




Transforming Text Classification with Hugging Face

MSBA 6331 - Group 5

Dat Luong - Yiwei Tang (Tony) - Amulya Konda - Pranjali Gaikwad - Jiaqing Zhang (Doris)



Current Issue Machine Learning Model



Big Data



Maintain & Monitor



Data Pattern Change



Stakeholders



Testing



Security



87%

Data Science Project **NOT** making it to production

*According to the Algorithmia's "2020 State of Enterprise ML" survey



87%

Data Science Project **NOT** making it to production



Solve some of these problems with pre-deployed model

*According to the Algorithmia's "2020 State of Enterprise ML" survey

Introducing

Hugging Face

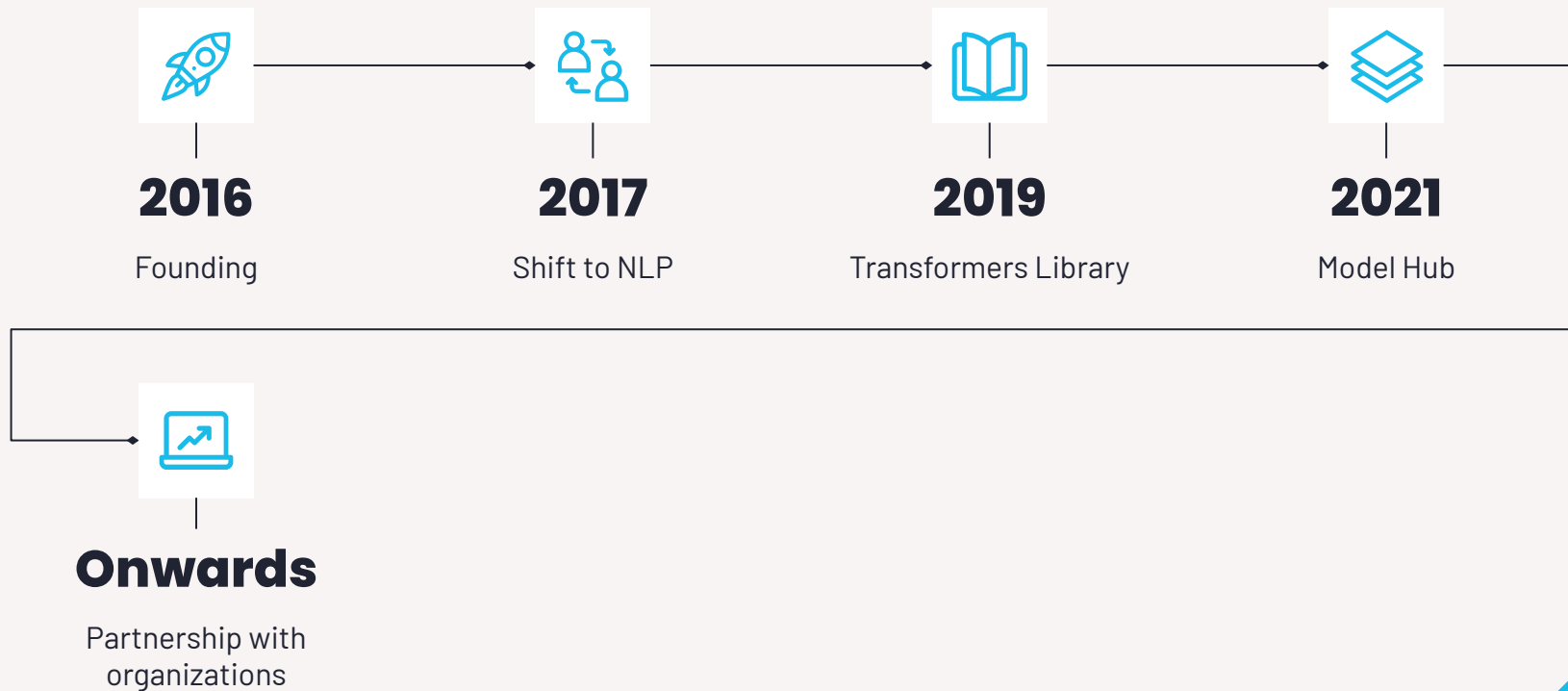




Hugging Face

- Leading company in the field of Natural Language Processing
- Development and democratization of state-of-the-art NLP models.
- Provides a collection of pre-trained models and tools for working with various NLP tasks
- Strong commitment to open source development
- Emphasizes the importance of ethical AI and responsible development practices.

