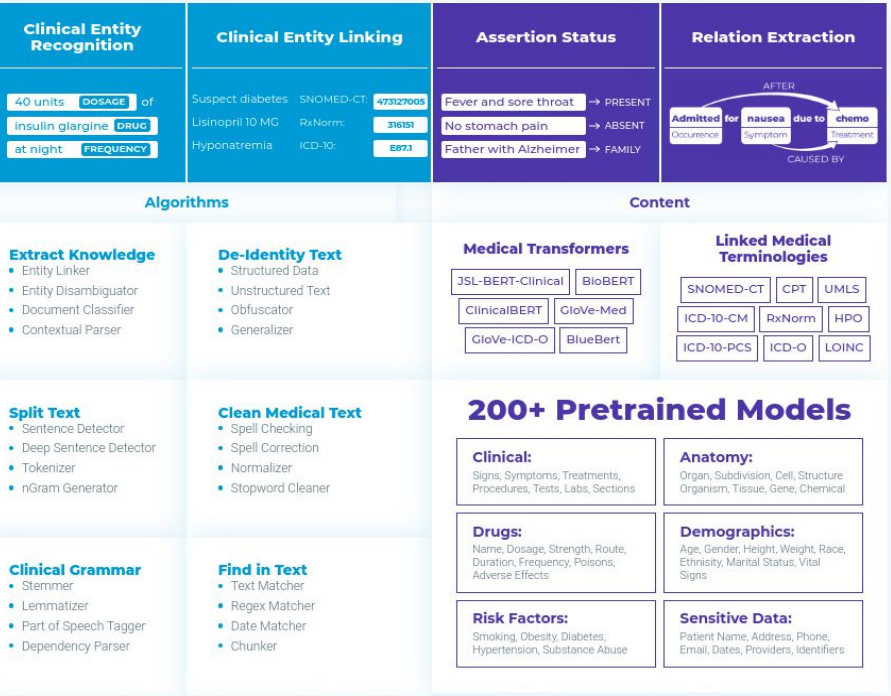

Georgia
Tech

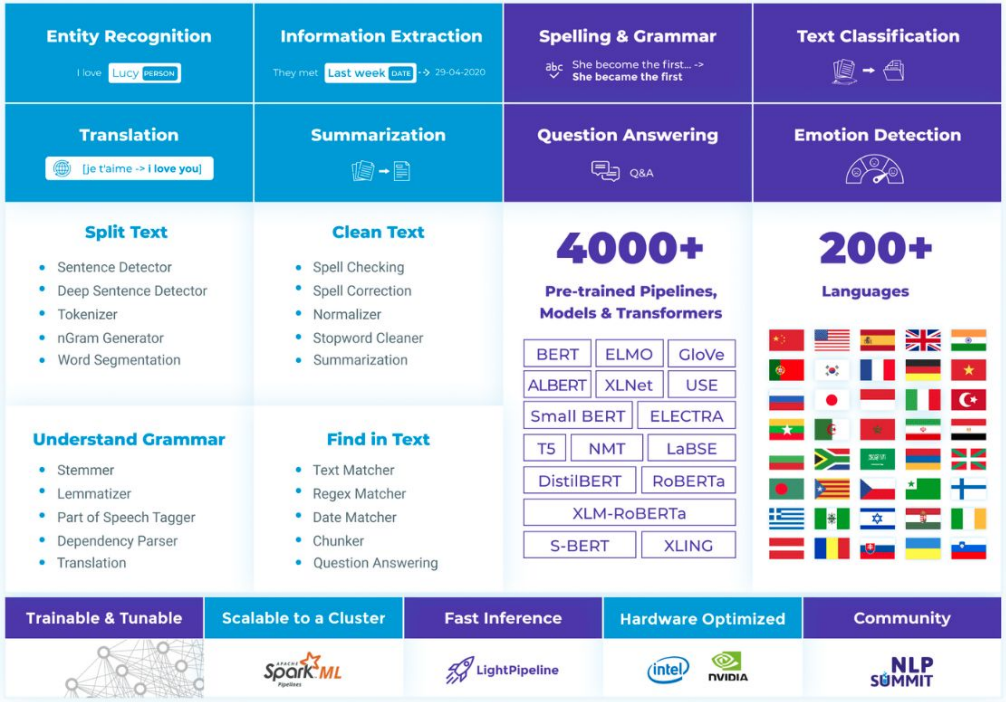

STANFORD
UNIVERSITY

Recognized by the Technology Experts

John Snow LABS

Spark NLP Modules





Spark NLP Modules (Enterprise and Public)

The most scalable, accurate and fastest arsenal of NLP transformers in history

John Snow LABS

Apache Spark

Core NLP Transformer Models

CamemBertEmbeddings	GloveEmbeddings	Word2VecEmbeddings	Doc2VecEmbeddings
DeBertaEmbeddings	LongformerEmbeddings	BertEmbeddings	XlmRoBertaEmbeddings
DeBertaForQuestionAnswering	LongformerForTokenClassification	BertForTokenClassification	XlmRoBertaForTokenClassification
GPT2Transformer	LongformerForSequenceClassification	BertForSequenceClassification	XlmRoBertaForSequenceClassification
MarianTransformer	LongformerForQuestionAnswering	BertForQuestionAnswering	XlmRoBertaForQuestionAnswering
T5Transformer	UniversalSentenceEncoder	BertSentenceEmbeddings	XlmRoBertaSentenceEmbeddings
RoBertaEmbeddings	XlnetEmbeddings	DistilBertEmbeddings	AlbertEmbeddings
RoBertaForTokenClassification	XlnetForTokenClassification	DistilBertForTokenClassification	AlbertForTokenClassification
RoBertaForSequenceClassification	XlnetForSequenceClassification	DistilBertForSequenceClassification	AlbertForSequenceClassification
RoBertaForQuestionAnswering	ElmoEmbeddings	DistilBertForQuestionAnswering	AlbertForQuestionAnswering

- Transformer for Sequence to Sequence
- Transformer For Token Classification
- Transformer For Sequence Classification
- Transformer For Question Answering
- Transformer Sentence Embeddings
- Transformer Word Embeddings
- Non Transformer Embeddings

Subflavored NLP Transformer Extensions

SciBertEmbeddings	ElectraForQuestionAnswering	MurilBertForQuestionAnswering	MurilBertEmbeddings	LabseSentenceEmbeddings
IndoBertForQuestionAnswering	ElectraSentenceEmbeddings	MurilBertSentenceEmbeddings	SciBertForQuestionAnswering	SapBertForQuestionAnswering
ElectraEmbeddings	BioBertForQuestionAnswering	CovidBertForQuestionAnswering	MiniLmForQuestionAnswering	MacBertForQuestionAnswering
BioFormerForQuestionAnswering	BioBertSentenceEmbeddings	CovidDistilBertForQuestionAnswering	LinkBertForQuestionAnswering	BioBertEmbeddings

