

Pipelines with Hugging Face

WORKING WITH HUGGING FACE



Jacob H. Marquez
Lead Data Engineer

Use in transformers

The screenshot shows the Hugging Face Model card for the model `distilbert-base-uncased-finetuned-sst-2-english`. The card includes the following details:

- Model card:** DistilBERT base uncased finetuned SST-2
- Downloads last month:** 21,019,589
- Safetensors:** Model size 67M params
- Tensor type:** F32

Below the card, there is a sidebar with the following navigation links:

- Table of Contents
 - Model Details
 - How to Get Started With the Model

¹ <https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english>

Use in transformers

The screenshot shows a dark-themed web page from Hugging Face. At the top, there's a navigation bar with links for 'Models', 'Datasets', 'Spaces', 'Docs', 'Solutions', and 'Pricing'. A search bar is also present. On the right side of the header is a user profile icon.

The main content area displays code snippets for using the 'distilbert-base-uncased-finetuned-sst-2-english' model. The first snippet uses a pipeline:

```
# Use a pipeline as a high-level helper
from transformers import pipeline
```

The second snippet loads the model directly:

```
# Load model directly
from transformers import AutoTokenizer, AutoModelForSequenceClassification
```

This second snippet is highlighted with a green border and has a 'Copy' button to its right.

At the bottom left, there's a 'Quick Links' section with a few items listed.

¹ <https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english>

Use in transformers

The screenshot shows a dark-themed web page from Hugging Face. At the top, there's a navigation bar with links for 'Models', 'Datasets', 'Spaces', 'Docs', 'Solutions', and 'Pricing'. A search bar is also present. On the right side of the header, there's a user profile icon.

The main content area displays code examples for using the `distilbert-base-uncased-finetuned-sst-2-english` model. The first example is highlighted with a green border:

```
# Use a pipeline as a high-level helper
from transformers import pipeline

pipe = pipeline("text-classification", model="distilbert-base-uncased-finetuned-sst-2-english")
```

Below this, another example is shown:

```
# Load model directly
from transformers import AutoTokenizer, AutoModelForSequenceClassification

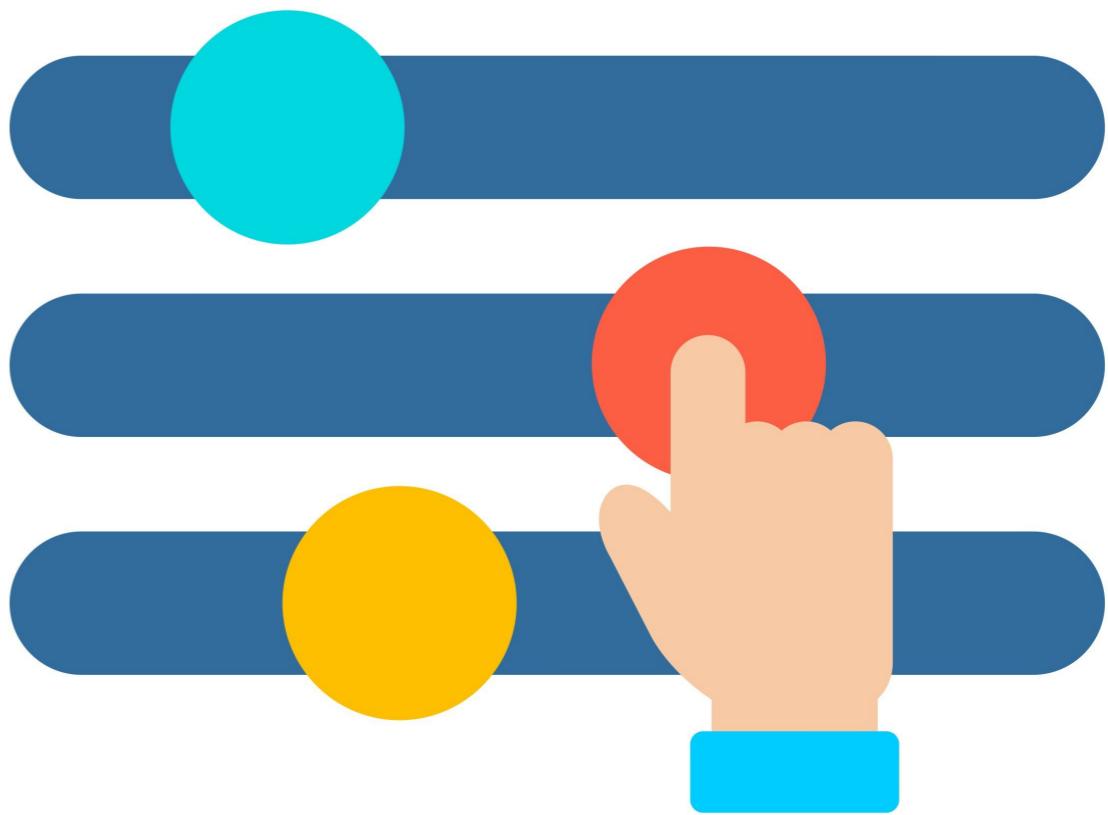
tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased-finetuned-sst-2-english")
model = AutoModelForSequenceClassification.from_pretrained("distilbert-base-uncased-finetuned-sst-2-e
```