Hugging Face Development Process



Hugging Face Use Case



NLU Chatbots

Developing chatbots with advanced natural language understanding capacities



Text Classification

Text classification tasks, such as sentiment analysis, spam detection, and topic categorization



Text Summarization

Analyze and summarize text from papers, articles, medical text, legal documents, etc.



Financial Sentiment Analysis

Analyze news articles, social media, and other text data to gauge market sentiment and make informed investment decisions

Hugging Face raises \$235M from investors, including Salesforce and Nvidia

Kyle Wiggers @kyle_l_wiggers / 9:00 AM CDT • August 24, 2023



Comr

Dell Technologies and Hugging Face to Simplify Generative AI with On-Premises IT

DELLTechnologies

NEWS PROVIDED BY

[Dell Technologies](#) →

14 Nov, 2023, 09:00 ET

SHARE THIS ARTICLE



TECH

Google, Amazon, Nvidia and other tech giants invest in AI startup Hugging Face, sending its valuation to \$4.5 billion

PUBLISHED THU, AUG 24 2023•10:00 AM EDT | UPDATED THU, AUG 24 2023•11:35 AM EDT



Kif Leswing
@KIFLESWING

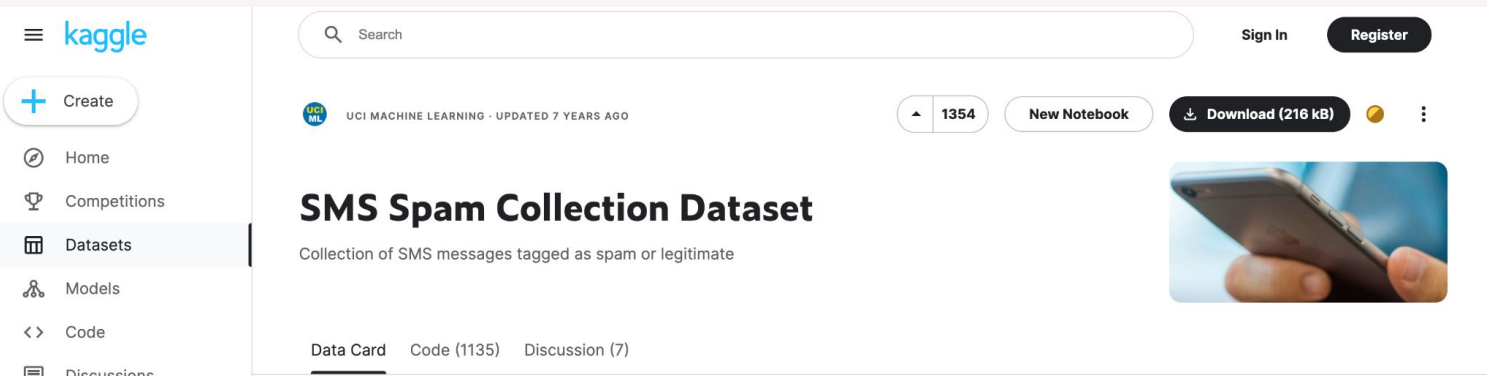
SHARE



NEWS

Case Study and Dataset

- We want to compare the following thing:
 - The process of building the model
 - Running speed of the model
 - Accuracy of the model
 - User friendliness