Search them all in the Modelshub

https://nlp.johnsnowlabs.com/models

Christian Kasim Loan
Lead Data Scientist
christian@johnsnowlabs.com

NLP Transformers & Embeddings

- BERT : https://arxiv.org/abs/1810.04805
- XLNet : https://arxiv.org/abs/1906.08237
- Albert : https://arxiv.org/abs/1909.11942
- Elmo : https://arxiv.org/abs/1802.05365
- USE : https://arxiv.org/abs/1803.11175
- DistilBert : https://arxiv.org/abs/1910.01108
- RoBerta : https://arxiv.org/abs/1907.11692
- DeBerta : https://arxiv.org/abs/2006.03654
- XlmRoberta : https://arxiv.org/abs/1911.02116
- Longformer : https://arxiv.org/abs/2004.05150
- Electra : https://arxiv.org/abs/2003.10555
- T5 : https://arxiv.org/abs/1910.10683
- Marian: https://arxiv.org/abs/1804.00344
- GPT2: https://openai.com/blog/better-language-models/
- SpanBERT https://arxiv.org/abs/1907.10529
- CamemBERT https://camembert-model.fr/
- SciBert https://www.aclweb.org/anthology/D19-1371/
- MiniLm https://arxiv.org/abs/2002.10957
- CovidBERT https://arxiv.org/abs/2005.07503
- BioBERT https://arxiv.org/abs/1901.08746
- indoBERT https://arxiv.org/abs/2011.00677
- MuRIL https://arxiv.org/abs/2103.10730
- sapBERT https://github.com/cambridgeltl/sapbert
- BioFormer https://github.com/WGLab/Bioformer
- LinkBERT https://arxiv.org/abs/2203.15827
- MacBERT https://aclanthology.org/2020.findings-emnlp.58
- DOC2Vec : https://arxiv.org/abs/1301.3781
- Word2Vec: https://arxiv.org/abs/1301.3781

Biomedical Named Entity Recognition at Scale

Veysel Kocaman
John Snow Labs Inc.
16192 Coastal Highway
Lewes, DE , USA 19958
veysel@johnsnowlabs.com

***Abstract*—Named entity recognition (NER) is a widely applicable natural language processing task and building block of question answering, topic modeling, information retrieval, etc. In the medical domain, NER plays a crucial role by extracting meaningful chunks from clinical notes and reports, which are then fed to downstream tasks like assertion status detection, entity resolution, relation extraction, and de-identification. Reimplementing a Bi-LSTM-CNN-Char deep learning architecture on top of Apache Spark, we present a single trainable NER model that obtains new state-of-the-art results on seven public biomedical benchmarks without using heavy contextual embeddings like BERT. This includes improving BC4CHEMD to 93.72% (4.1% gain), Species800 to 80.91% (4.6% gain), and JNLPBA to 81.29% (5.2% gain). In addition, this model is freely available within a production-grade code base as part of the open-source Spark NLP library; can scale up for training and inference in any Spark cluster; has GPU support and libraries for popular programming languages such as Python, R, Scala and Java; and can be extended to support other human languages with no code changes.**

I. Introduction

Electronic health records (EHRs) are the primary source of information for clinicians tracking the care of their patients. Information fed into these systems may be found in structured fields for which values are inputted electronically (e.g. laboratory test orders or results) [1] but most of the time information in these records is unstructured making it largely inaccessible

Spark NLP: Natural Language Understanding at Scale

Veysel Kocaman, David Talby

John Snow Labs Inc.
16192 Coastal Highway
Lewes, DE , USA 19958
eysel, david}@johnsnowlabs.com

Accurate Clinical and Biomedical Named Entity Recognition at Scale

Anonymous NAACL-HLT 2021 submission

Abstract

Named entity recognition (NER) is one of the most important building blocks of NLP tasks in the medical domain by extracting meaningful chunks from clinical notes and reports, which are then fed to downstream tasks like assertion status detection, entity resolution, relation extraction, and de-identification. Due to the growing volume of healthcare data in unstructured format, an increasingly important challenge is providing high accuracy implementations of state-of-the-art deep learning (DL) algorithms at scale. In this study, we introduce a production-grade clinical and biomedical NER algorithm based on a modified BiLSTM-CNN-Char DL architecture built on top of Apache Spark. This algorithm establishes new state-of-the-art accuracy on 7 of 8 well-known biomedical NER benchmarks and 3 clinical concept extraction challenges: 2010 i2b2/VA clinical concept extraction, 2014 n2c2 de-identification, and 2018 n2c2 medication extraction. Moreover, clinical NER models trained using this implemen-

Improving Clinical Document Understanding on COVID-19 Research with Spark NLP

Veysel Kocaman, David Talby
John Snow Labs Inc.
16192 Coastal Highway
Lewes, DE , USA 19958
{veysel, david}@johnsnowlabs.com

Abstract

Following the global COVID-19 pandemic, the number of scientific papers studying the virus has grown massively, leading to increased interest in automated literate review. We present a clinical text mining system that improves on previous efforts in three ways. First, it can recognize over 100 different entity types including social determinants of health, anatomy, risk factors, and adverse events in addition to other commonly used clinical and biomedical entities. Second, the text processing pipeline includes assertion status detection, to distinguish between clinical facts that are present, absent, conditional, or about someone other than the patient. Third, the deep learning models used are more accurate than previously available, leveraging an integrated pipeline of state-of-the-art pre-trained named entity recognition models, and improving on the previous best performing benchmarks for assertion status detection. We illustrate extracting trends and insights - e.g. most frequent disorders and symptoms, and most common vital signs and EKG findings – from the COVID-19 Open Research Dataset (CORD-19). The system is built using the Spark NLP library which natively supports scaling to use distributed clusters, leveraging GPU's, configurable and reusable NLP pipelines, healthcare-specific embeddings, and the ability to train models to support new entity types or human languages with no code changes.

be found in structured fields for which values are inputted electronically (e.g. laboratory test orders or results) (Liede et al. 2015) but most of the time information in these records is unstructured making it largely inaccessible for statistical analysis (Murdoch and Detsky 2013). These records include information such as the reason for administering drugs, previous disorders of the patient or the outcome of past treatments, and they are the largest source of empirical data in biomedical research, allowing for major scientific findings in highly relevant disorders such as cancer and Alzheimer's disease (Perera et al. 2014).

A primary building block in such text mining systems is named entity recognition (NER) - which is regarded as a critical precursor for question answering, topic modelling, information retrieval, etc (Yadav and Bethard 2019). In the medical domain, NER recognizes the first meaningful chunks out of a clinical note, which are then fed down the processing pipeline as an input to subsequent downstream tasks such as clinical assertion status detection (Uzuner et al. 2011), clinical entity resolution (Tzitzivacos 2007) and de-identification of sensitive data (Uzuner, Luo, and Szolovits 2007) (see Figure 1). However, segmentation of clinical and drug entities is considered to be a difficult task in biomedical NER systems because of complex orthographic structures of named entities

Peer-reviewed conference papers on Spark NLP NER

Optimized, Tested, Supported Integrations

What is NLU?

- All of the **9000+** John Snow Labs models for over **200 Languages** 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

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

TOKENIZATION & SPELL CHECKING

```
nlu.load('spell').predict('I liek pentut butr and jelli')
```

token	checked	id
I	I	1
liek	like	1
peantut	peanut	1
buttr	butter	1
and	and	1
jelli	jelly	1

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 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

NLU : Apache License 2.0

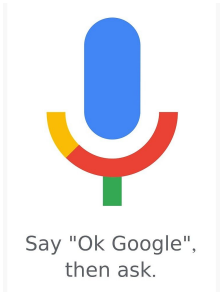
```
# Multiple binary sentiment classifiers trained on various datasets
nlu.load('classify.sentiment').predict('I love NLU and Python WebDev Conf 2021!')
nlu.load('classify.sentiment.imdb').predict('The Matrix was a pretty good movie')
nlu.load('classify.sentiment.twitter').predict('@elonmusk Tesla stock 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
- +9000 pre-trained models in 200 languages!

Where is Text Generation used in the Industry?

- Natural Language Generation (NLG) is a sub-field of natural language processing (NLP)
  - Automatically generate **coherent** and **useful** text for human consumption
- NLG Systems in use
  - Google Translate
  - Dialogue Systems (Siri, Alexa, Cortana, Google Home, Bigsby)
  - Summarizers (Google Search Results)
    - Emails, News, Papers, Reports, Conversations
  - Creative writing
    - Stories, Poetry, Narratives
    - Fake News, Botnets



Text Generation in Action : Code Time

Dipping our toes in NLG with NLU using GPT2

GPT2 NLU Notebook

Why are generation repeating and what did we just do?

A quick theory crash course follows

```
[ ] gpt2_model['gpt2'].setDoSample(False)
    generation_df = gpt2_model.predict("My Favorite Food is!")
    print(generation_df.generated.values[0])
    generation_df = gpt2_model.predict("My Favorite Food is!")
    print(generation_df.generated.values[0])
```

```
 My Favorite Food is!