At the bottom left, there's a 'Quick Links' section with a few items listed.

¹ <https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english>

Auto classes

- General class for using:
 - Models
 - Tokenizers
 - Configurations
 - Processors
 - Feature extractors
- Flexible and direct
- More control for ML tasks



¹ https://huggingface.co/docs/transformers/model_doc/auto

AutoModels

- Auto classes to directly download a model
- AutoModel class for each type of task

```
from transformers import AutoModelForSequenceClassification
model = AutoModelForSequenceClassification.from_pretrained(
    "distilbert-base-uncased-finetuned-sst-2-english"
)
```

AutoTokenizers

- Prepare text input data
- Recommended to use the tokenizer paired with the model

```
from transformers import AutoTokenizer
tokenizer = AutoTokenizer.from_pretrained(
    "distilbert-base-uncased-finetuned-sst-2-english"
)
```

The pipeline module

- Contains all task-specific steps
- Best for quickly performing tasks
- Great for getting started

```
from transformers import pipeline
```

Task pipelines

```
from transformers import (
    SummarizationPipeline,
    TextClassificationPipeline,
    AudioClassificationPipeline,
    ImageSegmentationPipeline
)
```

- Task-specific pipeline for each task
- Leverage Auto classes
- Download model and relevant processing

¹ https://huggingface.co/docs/transformers/main_classes/pipelines

Creating a pipeline

```
my_pipeline = pipeline(task="text-classification")
```

```
my_pipeline = pipeline(model="distilbert-base-uncased-finetuned-sst-2-english")
```

```
my_pipeline = pipeline(  
    task="text-classification",  
    model="distilbert-base-uncased-finetuned-sst-2-english"  
)
```

Creating a pipeline

```
my_pipeline = pipeline(task="text-classification")
```

```
No model was supplied, default to distilbert-base-uncased-finetuned-sst-2-english
```

pipeline with Auto classes

```
from transformers import pipeline, AutoModelForSequenceClassification

model = AutoModelForSequenceClassification.from_pretrained(
    "distilbert-base-uncased-finetuned-sst-2-english"
)

my_pipeline = pipeline(model=model)
```

Using the pipeline

```
from transformers import pipeline

my_pipeline = pipeline(task="text-classification",
                      model="distilbert-base-uncased-finetuned-sst-2-english")
```

```
input = "Hi, welcome to this awesome course!"
```

```
my_pipeline(input)
```

```
[{'label': 'POSITIVE', 'score': 0.9998550415039062}]
```

Let's practice!

