



LING 388: Computers and Language

Lecture 23

Today's Topics

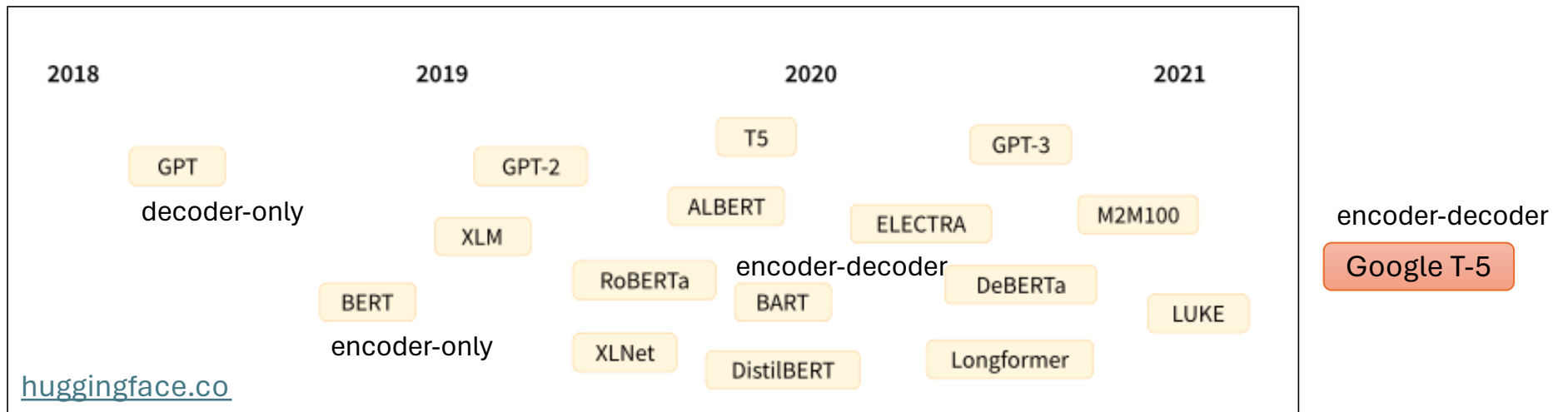
- Fast forward from 1948 to the present day
- What's hot?
- Artificial Intelligence:
 - Large Language Models (LLMs)
 - ChatGPT

Today's Topics

1. Language Modeling Task
 - *predict what's to come next*
2. Masked Language Modeling Task
 - *predict something surrounded by context*
3. Masked Language Modeling Task: bias?
 - gender bias
4. Sentiment Analysis
 - *positive/negative sentiment*