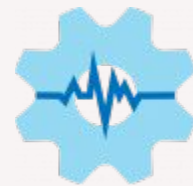


The screenshot shows the Kaggle website interface. On the left is a sidebar with navigation links: Home, Competitions, Datasets (highlighted), Models, Code, and Discussions. The main content area displays the 'SMS Spam Collection Dataset' by UCI Machine Learning, updated 7 years ago. It has 1354 votes and a 'New Notebook' button. A 'Download (216 kB)' button is also visible. Below the title, it says 'Collection of SMS messages tagged as spam or legitimate'. At the bottom, there are tabs for 'Data Card', 'Code (1135)', and 'Discussion (7)'. A search bar is at the top, and 'Sign In' and 'Register' buttons are on the right. A small image of a hand holding a smartphone is shown on the right side of the dataset card.

- + SMS Spam Research
- + 5574 English Message
- + Tag as Ham/Spam

“Control Object”: Spark NLP

- NLP Library built on Apache Spark, an open source and distributed computing system
- Developed by John Snow Labs
- Scaled horizontally across a cluster of machines
- Offers pre-trained models for multiple NLP tasks (such as text classification, and sentiment analysis)



Spark NLP

Comparison between Hugging Face and SparkNLP

Hugging Face	SparkNLP
Emphasizes pre-trained models, model fine-tuning	Geared towards scalable and efficient NLP processing in big data environments
Large and active community of developers and researchers	Community support and engagement, but may not be as extensive as Hugging Face's.
Simple API for using models in various NLP applications	Requires knowledge of Spark for efficient utilization
Offers a diverse set of pre-trained models, including BERT, GPT-2, GPT-3, and more.	Provides models and pipelines specifically designed for Apache Spark.



Code and Result Demonstration

Spark NLP with Bert Classification model

```
import sparknlp
from sparknlp.base import *
from sparknlp.annotator import *
from pyspark.ml import Pipeline

# Initialize Spark NLP components
document_assembler = DocumentAssembler() \
    .setInputCol("text") \
    .setOutputCol("document")

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

bert_embeddings = BertEmbeddings.pretrained("bert_base_uncased", "en") \
    .setInputCols(["document", "token"]) \
    .setOutputCol("embeddings")

sentence_embeddings = SentenceEmbeddings() \
    .setInputCols(["document", "embeddings"]) \
    .setOutputCol("sentence_embeddings") \
    .setPoolingStrategy("AVERAGE")

classifier = ClassifierDLApproach() \
    .setInputCols(["sentence_embeddings"]) \
    .setOutputCol("class") \
    .setLabelColumn("label") \
    .setMaxEpochs(5) \
    .setBatchSize(8)

# Define the pipeline
pipeline = Pipeline().setStages([
    document_assembler,
    tokenizer,
    bert_embeddings,
    sentence_embeddings,
    classifier
])
```

```
from pyspark.sql.functions import col
from pyspark.sql.types import DoubleType
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Ensure the prediction column is of type double
predictions = predictions.withColumn("prediction", col("prediction").cast(DoubleType()))

# Ensure the label column is also of type double
predictions = predictions.withColumn("label", col("label").cast(DoubleType()))

# Select the prediction and true label
predictions = predictions.select(col("prediction"), col("label"))

# Evaluate the model
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction", labelCol="label", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print(f"Accuracy: {accuracy}")
```

Accuracy: 0.8682242990654205

Spark NLP Bert embeddings

[Home](#)[Docs](#)[Models](#)[Demo](#)[Blog](#)[Star on GitHub](#)

BertEmbeddings

Token-level embeddings using BERT. BERT (Bidirectional Encoder Representations from Transformers) provides dense vector representations for natural language by using a deep, pre-trained neural network with the Transformer architecture.

Pretrained models can be loaded with `pretrained` of the companion object:

```
val embeddings = BertEmbeddings.pretrained()
  .setInputCols("token", "document")
  .setOutputCol("bert_embeddings")
```

The default model is `"small_bert_L2_768"`, if no name is provided.

For available pretrained models please see the [Models Hub](#).