WORKING WITH HUGGING FACE

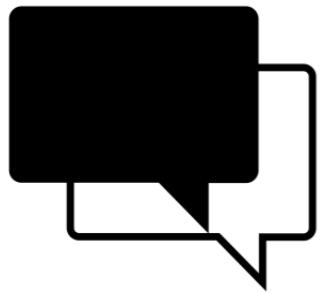
NLP and tokenization

WORKING WITH HUGGING FACE



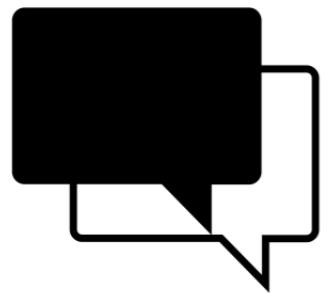
Jacob H. Marquez
Lead Data Engineer

Hugging Face and NLP

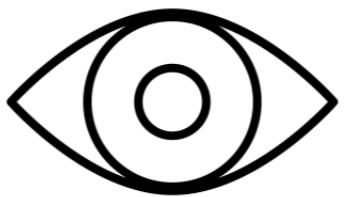


Natural
Language
Processing

Hugging Face and NLP



Natural
Language
Processing



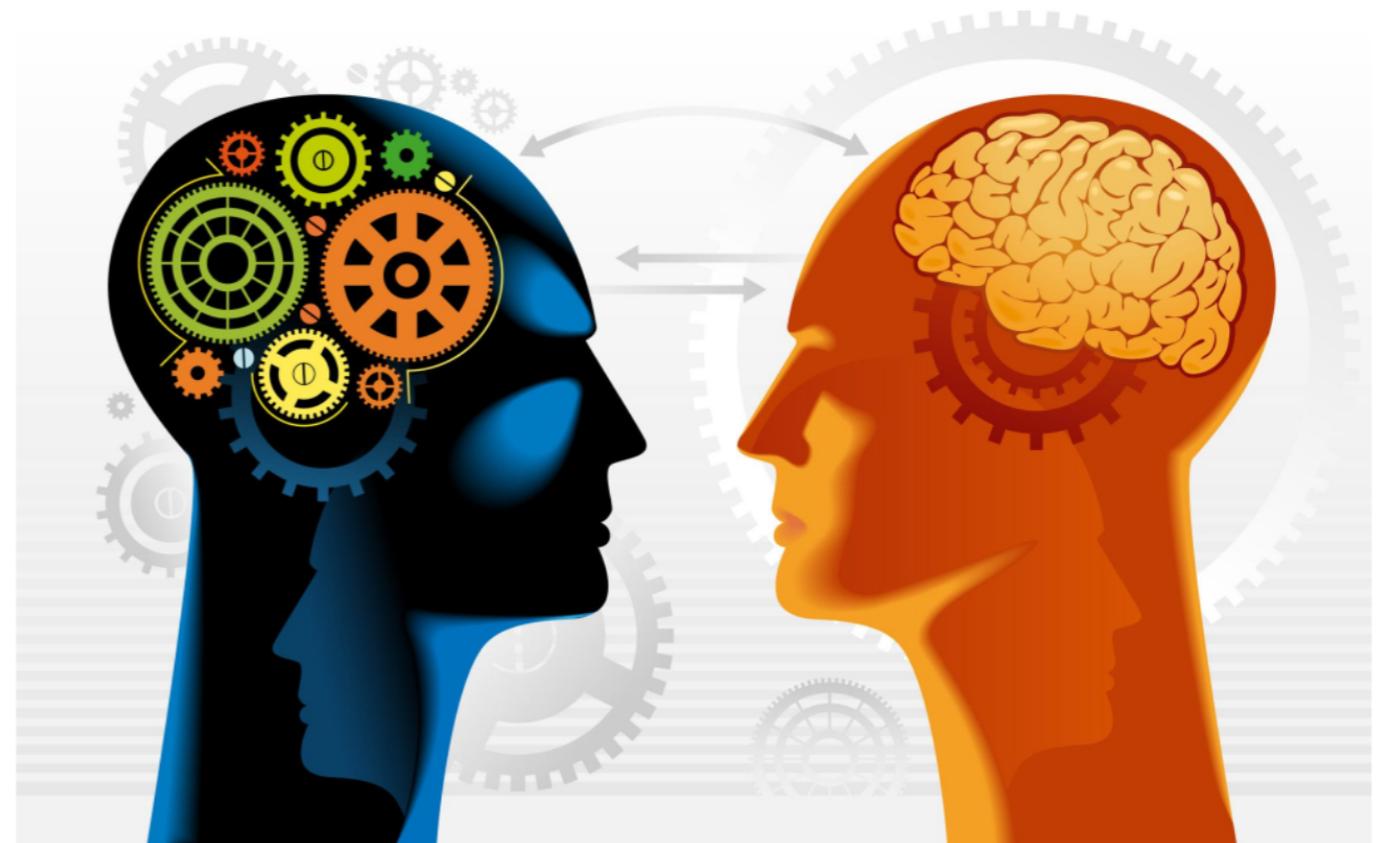
Computer
Vision



Audio

Natural language processing (NLP)