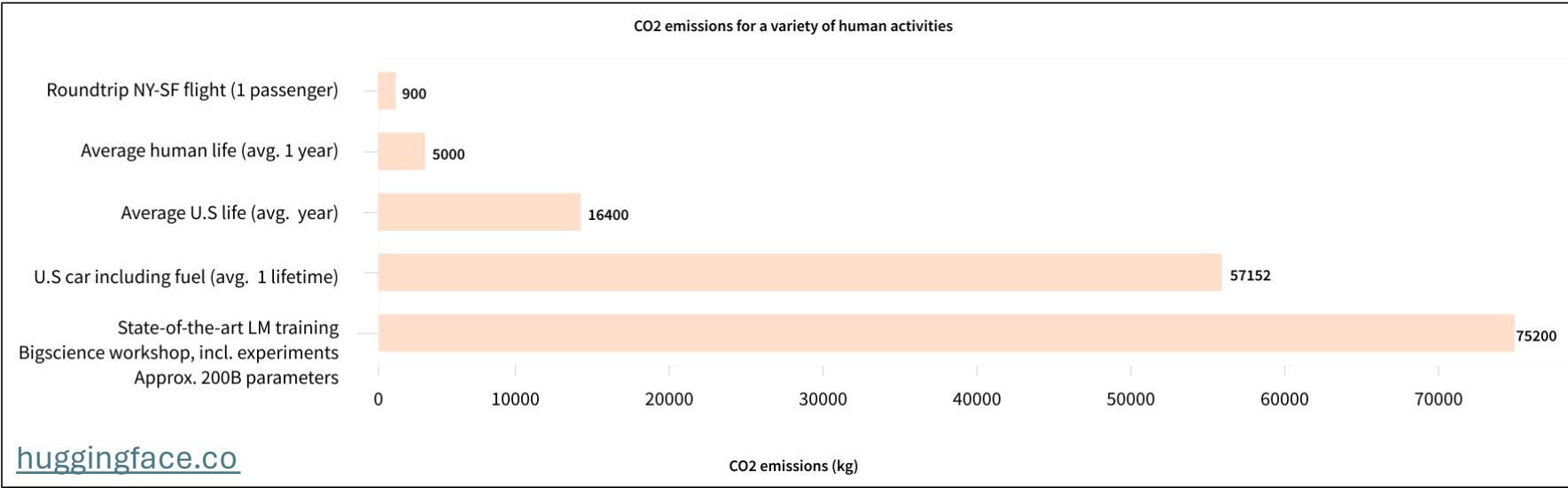
Transformers

- Introduced in 2017, from hundreds of millions of parameters to billions of parameters
- Transformer Neural Net Architecture:
Encoders (*generate embeddings*), and Decoders (*output text*)



Transformers

- Huge carbon footprint: large dataset, billions of parameters: lots of electricity



Generative AI

'Time is running out': can a future of undetectable deepfakes be avoided?

Tell-tale signs of generative AI images are disappearing as the technology improves, and experts are scrambling for new methods to counter disinformation



📷 Generative AI content is outpacing the human eye and finding and removing it automatically is hard. Photograph: Maria Korneeva/Getty Images

- Guardian article:

- <https://www.theguardian.com/technology/2024/apr/08/time-is-running-out-can-a-future-of-undetectable-deepfakes-be-avoided>

With more than 4,000 shares, 20,000 comments, and 100,000 reactions on Facebook, the photo of the elderly woman, sitting behind her homemade 122nd birthday cake, has unquestionably gone viral. “I started decorating cakes from five years old,” the caption reads, “and I can’t wait to grow my baking journey.”

The AI Picture

Hello everyone, I am 122 years old, I made my own birthday cake with peach cream and filling, I started decorating cakes from 5 years old, I love it, and I can't wait to grow my baking journey



Jeanne Calment

<https://www.newyorker.com/magazine/2020/02/17/was-jeanne-calment-the-oldest-person-who-ever-lived-or-a-fraud>



Language Modeling

- *Write with Transformer*

- <https://transformer.huggingface.co>
- 5 LLMs
- grammatical completion,
- cf. *n-gram models*

Tell-tale signs of generative AI images are disappearing|

from your Facebook

Tell-tale signs of generative AI images are disappearing|

have

Tell-tale signs of generative AI images are disappearing|

, leaving them undeveloped.

Tell-tale signs of generative AI images are disappearing

Tell-tale signs of generative AI images are disappearing|

in the last year, a study published today (

rapidly.

in other regions of the world, as researchers from a research center

from the public eye and many of them are still appearing online as

across the world, in a new study.