I love the way the food is cooked. I love the texture. I like the way it
 My Favorite Food is!

I love the way the food is cooked. I love the texture. I like the way it
```

Why are generation repeating?

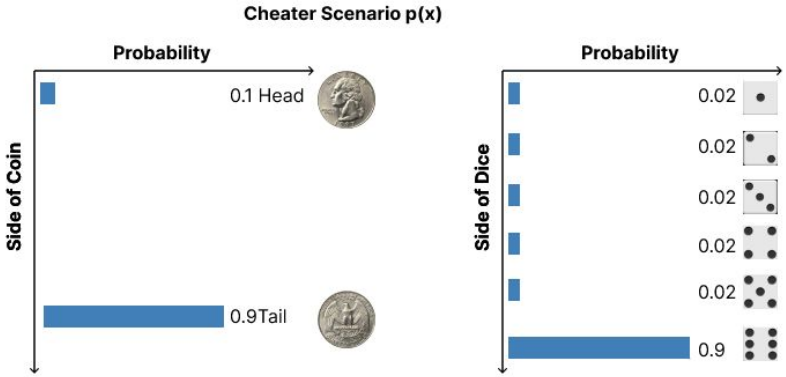
```
[ ] gpt2_model['gpt2'].setDoSample(True)
    gpt2_model['gpt2'].setMaxOutputLength(25)
    generation_df = gpt2_model.predict("My Favorite Food is!")
    print(generation_df.generated.values[0])
    generation_df = gpt2_model.predict("My Favorite Food is!")
    print(generation_df.generated.values[0])
```

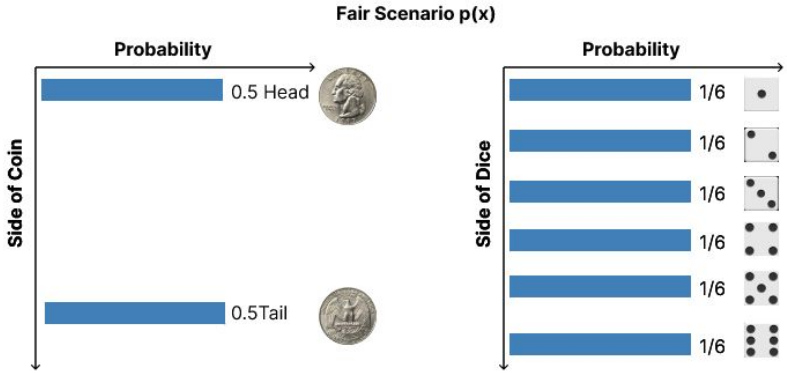
```
 My Favorite Food is!

We have a great family who loves having family gatherings with our kids. From our family gatherings to
 My Favorite Food is!