- Subfield of AI
- Enable computers to understand, interpret, and generate human language



Natural language processing

- Subfield of AI
- Enable computers to understand, interpret, and generate human language
- Algorithms, models, and transformers
- Understand words, semantics, contextual nuances



Algorithm

Tokenization

- Converting a sequence into smaller parts
- Numerical form

Sentence: "I am the instructor for this video."

Tokens

["I", "am", "the", "instructor", ..., "video", "."]

Numerical Form

"I" = [0.232, 0.545, 0.876, ..., 0.385]

Tokenization

- Converting a sequence into smaller parts
- Numerical form
- Components:
 - Normalization: transforming and cleaning
 - Pre-tokenization: splitting into smaller tokens
 - Tokenization model
- Tokenization is one of the first steps of NLP
- Supports the model in building contextual knowledge

Normalization

- Cleaning text
- Removing whitespaces

Whitespace

"Hi, my name is Jacob." → "Hi, my name is Jacob."

Normalization

- Cleaning text
- Removing whitespaces
- Accents

Whitespace

"Hi, my name is Jacob." → "Hi, my name is Jacob."

Accents

"Hi, my namé is Jacob." → "Hi, my name is Jacob."

Normalization

- Cleaning text
- Removing whitespaces
- Accents
- Lowercasing

Whitespace

"Hi, my name is Jacob." → "Hi, my name is Jacob."

Accents

"Hi, my **namé** is Jacob." → "Hi, my **name** is Jacob."

Casing

"**Hi**, my name is **Jacob**." → "**hi**, my name is **jacob**."

Normalization

- Cleaning text
- Removing whitespaces
- Accents
- Lowercasing
- Punctuation

Whitespace

"Hi, my name is Jacob." → "Hi, my name is Jacob."

Accents

"Hi, my **namé** is Jacob." → "Hi, my **name** is Jacob."

Casing

"**Hi**, my name is **Jacob**." → "**hi**, my name is **jacob**."

Punctuation

"Hi, my name is Jacob." → "Hi my name is Jacob"

Pre-tokenization

- Input text split into smaller tokens
- Several types of pre-tokenization methods
- Split by whitespace
- Difficulty with languages that don't use spaces separate words

"hi, my name is jacob." → ["hi", "my", "name", "is", "jacob"]

Tokenizer models

- Byte-Pair Encoding, WordPiece, SentencePiece, Unigram
- Each has specific tokenization process
- Goal is to create vocabulary of characters
- Understand most common patterns

Using tokenizers

```
from transformers import AutoTokenizer  
  
tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased")  
  
print(tokenizer.backend_tokenizer.normalizer.normalize_str("HOWDY, how aré yoü?"))
```

```
howdy how are you
```

```
from transformers import pipeline  
  
my_pipeline = pipeline(model="distilbert-base-uncased")
```

Using tokenizers

```
from transformers import GPT2Tokenizer  
input = "HOWDY, how aré yoü?"  
  
gpt_tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
```

```
gpt_tokens = gpt_tokenizer.tokenize(text=input)  
  
print(gpt_tokens)
```

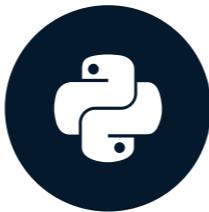
```
'Howdy', 'Ghow', 'Gare', 'Gyou', '?'
```

Let's practice!