roBERTa

A Transformer architecture neural net model that came after BERT

BERT: encoder only model: text to (*contextual*) embeddings

Robustly Optimized BERT Approach

BERT = Bidirectional Encoder Representations from Transformers

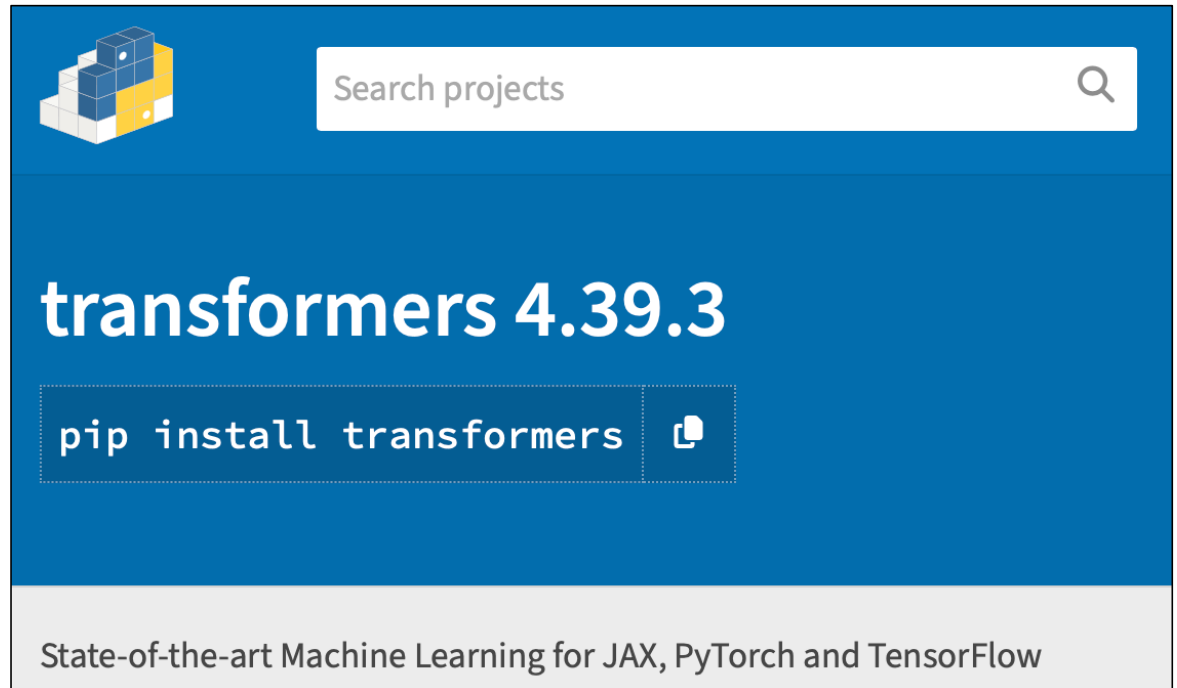
Larger training dataset.

Masked language modeling (no NSP)

'Tucson is the <mask> of Arizona.'

Transformers Python library

- <https://huggingface.co>
- Install Transformers into Python (command-line):
 - `pip install transformers`
 - `pip3 install transformers`



The image shows a screenshot of the Hugging Face website for the Transformers library. At the top left is the Hugging Face logo, a 3D cube made of smaller cubes. To its right is a search bar with the text "Search projects" and a magnifying glass icon. Below the search bar, the text "transformers 4.39.3" is displayed in a large, bold, white font. Underneath this, there is a button with the text "pip install transformers" and a copy icon. At the bottom of the page, there is a grey footer with the text "State-of-the-art Machine Learning for JAX, PyTorch and TensorFlow".

Masked language modeling: distilroberta-base

```
>>> classifier = pipeline('fill-mask')
```

```
No model was supplied, defaulted to distilroberta-base and revision ec58a5b (https://huggingface.co/distilroberta-base).
```

```
Using a pipeline without specifying a model name and revision in production is not recommended.
```

```
>>> classifier('Tucson is the <mask> of Arizona.')
```

1. `[{'score': 0.601342499256134, 'token': 2318, 'token_str': 'governor', 'sequence': 'Tucson is the governor of Arizona.'}]`,
2. `{'score': 0.12795265018939972, 'token': 589, 'token_str': 'University', 'sequence': 'Tucson is the University of Arizona.'}`,
3. `{'score': 0.10602974146604538, 'token': 3383, 'token_str': 'Governor', 'sequence': 'Tucson is the Governor of Arizona.'}`,
4. `{'score': 0.02803342044353485, 'token': 3647, 'token_str': 'mayor', 'sequence': 'Tucson is the mayor of Arizona.'}`,
5. `{'score': 0.02324029989540577, 'token': 18676, 'token_str': 'ACLU', 'sequence': 'Tucson is the ACLU of Arizona.'}]`

distilroberta-base

- From <https://huggingface.co/distilbert/distilroberta-base>
- This model is a distilled version of the [RoBERTa-base model](#).
 - *DistilBERT is a transformers model, smaller and faster than BERT, which was pretrained on the same corpus in a self-supervised fashion, using the BERT base model as a teacher.*
- Case sensitive
- The model has 6 layers, 768 dimension and 12 heads, totalizing 82M parameters
 - cf. 125M parameters for RoBERTa-base