A recent "food for thought" article, entitled "How to Make a Food that is Easy
```

Why are generations **not** repeating?

What is a **Probability Distribution**?

- A probability distribution is simply a function $p : \Omega \rightarrow [0, 1]$ which returns a probability for events x form a sample space Ω
- In a coin flip scenario, sample space is $\Omega = \{Heads, Tails\}$ and for every event $p(x) = 0.5$ if the coin is fair.
- In a 6-sided-Dice throw scenario, sample space is $\Omega = \{1, 2, 3, 4, 5, 6\}$ and for every $p(x) = 1/6$ if the dice is fair.

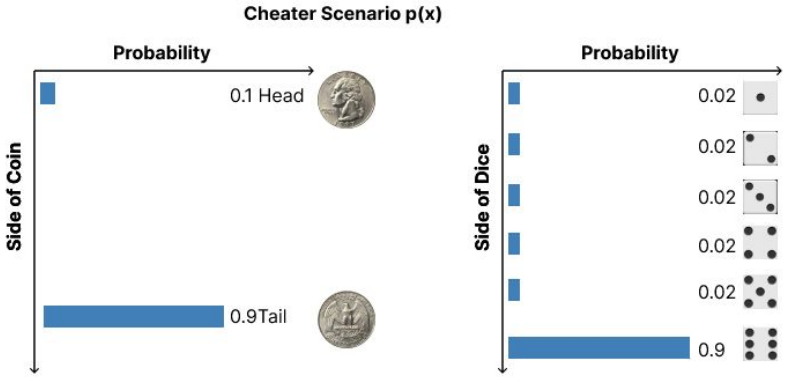


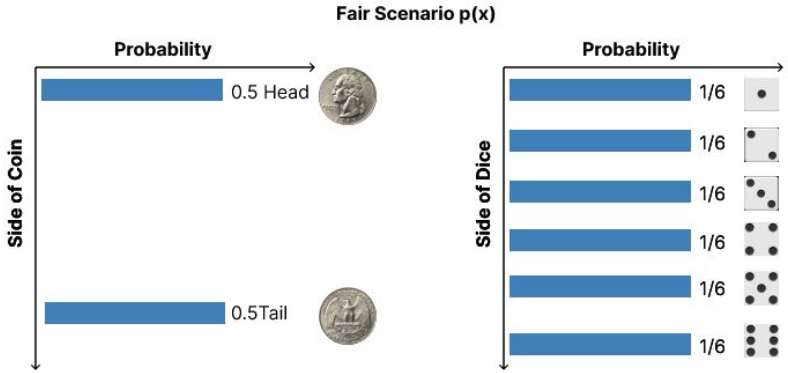


What is **drawing** from a probability distribution?

Also known as **sampling** from a distribution

- If we know the probability distribution of a sample space, we can **simulate** it
- We simulate by **drawing** elements in Ω using some **sampling algorithm**

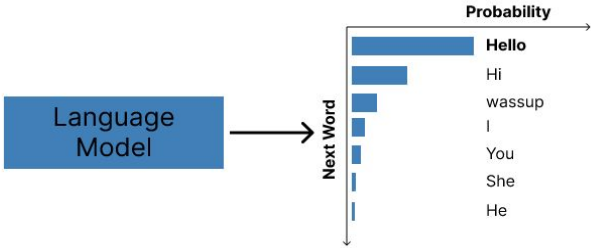




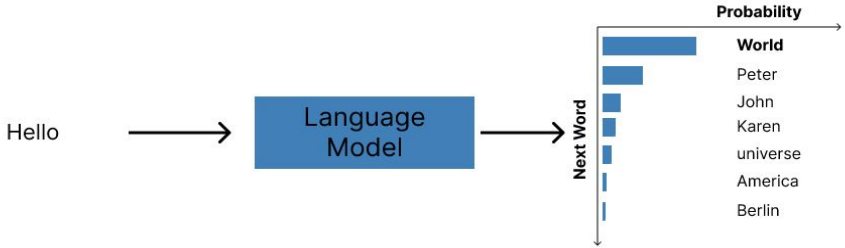
What is a **Language Model**?

- A Language Model is simply a probability distribution $p(x)$ where the sample space Ω is the **set of words in the human language**
- $p(hello)$ should tell us the probability of the word **hello** occuring as **beginning of human written text**
- $p(world \mid hello)$ should tell us the probability of the word **world** given that humman written text started with **hello**

p(x) = Softmax(Logits)



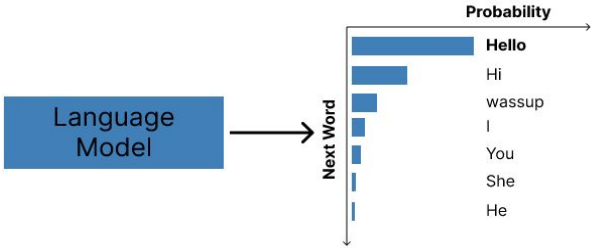
p(Hello | World) = Softmax(Logits)



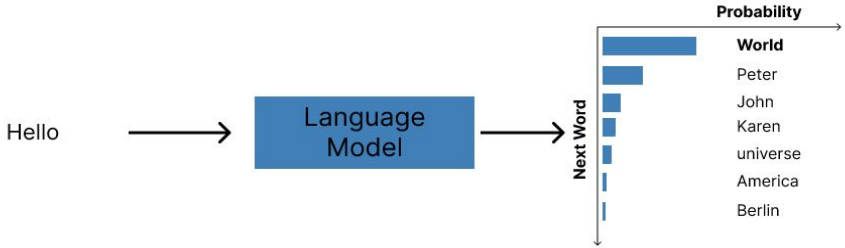
How does a Language Model **generate Text**?

- The function $p(x)$ is used to draw a word from the human language
- The function $p(x|y)$ is used to draw a word from the human language, given a start word
    - We can feed the outputs of $p(x|y)$ back to itself to generate one word at a time, conditioned on preceeding words
    - **y** is reffered to as **the prompt** we are giving to the model
- Finding an decent estimation of p has been solved by Transformer based deep learning architectures like Bert,GPT,T5, etc..
- When we use these Pretrained Language Models to generate text our main concern **shaping** and **sampling** from these distrubtions
- Final layer of generative NLP Transformer exposes one logit for every word in the training vocabulary.
    - Applying Softmax function to outputs of logits, yields probability distrubtion $p(x \mid y)$ which we sample from

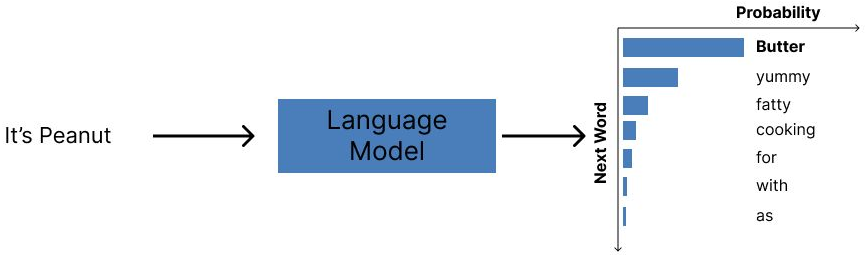
p(x) = Softmax(Logits)



p(Hello | World) = Softmax(Logits)

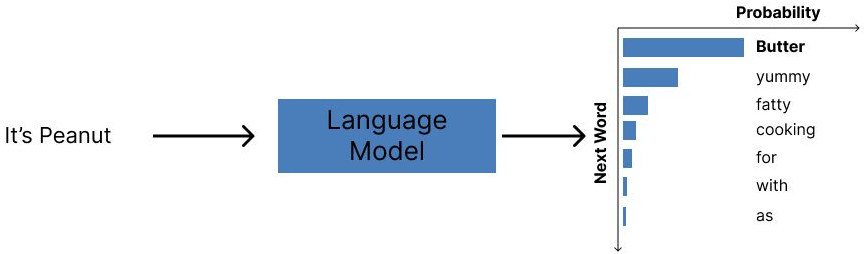


P(Next-Word | It's, Peanut) = Softmax(Logits)

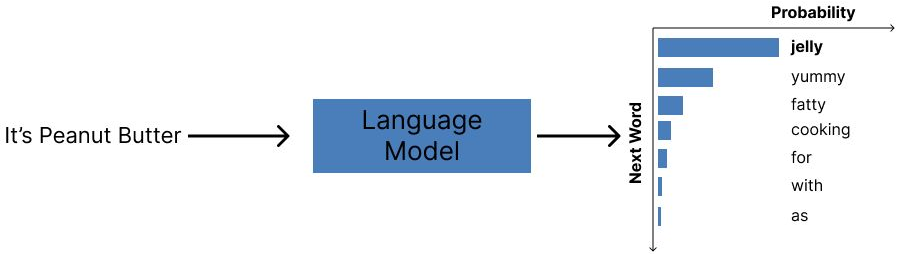


How does a Language Model **generate Text**?

P(Next-Word | It's, Peanut) = Softmax(Logits)

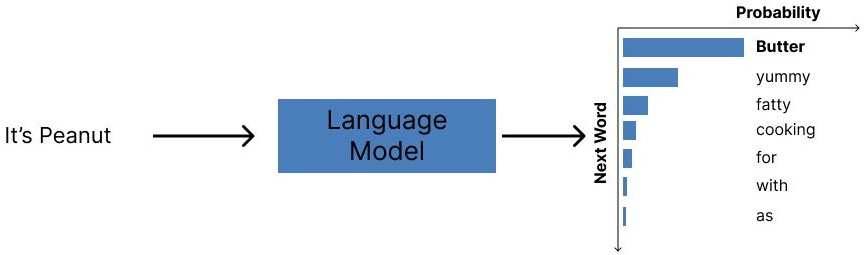


P(Next-Word | It's, Peanut, Butter) = Softmax(Logits)

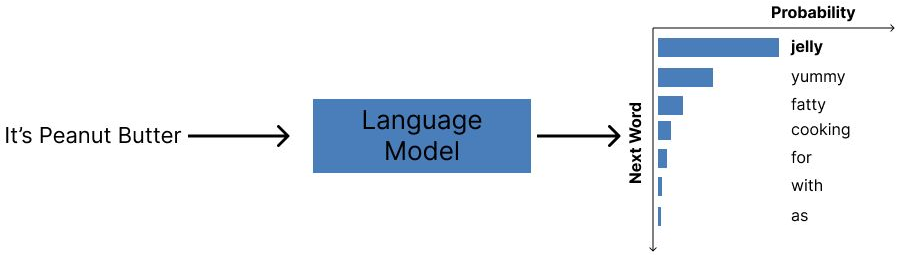