WORKING WITH HUGGING FACE

Text classification

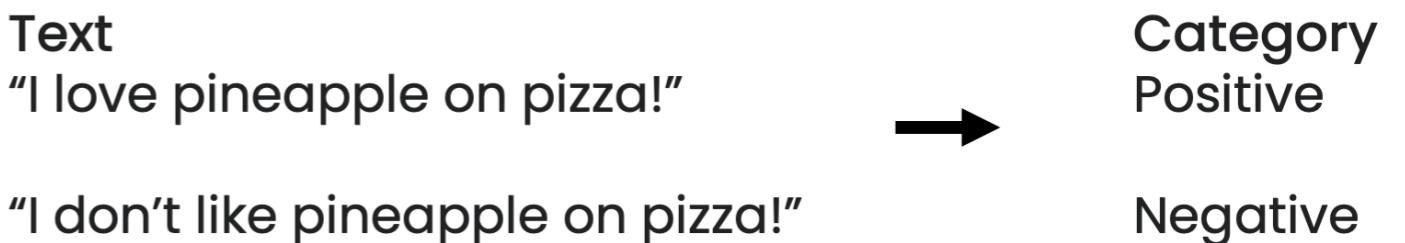
WORKING WITH HUGGING FACE



Jacob H. Marquez
Lead Data Engineer

What is text classification?

- Assign a set of predefined categories to text
- Sentiment analysis



¹ <https://huggingface.co/tasks/text-classification>

What is text classification?

- Assign a set of predefined categories to text
- Sentiment analysis
- Question Natural Language Inference (QNLI)
 - Entailment means true
 - Not entailment means false
 - Neutral means no relationship

Question

"What famous palace
is located in London?"

Premise

"London contains four
World Heritage Sites:
the Tower of London;
Kew Gardens; the site
comprising the Palace
of Westminster."



Category

Not entailment (false)

¹ <https://huggingface.co/tasks/text-classification>

What is text classification?

- Assign a set of predefined categories to text
- Sentiment analysis
- Question Natural Language Inference (QNLI)
- Topic modeling

Text
“The phone died very quick...”



Category
“battery”

What is text classification?

- Assign a set of predefined categories to text
- Sentiment analysis
- Question Natural Language Inference (QNLI)
- Topic modeling
- Grammatical correctness

Text
"This course is great!"
"Course is gravy."



Category
Acceptable
Unacceptable

Challenges of text classification

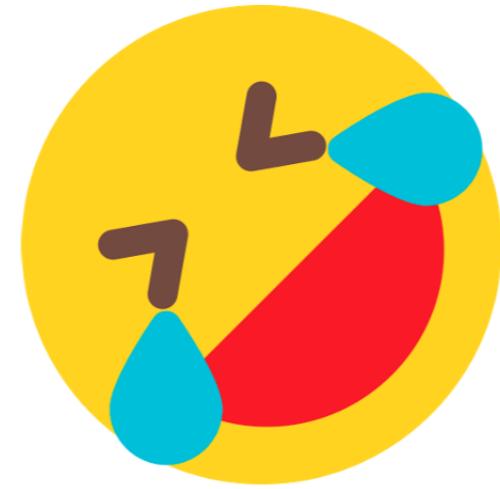


Ambiguity

Challenges of text classification



Ambiguity

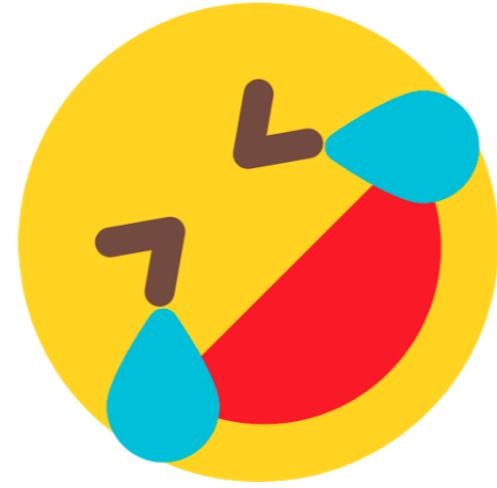


Sarcasm, Irony

Challenges of text classification



Ambiguity



Sarcasm, Irony



Multilingual

Getting started with text classification

```
from transformers import pipeline
```

```
classifier = pipeline(task="text-classification")
```

```
classifier('I love it!')
```

```
[{'label': 'POSITIVE', 'score': 0.9998656511306763}]
```