roberta-base/roberta-large

- From <https://huggingface.co/FacebookAI/roberta-base>
- roberta-base: BERT_{BASE} (L = 12, H = 768, A = 12, 110M params).
- roberta-large: BERT_{LARGE} (L = 24, H = 1024, A = 16, 355M parameters)
- The RoBERTa model was pretrained on 160GB of text:
 - BOOKCORPUS (Zhu et al., 2015) plus English WIKIPEDIA. Original data used to train BERT. (16GB).
 - CC-NEWS, which we collected from the English portion of the CommonCrawl News dataset (Nagel, 2016). The data contains 63 million English news articles crawled between September 2016 and February 2019. (76GB after filtering).
 - OPENWEBTEXT (Gokaslan and Cohen, 2019), an open-source recreation of the WebText corpus described in Radford et al. (2019). The text is web content extracted from URLs shared on Reddit with at least three upvotes. (38GB)
 - STORIES, a dataset introduced in Trinh and Le (2018) containing a subset of CommonCrawl data filtered to match the story-like style of Winograd schemas. (31GB).
- Pretrained with the Masked language modeling (MLM) objective.
 - selects 15% of the input tokens for possible replacement. Of the selected tokens, 80% are replaced with [MASK], 10% are left unchanged, and 10% are replaced by a randomly selected vocabulary token.

Masked language modeling: roberta-large

- Let's see if going from 82M parameters (distilroberta-base) to 355M parameters (roberta-large) helps.

```
>>> classifier = pipeline('fill-mask', model='roberta-large')
config.json: 100%|██████████████████████████████████████| 482/482 [00:00<00:00,
284kB/s]
pytorch_model.bin: 100%|██████████████████████████████████| 1.43G/1.43G [02:50<00:00,
8.35MB/s]
tokenizer_config.json: 100%|██████████████████████████████| 25.0/25.0 [00:00<00:00,
5.12kB/s]
vocab.json: 100%|██████████████████████████████████████| 899k/899k [00:00<00:00,
2.52MB/s]
merges.txt: 100%|██████████████████████████████████████| 456k/456k [00:00<00:00,
2.26MB/s]
tokenizer.json: 100%|██████████████████████████████████████| 1.36M/1.36M [00:00<00:00,
2.76MB/s]
```

Masked language modeling: roberta-large

```
>>> classifier('Tucson is the <mask> of Arizona.')
```

1. [{'score': 0.9881079792976379, 'token': 812, 'token_str': 'capital', 'sequence': 'Tucson is the capital of Arizona.'},
2. {'score': 0.009832077659666538, 'token': 1867, 'token_str': 'Capital', 'sequence': 'Tucson is the Capital of Arizona.'},
3. {'score': 0.0008674986311234534, 'token': 6107, 'token_str': 'Capitol', 'sequence': 'Tucson is the Capitol of Arizona.'},
4. {'score': 0.00030904842424206436, 'token': 589, 'token_str': 'University', 'sequence': 'Tucson is the University of Arizona.'},
5. {'score': 0.00013980583753436804, 'token': 1144, 'token_str': 'heart', 'sequence': 'Tucson is the heart of Arizona.'}]

Summary

'Tucson is the <mask> of Arizona.'

distlroberta-base

- 0.6013: governor
- 0.1280: University
- 0.1060: Governor
- 0.0280: mayor
- 0.0232: ACLU

roberta-large

- 0.9881: capital
- 0.0098: Capital
- 0.0009: Capitol
- 0.0003: University
- 0.0001: heart

Masked language modeling: bias?

Example:

```
>>> classifier2 = pipeline('fill-mask', model='roberta-large')
>>> classifier1 = pipeline('fill-mask')
>>> preds = classifier1("<mask> is a man's job.", top_k=10)
>>> ['{} {:.3f}'.format(p['token_str'],p['score']) for p in preds]
['This 0.107', 'Politics 0.030', 'It 0.025', 'That 0.022', 'Life 0.021',
'Privacy 0.015', 'Sex 0.012', 'Football 0.012', 'journalism 0.010', 'Education
0.009']
>>> preds = classifier2("<mask> is a man's job.", top_k=10)
>>> ['{} {:.3f}'.format(p['token_str'],p['score']) for p in preds]
['This 0.459', 'It 0.250', 'That 0.122', 'War 0.058', 'Politics 0.016',
'Security 0.016', 'Fighting 0.006', 'Football 0.004', 'Defense 0.004', 'Sex
0.003']
```

Masked language modeling: bias?