How does a Language Model **generate Text**?

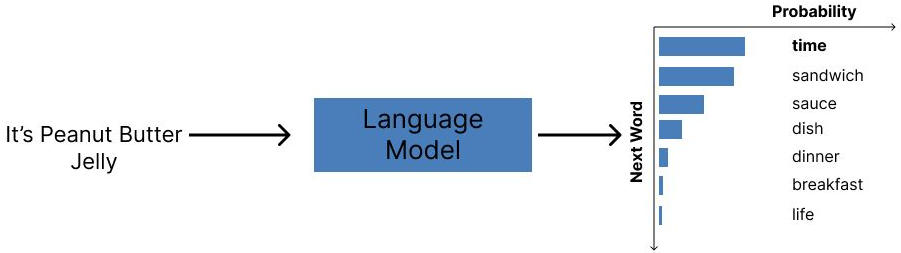
P(Next-Word | It's, Peanut) = Softmax(Logits)



P(Next-Word | It's, Peanut, Butter) = Softmax(Logits)

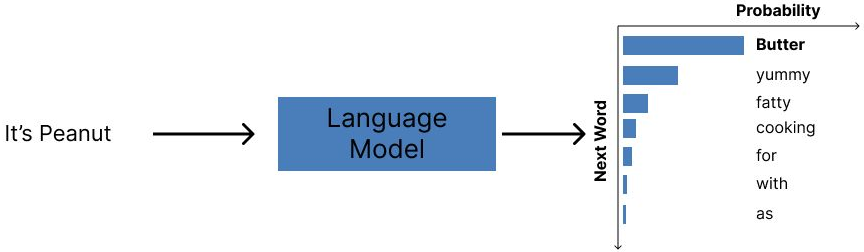


P(Next-Word | It's, Peanut, Butter,Jelly) = Softmax(Logits)

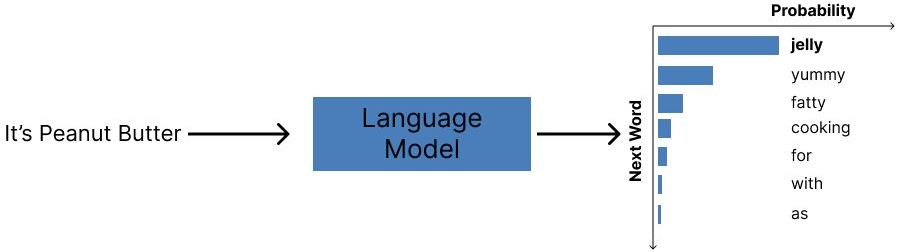


How does a Language Model **generate Text**?

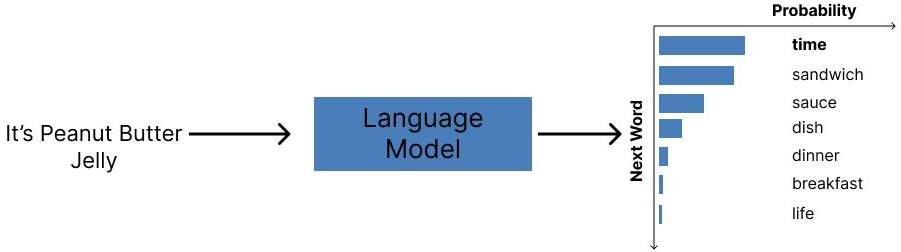
P(Next-Word | It's, Peanut) = Softmax(Logits)



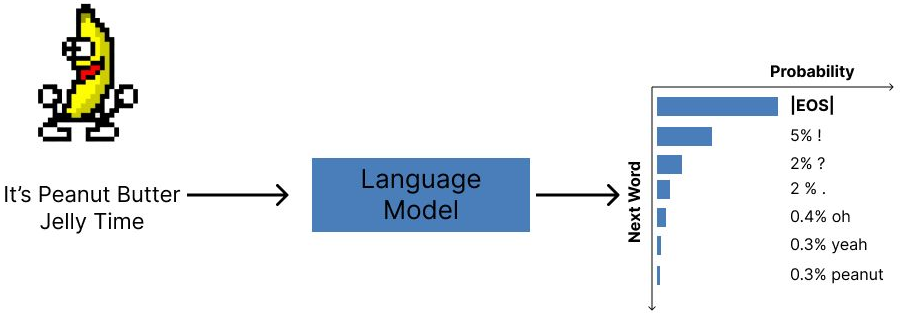
P(Next-Word | It's, Peanut, Butter) = Softmax(Logits)



P(Next-Word | It's, Peanut, Butter,Jelly) = Softmax(Logits)



P(Next-Word | It's, Peanut, Butter) = Softmax(Logits)

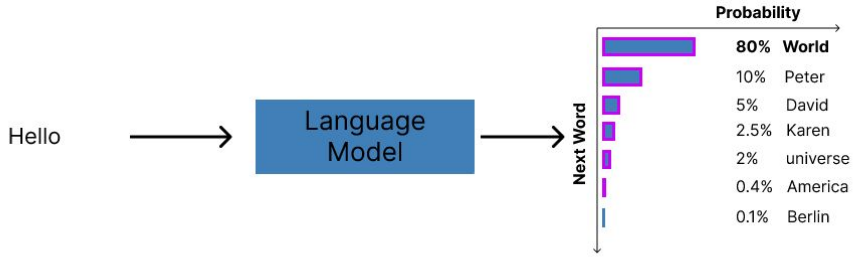


How does a Language Model **generate Text**?

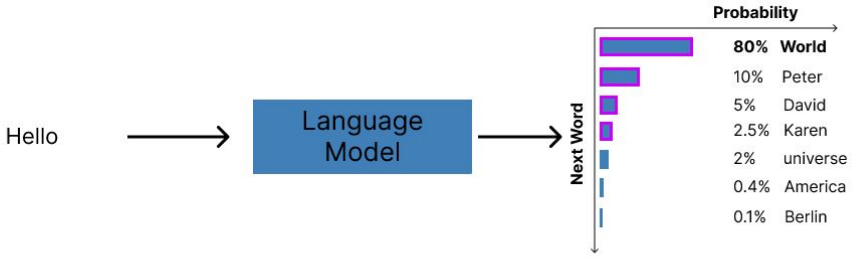
What are some **Sampling Algorithms** and **Parameters** ?

- **Parameters:**
  - Repetition Penalty: Penalize generation candidates based on how often they already exist based See CTRL
  - Max Repeated NGrams: Block some NGrams to occur more than K times
  - Temperature: Reshape distributed given by Softmax
  - Min/Max Output Length: Limit the length of generated
- **Sampling Algorithms**:
  - ArgMax based : Pick word with highest probability of occurring next, Greedy Algorithm.
    - Makes generation deterministic, i.e. always generates the same text for the same prompt
    - When **.doSample(False)** our will use this algorithm and all **previous examples used this algorithm**
  - Top-K Sampling: choose from **Top-K highest probability** words in distribution
  - Top-P Sampling: choose from words with **summed probability** equal or greater than **P**

Top-K Sampling
Choose from first K word with highest probability
For K=6



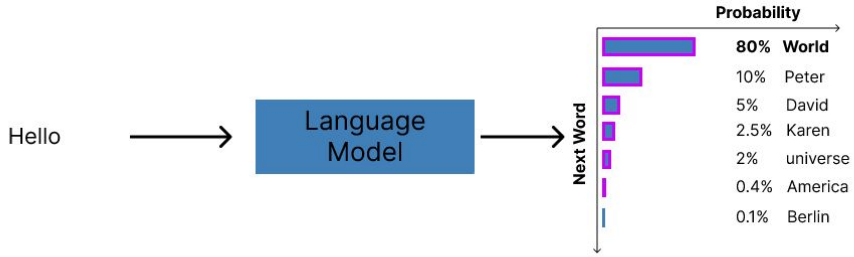
Top-P Sampling
Choose from first tokens whoose collective probability is at least P
For P = 0.95



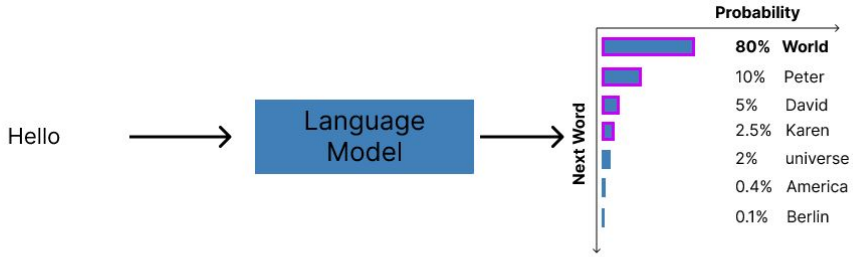
What's the difference between **Top-P** and **Top-K Sampling**?

- **Higher K and P:** makes choices more diverse and risky. Less logical and coherent
- **Lower K and P:** makes choices more generic and safe Also more logical and coherent
- Softmax normalizes logits between 0 and 1. We can tweak the function with the T parameter
- **High T:** Make Distribution more uniform and diverse.
  Associated with creativity and randomness of generations.
- **Low T**: Make Distribution concentrate around top words, more spiky and less diverse.
  Associated with more logical and coherent generations.

Top-K Sampling
Choose from first K word with highest probability
For K=6



Top-P Sampling
Choose from first tokens whoose collective probability is at least P
For P = 0.95



What is **Prompt Engineering**?

- Prompt Engineering is the art of **finding prompts** which shape the conditioned probability $p(x \mid y)$ such that only tokens revelant to a specific use case have a high chance of beeing sampled