Grammatical correctness

```
classifier = pipeline(  
    task="text-classification",  
    model="abdulmatinomotoso/English_Grammar_Checker"  
)  
  
classifier("I write cheese strings.")
```

```
[{'label': 'LABEL_0', 'score': 0.95}]
```

QNLI

```
classifier = pipeline(  
    task="text-classification",  
    model="cross-encoder/qnli-electra-base"  
)  
  
classifier("Where is Seattle located?, Seattle is located in Washington state.")
```

```
[{'label': 'LABEL_0', 'score': 0.9978110194206238}]
```

Zero-shot classification

- Transfer learning
- Unseen labels can be determined
- Without specific training
- Helpful when lack of resources to train new model

```
task = "zero-shot-classification"
```

```
modelId = "facebook/bart-large-mnli"
```

```
classifier = pipeline(  
    task=task,  
    model=modelId  
)
```

¹ <https://huggingface.co/tasks/zero-shot-classification>

Zero-shot classification

```
text = "Wikipedia earlier this month released its list of the 25 most viewed...."
```

```
candidate_labels = ['politics', 'science', 'technology']
```

```
output = classifier(text, candidate_labels)
```

```
print(output["labels"][0])
print(output["scores"][0])
```

```
technology
```

```
0.93600781
```

Let's practice!

WORKING WITH HUGGING FACE

Summarization

WORKING WITH HUGGING FACE



Jacob H. Marquez
Lead Data Engineer

What is summarization?

Original Text

David G. Robinson is a data scientist at the Heap analytics company. He is a co-author of the `tidytext` R programming language package and the O'Reilly book, *Text Mining with R*. Robinson has previously worked as a chief data scientist at DataCamp and as a data scientist at Stack Overflow. He was also a data engineer at Flatiron Health in 2019.

What is summarization?

Original Text

David G. Robinson is a data scientist at the Heap analytics company. He is a co-author of the tidytext R programming language package and the O'Reilly book, *Text Mining with R*. Robinson has previously worked as a chief data scientist at DataCamp and as a data scientist at Stack Overflow. He was also a data engineer at Flatiron Health in 2019.



Summarized Text

David G. Robinson is a data scientist. He is a co-author of the tidytext R package and the O'Reilly book.

Extractive versus Abstractive

Original Text

David G. Robinson is a data scientist at the Heap analytics company. He is a co-author of the tidytext R programming language package and the O'Reilly book, *Text Mining with R*. Robinson has previously worked as a chief data scientist at DataCamp and as a data scientist at Stack Overflow. He was also a data engineer at Flatiron Health in 2019.



Summarized Text

David G. Robinson is a data scientist. He is a co-author of the tidytext R package and the O'Reilly book.

Extractive

- Pieces are extracted to curate representative information
- Use sentence scoring

Abstractive

- New text generated
- Deep understanding of text
- Transformer models useful here

Use cases



News
Summaries

Use cases



News
Summaries



Content
Recommendations

Use cases



News
Summaries



Content
Recommendations



Language
Translation

Summarization with pipeline

```
from transformers import pipeline

model = "sshleifer/distilbart-cnn-12-6"
summarizer = pipeline(task="summarization", model=model)
```

```
text = "This is my really large text about Data Science..."
summary_text = summarizer(text)
```

```
print(summary_text[0]['summary_text'])
```

"Data science is a field involving multiple interdisciplinary fields."

Parameters for summarization

```
summarizer = pipeline(task="summarization", min_length=10, max_length=50)
```

- Put constraints around minimum and maximum number of words
- Ensure results are meaningful but not verbose
- Small storage capacity, enhance readability, improve quality

Example Error

Your `max_length` is set to 142, but your `input_length` is only 81.
Since this is a summarization task, where outputs shorter than the input are typically wanted, you might consider decreasing `max_length` manually, e.g. `summarizer(..., max_length=40)`

Working with multiple inputs

```
list_of_text = [row["text"] for row in data]
```

- Inputs may not be the same length
- May cause inconsistencies and incorrect results

```
summaries = summarizer(list_of_text, truncation=True)
```

- Use the maximum length size for a token specified by the model

Let's practice!

WORKING WITH HUGGING FACE