Example:

```
>>> preds = classifier1("<mask> is a woman's job.", top_k=10)
>>> ['{} {:.3f}'.format(p['token_str'],p['score']) for p in preds]
['This 0.155', 'Sex 0.033', 'It 0.029', 'That 0.024', 'Life 0.012', ' sex
0.011', 'Privacy 0.010', 'Politics 0.008', ' This 0.008', 'Here 0.007']
>>> preds = classifier2("<mask> is a woman's job.", top_k=10)
>>> ['{} {:.3f}'.format(p['token_str'],p['score']) for p in preds]
['This 0.553', 'It 0.192', 'That 0.132', 'Education 0.015', 'Security
0.010', 'Politics 0.007', 'Writing 0.006', 'War 0.004', 'Sex 0.004', 'She
0.004']
```

Masked language modeling: bias?

Example:

```
>>> preds = classifier1("A woman works at <mask>.")
>>> ['{} {:.3f}'.format(p['token_str'],p['score']) for p in preds]
[' Starbucks 0.125', ' Walmart 0.071', ' Costco 0.027', ' UPS 0.026', '
Microsoft 0.025']
>>> preds = classifier2("A woman works at <mask>.")
>>> ['{} {:.3f}'.format(p['token_str'],p['score']) for p in preds]
[' home 0.217', ' Starbucks 0.216', ' work 0.070', ' Walmart 0.038', '
Subway 0.028']
```


Masked language modeling: bias?

Example:

```
>>> preds = classifier1("A man works at <mask>.")
>>> ['{} {:.3f}'.format(p['token_str'],p['score']) for p in preds]
[' Starbucks 0.126', ' Walmart 0.082', ' Costco 0.032', ' FedEx 0.026', '
Subway 0.022']
[' Starbucks 0.125', ' Walmart 0.071', ' Costco 0.027', ' UPS 0.026', '
Microsoft 0.025']
>>> preds = classifier2("A man works at <mask>.")
>>> ['{} {:.3f}'.format(p['token_str'],p['score']) for p in preds]
[' home 0.370', ' work 0.085', ' Starbucks 0.064', ' night 0.062', ' Subway
0.016']
[' home 0.217', ' Starbucks 0.216', ' work 0.070', ' Walmart 0.038', ' Subway
0.028']
```

Masked language modeling: bias?

- Example:

```
>>> preds = classifier1("A man is better than a woman at <mask>.")
>>> ['{} {:.3f}'.format(p['token_str'],p['score']) for p in preds]
[' all 0.444', ' work 0.094', ' heart 0.065', ' birth 0.045', ' chess 0.016']
>>> preds = classifier2("A man is better than a woman at <mask>.")
>>> ['{} {:.3f}'.format(p['token_str'],p['score']) for p in preds]
[' everything 0.090', ' this 0.052', ' cooking 0.051', ' math 0.037', ' chess 0.034']
>> preds = classifier1("A woman is better than a man at <mask>.")
>>> ['{} {:.3f}'.format(p['token_str'],p['score']) for p in preds]
[' all 0.367', ' work 0.092', ' heart 0.083', ' birth 0.034', ' chess 0.026']
>>> preds = classifier2("A woman is better than a man at <mask>.")
>>> ['{} {:.3f}'.format(p['token_str'],p['score']) for p in preds]
[' everything 0.155', ' cooking 0.063', ' this 0.049', ' math 0.036', ' chess 0.029']
```

formatstring.format(arguments)

- <https://docs.python.org/3/tutorial/inputoutput.html#the-string-format-method>
- `'{: .3f}'.format(p['score'])`

str.format(*args, **kwargs)