```
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.

Let’s go back to the notebook and apply what we just learned

GPT2 NLU Notebook

What is **Data Augmentation** with Language Models?

- Since the largest Language Modes are trained on almost the entire Internet as text corpus, they can generate realistic text for almost any topic.

- We can **improve accuracy** of most NLP Classifiers by **augmenting our dataset** generated examples
  - Especially useful when only very little training data is available and creating new by hand is tedious
- The newly NLP classifier is likely much faster and smaller than the large Language Model
  - Essentially a distilled version or the Language Models knowledge

- How to **create more training** data and Augmented Dataset:
  - Use the label itself as prompt and suffix it with a chunk of the original text.
    Try to use chunks which have enough semantic information to justify a prediction of the label.
    One of the most simple methods is to use use the first N token or characters as suffix to the label

- The Language Model could be used for training, but the trained model is much faster, efficient and possibly more accurate

How to create **more training** data and **Augmented Dataset**?

- There are dozens of techniques and approaches, see A Survey of Data Augmentation Approaches for NLP
- For our simple demo we will :
  - Use the label itself as prompt and suffix it with a chunk of the original text.
    Try to use chunks which have enough semantic information to justify a prediction of the label.
    One of the most simple methods is to use use the first N token or characters as suffix to the label

DA Method	Ext.Know	Pretrained	Preprocess	Level	Task-Agnostic
SYNONYM REPLACEMENT (Zhang et al., 2015)	✓	×	tok	Input	✓
RANDOM DELETION (Wei and Zou, 2019)	×	×	tok	Input	✓
RANDOM SWAP (Wei and Zou, 2019)	×	×	tok	Input	✓
BACKTRANSLATION (Sennrich et al., 2016)	×	✓	Depends	Input	✓
SCPN (Wieting and Gimpel, 2017)	×	✓	const	Input	✓
SEMANTIC TEXT EXCHANGE (Feng et al., 2019)	×	✓	const	Input	✓
CONTEXTUALAUG (Kobayashi, 2018)	×	✓	-	Input	✓
LAMBADA (Anaby-Tavor et al., 2020)	×	✓	-	Input	×
GECA (Andreas, 2020)	×	×	tok	Input	×
SEQMIXUP (Guo et al., 2020)	×	×	tok	Input	×
SWITCHOUT (Wang et al., 2018b)	×	×	tok	Input	×
EMIX (Jindal et al., 2020a)	×	×	-	Emb/Hidden	✓
SPEECHMIX (Jindal et al., 2020b)	×	×	-	Emb/Hidden	Speech/Audio
MIXTEXT (Chen et al., 2020c)	×	×	-	Emb/Hidden	✓
SIGNEDGRAPH (Chen et al., 2020b)	×	×	-	Input	×
DTREEMORPH (Şahin and Steedman, 2018)	×	×	dep	Input	✓
Sub^2 (Shi et al., 2021)	×	×	dep	Input	Substructural
DAGA (Ding et al., 2020)	×	×	tok	Input+Label	×
WN-HYPERS (Feng et al., 2020)	✓	×	const+KWE	Input	✓
SYNTHETIC NOISE (Feng et al., 2020)	×	×	tok	Input	✓
UEDIN-MS (DA part) (Grundkiewicz et al., 2019)	✓	×	tok	Input	✓
NONCE (Gulordava et al., 2018)	✓	×	const	Input	✓
XLDA (Singh et al., 2019)	×	✓	Depends	Input	✓
SEQMIX (Zhang et al., 2020)	×	✓	tok	Input+Label	×
SLOT-SUB-LM (Louvan and Magnini, 2020)	×	✓	tok	Input	✓
UBT & TBT (Vaibhav et al., 2019)	×	✓	Depends	Input	✓
SOFT CONTEXTUAL DA (Gao et al., 2019)	×	✓	tok	Emb/Hidden	✓
DATA DIVERSIFICATION (Nguyen et al., 2020)	×	✓	Depends	Input	✓
DIPS (Kumar et al., 2019a)	×	✓	tok	Input	✓
AUGMENTED SBERT (Thakur et al., 2021)	×	✓	-	Input+Label	Sentence Pairs

Table 1: Comparing a selection of DA methods by various aspects relating to their applicability, dependencies, and requirements. *Ext.Know*, *KWE*, *tok*, *const*, and *dep* stand for External Knowledge, keyword extraction, tokenization, constituency parsing, and dependency parsing, respectively. *Ext.Know* refers to whether the DA method requires external knowledge (e.g. WordNet) and *Pretrained* if it requires a pretrained model (e.g. BERT). *Preprocess* denotes preprocessing required, *Level* denotes the depth at which data is modified by the DA, and *Task-Agnostic* refers to whether the DA method can be applied to different tasks. See Appendix B for further explanation.

A Survey of Data Augmentation Approaches for NLP

Why does this work?

- **Universal Approximation Theorem** states: More data = more accuracy
  Holds for Neural Networks and many ML models
- Leverage this when we augment dataset size with generative models
- Bad generation can make our model perform worse though, we need to be careful how and what is generated

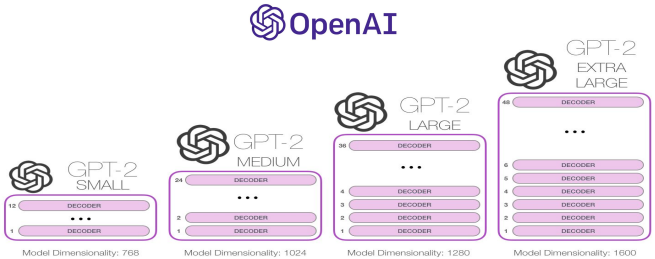
Data Augmentation in Action, apply what we just learned