For extended examples of usage, see the [Examples](#) and the [BertEmbeddingsTestSpec](#).

Sources :

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#)

<https://github.com/google-research/bert>

Hugging Face pre-trained model

```
from transformers import AutoTokenizer

# Load the tokenizer for "distilbert-base-uncased" model.
tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased")
def tokenize_function(examples):
    # Pad/truncate each text to 512 tokens. Enforcing the same shape
    # could make the training faster.
    return tokenizer(
        examples["text"],
        padding="max_length",
        truncation=True,
        max_length=128,
    )

train_tokenized = train_dataset.map(tokenize_function).remove_columns(["text"]).shuffle(seed=42)
test_tokenized = test_dataset.map(tokenize_function).remove_columns(["text"]).shuffle(seed=42)
```

Map: 100%  4468/4468 [00:02<00:00, 1444.49 examples/s]

Map: 100%  1106/1106 [00:00<00:00, 1352.63 examples/s]

```
from transformers import AutoModelForSequenceClassification

model = AutoModelForSequenceClassification.from_pretrained(
    "distilbert-base-uncased",
    num_labels=2,
    label2id=label2id,
    id2label=id2label,
)
```

```
accuracy = eval_result['eval_accuracy']
```

```
# Now print the accuracy
print(f"Accuracy: {accuracy}")
```

Accuracy: 0.9945750452079566

Hugging Face – AutoModels

AutoModel
AutoModelForPreTraining
AutoModelWithLMHead
AutoModelForSequenceClassification
AutoModelForQuestionAnswering
AutoModelForTokenClassification
Encoder Decoder Models
BERT
OpenAI GPT
Transformer XL
OpenAI GPT2
XLNet
XLNet
RoBERTa
DistilBERT
CTRL
CamemBERT
ALBERT
XLNet-RoBERTa
FlauBERT
Bart
T5
ELECTRA
DialoGPT
Reformer
MarianMT
Unbabel

AutoModelForSequenceClassification

class transformers.AutoModelForSequenceClassification [\[SOURCE\]](#)

[AutoModelForSequenceClassification](#) is a generic model class that will be instantiated as one of the sequence classification model classes of the library when created with the `AutoModelForSequenceClassification.from_pretrained(pretrained_model_name_or_path)` class method.

This class cannot be instantiated using `__init__` (throws an error).

classmethod `from_config` (*config*) [\[SOURCE\]](#)

Instantiates one of the base model classes of the library from a configuration.

Note

Loading a model from its configuration file does **not** load the model weights. It only affects the model's configuration. Use `from_pretrained()` to load the model weights

Parameters

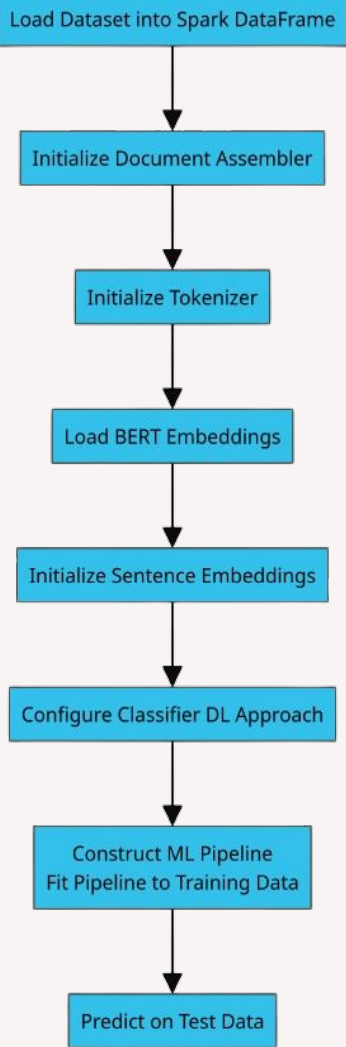
config ([PretrainedConfig](#)) –

The model class to instantiate is selected based on the configuration class:

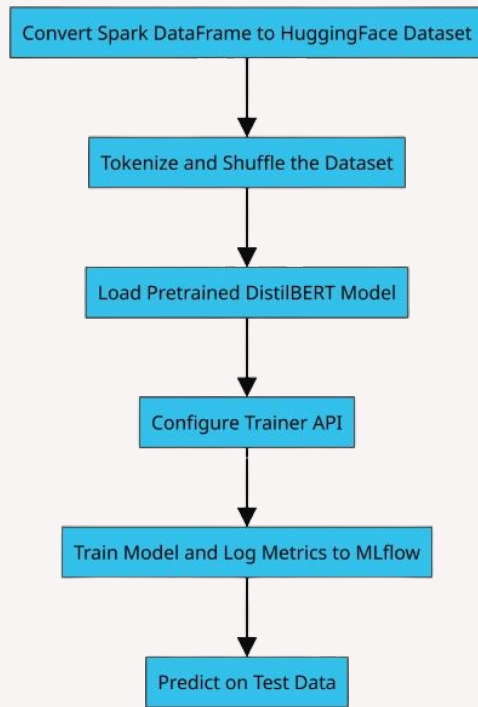
- Is instance of `distilbert` configuration class: [DistilBertForSequenceClassification](#) (DistilBERT model)
- Is instance of `albert` configuration class: [AlbertForSequenceClassification](#) (ALBERT model)
- Is instance of `camembert` configuration class: [CamembertForSequenceClassification](#) (CamemBERT model)
- Is instance of `xlm_roberta` configuration class: [XLMRobertaForSequenceClassification](#) (XLM-RoBERTa model)
- Is instance of `roberta` configuration class: [RobertaForSequenceClassification](#) (RoBERTa model)
- Is instance of `bert` configuration class: [BertForSequenceClassification](#) (Bert model)
- Is instance of `xlnet` configuration class: [XLNetForSequenceClassification](#) (XLNet model)
- Is instance of `xlm` configuration class: [XLMForSequenceClassification](#) (XLM model)
- Is instance of `flaubert` configuration class: [FlaubertForSequenceClassification](#) (Flaubert model)

Process steps for Spark NLP vs Hugging Face

SparkNLP



Hugging Face



Hugging Face Practicality

More than 50,000 organizations are using Hugging Face



Allen Institute for AI

Non-Profit • 228 models



AI at Meta

Company • 1877 models



Amazon Web Services

Company • 10 models



Google

Company • 617 models



Intel

Company • 134 models



Microsoft

Company • 263 models



Grammarly

Company • 6 models



Writer

Company • 8 models

Enterprise

COMPETITORS



Anthropic



Stability AI



deepset



Cohere



OpenAI

Future Direction

1. We can see that Hugging Face face many challenges, and one of them is run time, so we can try to find other solutions that can give us the ease of Hugging Face (250.6172 seconds) but the speed of Spark NLP (123.48 seconds)
2. Create another case study for Hugging Face to test its ability using different dataset with different data constraint



Conclusion + Lesson Learnt

- Beneficial using pre-trained/pre-deployed model
 - Thorough testing
 - Less maintenance
 - Security
- Ease model development process
- Rapid prototyping
- Community powered by Model Hub



Hugging Face

References & Appendix

BERT embeddings of Spark NLP

<https://sparknlp.org/docs/en/transformers#bertembeddings>

AutoModelForSequenceClassification:

https://huggingface.co/transformers/v2.11.0/model_doc/auto.html#automodelforsequenceclassification

Getting started with NLP using Hugging Face transformers

pipelines: <https://www.databricks.com/blog/2023/02/06/getting-started-nlp-using-hugging-face-transformers-pipelines.html>

Hugging Face:

<https://www.techtarget.com/whatis/definition/Hugging-Face#:~:text=Hugging%20Face%20provides%20access%20to%20a%20vast%20community%2C%20continuously%20updated,Face's%20hosted%20models%20saves%20money>