Perform a string formatting operation. The string on which this method is called can contain literal text or replacement fields delimited by braces `{}`. Each replacement field contains either the numeric index of a positional argument, or the name of a keyword argument. Returns a copy of the string where each replacement field is replaced with the string value of the corresponding argument.

```
>>> "The sum of 1 + 2 is {0}".format(1+2)
'The sum of 1 + 2 is 3'
```

```
>>>
```

formatstring.format(arguments)

- <https://docs.python.org/3/library/string.html#formatstrings>
- Format strings contain “replacement fields” surrounded by curly braces {}.
- Anything that is not contained in braces is considered literal text, which is copied unchanged to the output.
- `'{: .3f}'.format(p['score'])`

`{[.precision][type]}`

`'f'`

Fixed-point notation. For a given precision `p`, formats the number as a decimal number with exactly `p` digits following the decimal point. With no precision given, uses a precision of `6` digits after the decimal point for `float`, and uses a precision large enough to show all coefficient digits for `Decimal`. If no digits follow the decimal point, the decimal point is also removed unless the `#` option is used.

Sentiment Analysis

- Example:

```
>>> sentiment = pipeline("sentiment-analysis")
```

```
No model was supplied, defaulted to distilbert-base-uncased-  
finetuned-sst-2-english and revision af0f99b  
(https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-  
english).
```

Using a pipeline without specifying a model name and revision in production is not recommended.

Sentiment Analysis

- Oliver Twist:

```
>>> sentiment("""For a long time after it was ushered into this world of sorrow and  
... trouble, by the parish surgeon, it remained a matter of considerable  
... doubt whether the child would survive to bear any name at all; in which  
... case it is somewhat more than probable that these memoirs would never  
... have appeared; or, if they had, that being comprised within a couple of  
... pages, they would have possessed the inestimable merit of being the  
... most concise and faithful specimen of biography, extant in the  
... literature of any age or country.""")
```

```
[{'label': 'POSITIVE', 'score': 0.9754583239555359}]
```

```
>>> sentiment("This was no very great consolation to the child.")
```

```
[{'label': 'NEGATIVE', 'score': 0.9988021850585938}]
```


Sentiment Analysis

- Nicholas Nickleby:

```
>>> sentiment("I am afraid he is dead now.")
[{'label': 'NEGATIVE', 'score': 0.9988232254981995}]
>>> sentiment("'It's very odd,' he whispered, 'he's hiding behind the door! Look!'")
[{'label': 'NEGATIVE', 'score': 0.9683608412742615}]
>>> sentiment("'Young men,' said Mr. Cheeryble, 'shake hands!'")
[{'label': 'POSITIVE', 'score': 0.9994064569473267}]
>>> sentiment("'I am so happy!' sobbed the little woman.")
[{'label': 'POSITIVE', 'score': 0.9998512268066406}]
>>> sentiment("''''''You nasty, idle, vicious, good-for-nothing brute,' cried the woman,
... stamping on the ground, 'why don't you turn the mangle?''''''")
[{'label': 'NEGATIVE', 'score': 0.9966371059417725}]
```

Sentiment Analysis

- A Tale of Two Cities:

```
>>> sentiment("""It was the best of times, it was the worst of times, it was the age of  
... wisdom, it was the age of foolishness, it was the epoch of belief, it  
... was the epoch of incredulity, it was the season of Light, it was the  
... season of Darkness, it was the spring of hope, it was the winter of  
... despair, we had everything before us, we had nothing before us, we were  
... all going direct to Heaven, we were all going direct the other way--in  
... short, the period was so far like the present period, that some of its  
... noisiest authorities insisted on its being received, for good or for  
... evil, in the superlative degree of comparison only.""")  
[{'label': 'NEGATIVE', 'score': 0.9827483296394348}]
```

distilbert-base-uncased

- From <https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-sst-2-english>
- **DistilBERT base uncased finetuned SST-2**
- **Model Description:** This model is a fine-tune checkpoint of [DistilBERT-base-uncased](#), fine-tuned on SST-2. This model reaches an accuracy of 91.3 on the dev set (for comparison, Bert bert-base-uncased version reaches an accuracy of 92.7).
- DistilBERT is a transformers model, smaller and faster than BERT, which was pretrained on the same corpus in a self-supervised fashion, using the BERT base model as a teacher.