GPT2 NLU Notebook

Text Generating Sequence to Sequence Transformers in in NLU & Spark NLP

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

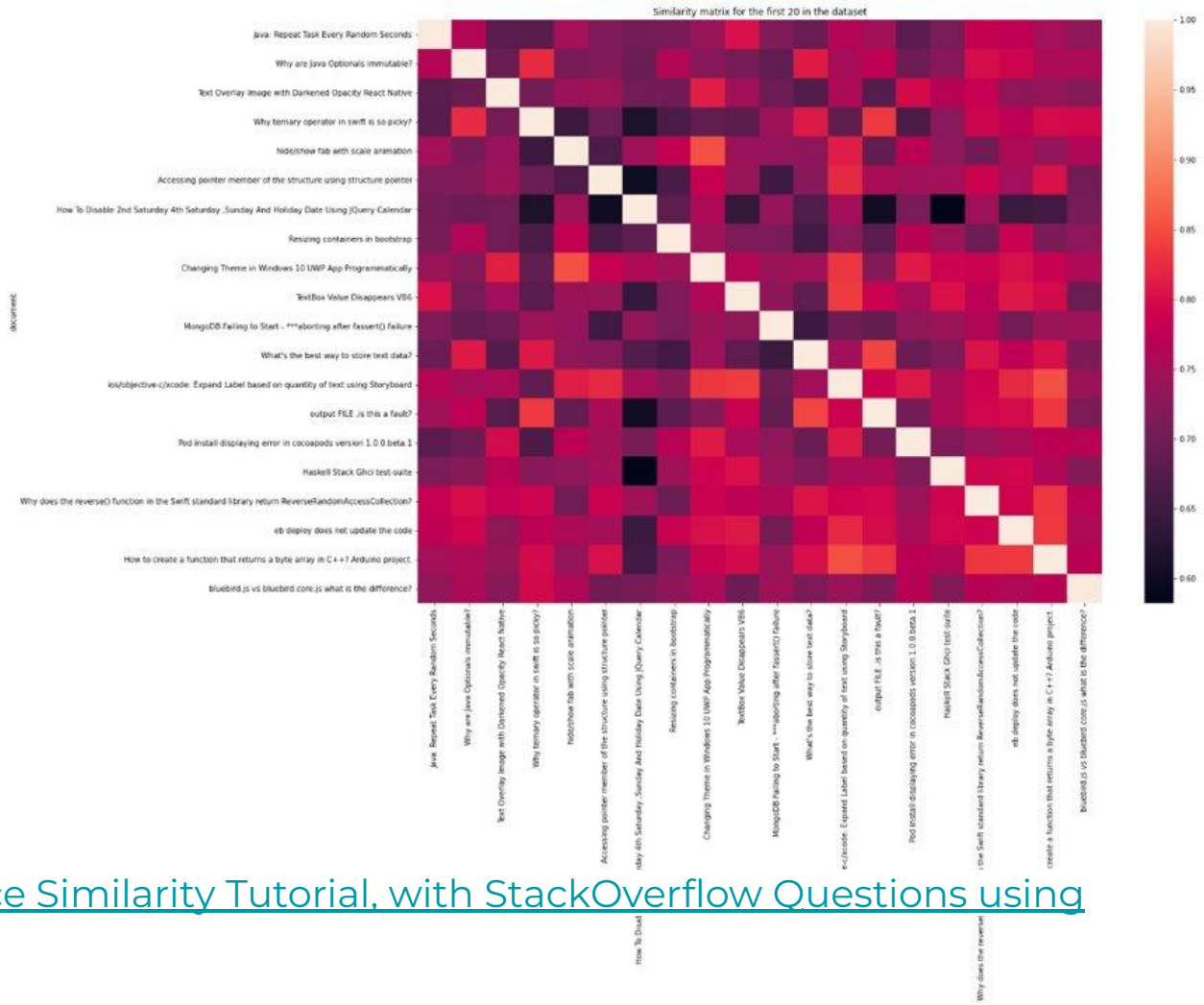


Summarize, answer questions, generate SQL and more with Google's T5
Showcased in this tutorial Webinar

Generate long sequences of text, Augment Text and more with Open AI's GPT2
Showcased today

Translate between over 200 languages with Microsoft's MarianNMT
showcased on chinese in this tutorial webinar

Sentence Similarity With BERTology Embeds or T5



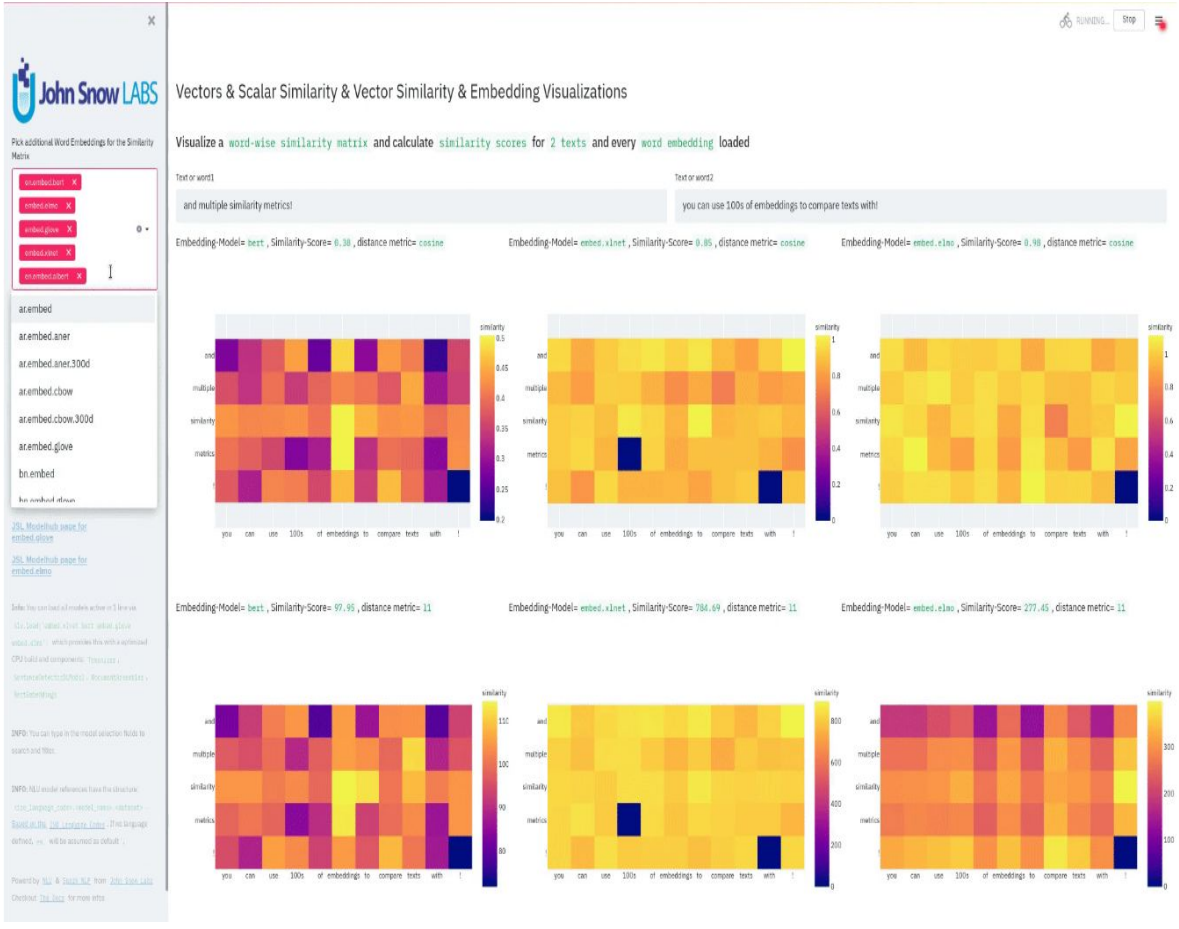
[Indepth and Easy Sentence Similarity Tutorial, with StackOverflow Questions using BERTology embeddings]

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]

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

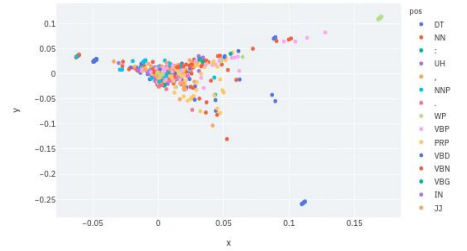
- Compare Multiple similarity Metrics And Embeddings at the same time
- Supported similarities :
  - Cosine
  - Cityblock
  - Euclidean
  - L2
  - L1
  - Manhattan



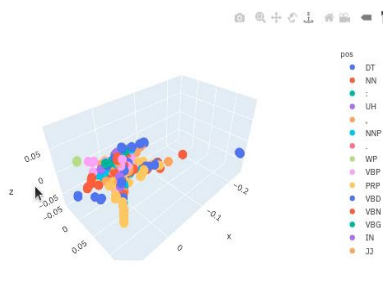
Explore 100+ Embeddings via 10+ Manifold and Matrix Decomposition with 0 lines of code with NLU & Streamlit

Compare multiple dimension reduction
Techniques and
Embeddings at the same time

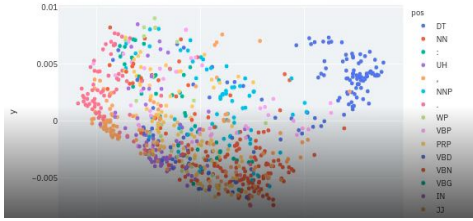
Word-Embeddings = `small_bert_L2_128` , Manifold-Algo = `LLE` for `D=2`



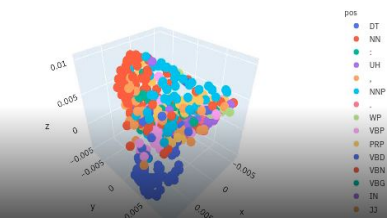
Word-Embeddings = `small_bert_L2_128` , Manifold-Algo = `LLE` for `D=3`

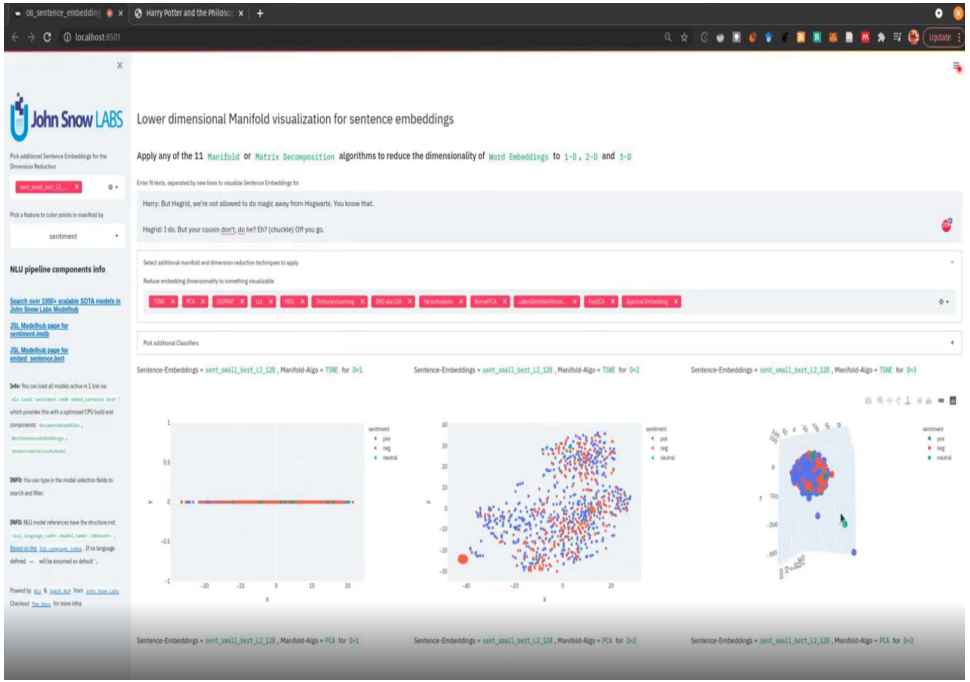


Word-Embeddings = `small_bert_L2_128` , Manifold-Algo = `Spectral Embedding` for `D=2`



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()
```

- Extract text from
  - **Images**
  - **PDFs**
  - **DOC/DOCX**
- Extract tables to Pandas Dataframes from
  - **PDFs**
  - **DOC/DOCX**
  - **PPT**

Overview of all OCR features in NLU

Overview of OCR Text Extractors

These models grab the text directly from your input file and returns it as a Pandas DataFrame

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

Overview of OCR Table Extractors

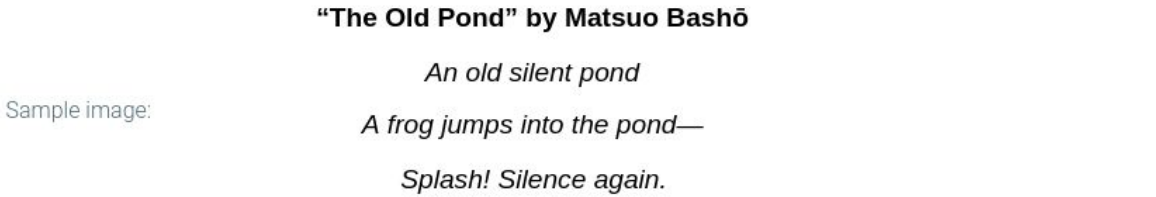
These models grab all Table data from the files detected and return a `list of Pandas DataFrames`, containing Pandas DataFrame for every table detected

NLU Spell	Transformer Class
nlu.load(`pdf2table`)	**PdfToTextTable**
nlu.load(`ppt2table`)	**PptToTextTable**
nlu.load(`doc2table`)	**DocToTextTable**

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

Image to Text

Sample image:




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

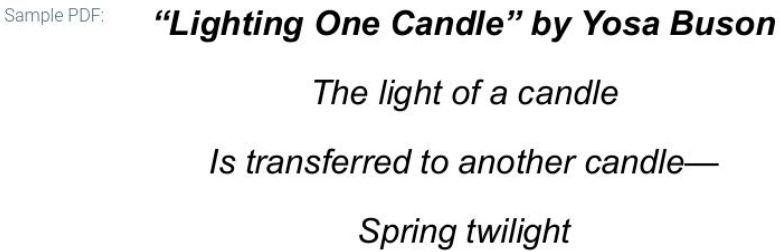
Output of IMG OCR:

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.

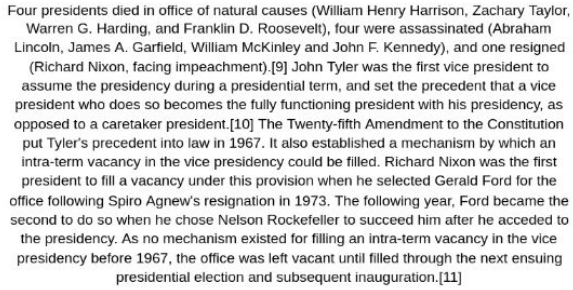
```
nlu.load('doc2text').predict('path/to/haiku.docx')
```

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

OCR Tables from PDFs, DOC, DOCX and PPT files in 1 Line of code with NLU 4.0

```
[ ] nlu.load('pdf2table').predict('/content/tables.pdf')
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear
1	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4
2	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4
3	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4
4	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3
5	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3
6	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3
7	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3
8	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4
9	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4
10	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4
11	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4
12	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3
13	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3
14	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3
15	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3
16	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3
17	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3
18	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4
19	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4
20	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4

sample.pdf

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4

Christian Kasim Loan
Lead Data Scientist
christian@johnsnowlabs.com

Overview of all OCR features in NLU

Overview of OCR Text Extractors

These models grab the text directly from your input file and returns it as a Pandas DataFrame

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

Overview of OCR Table Extractors

These models grab all Table data from the files detected and return a `list of Pandas DataFrames`, containing Pandas DataFrame for every table detected

NLU Spell	Transformer Class
nlu.load(`pdf2table`)	**PdfToTextTable**
nlu.load(`ppt2table`)	**PptToTextTable**
nlu.load(`doc2table`)	**DocToTextTable**

More info on
https://nlu.johnsnowlabs.com/docs/en/nlu_for_ocr

More NLU Ressources

- Join our Slack
- NLU Website
- NLU Github
- 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+ Healthcare models in NLU
- Multi Lingual NLU Webinar - Working with Chinese Text Data
- 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

Thank you.


christian@JohnSnowLabs.com


@ckl_it


In/Christian-Kasim-Loan

Medium.com/@Christian.Kasim.Loan