


 [huggingface](#) / [transformers](#)

 Transformers: State-of-the-art Natural Language Processing for Pytorch and TensorFlow 2.0.

 [huggingface.co/transformers](#)

 Apache-2.0 License

☆ 38.3k stars 🍴 9.4k forks

☆ Star

👁 Watch

<> Code

! Issues 488

🔗 Pull requests 81

▶ Actions

📁 Projects 20

🛡 Secur

🔗 master ▼

Go to file



**sgugger** Add timing inside Trainer (#9196) ...

● 36 minutes ago ⌚ 6,156

[View code](#)

README.md



# Transformers

build passing

license Apache-2.0

website online

release v4.1.1

Contributor Covenant

v2.0 adopted

## State-of-the-art Natural Language Processing for PyTorch and TensorFlow 2.0



Transformers provides thousands of pretrained models to perform tasks on texts such as classification, information extraction, question answering, summarization, translation, text generation, etc in 100+ languages. Its aim is to make cutting-edge NLP easier to use for everyone.



Transformers provides APIs to quickly download and use those pretrained models on a given text, fine-tune them on your own datasets then share them with the community on our [model hub](#). At the same time, each python module defining an architecture can be used as a standalone and modified to enable quick research experiments.



Transformers is backed by the two most popular deep learning libraries, [PyTorch](#) and [TensorFlow](#), with a seamless integration between them, allowing you to train your models with one then load it for inference with the other.

## Online demos

---

You can test most of our models directly on their pages from the [model hub](#). We also offer an [inference API](#) to use those models.

Here are a few examples:

- [Masked word completion with BERT](#)
- [Name Entity Recognition with Electra](#)
- [Text generation with GPT-2](#)
- [Natural Language Inference with RoBERTa](#)
- [Summarization with BART](#)
- [Question answering with DistilBERT](#)
- [Translation with T5](#)

[Write With Transformer](#), built by the Hugging Face team, is the official demo of this repo's text generation capabilities.

## Quick tour

---

To immediately use a model on a given text, we provide the `pipeline` API. Pipelines group together a pretrained model with the preprocessing that was used during that model training. Here is how to quickly use a pipeline to classify positive versus negative texts

```
>>> from transformers import pipeline

# Allocate a pipeline for sentiment-analysis
>>> classifier = pipeline('sentiment-analysis')
>>> classifier('We are very happy to include pipeline into the transformers r
[{'label': 'POSITIVE', 'score': 0.9978193640708923}]
```

The second line of code downloads and caches the pretrained model used by the pipeline, the third line evaluates it on the given text. Here the answer is "positive" with a confidence of 99.8%.

This is another example of pipeline used for that can extract question answers from some context:

```
>>> from transformers import pipeline

# Allocate a pipeline for question-answering
>>> question_answerer = pipeline('question-answering')
>>> question_answerer({
...     'question': 'What is the name of the repository ?',
...     'context': 'Pipeline have been included in the huggingface/transformers'
... })
{'score': 0.5135612454720828, 'start': 35, 'end': 59, 'answer': 'huggingface/'}
```

On top of the answer, the pretrained model used here returned its confidence score, along with the start position and its end position in the tokenized sentence. You can learn more about the tasks supported by the `pipeline` API in [this tutorial](#).

To download and use any of the pretrained models on your given task, you just need to use those three lines of codes (PyTorch version):

```
>>> from transformers import AutoTokenizer, AutoModel

>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
>>> model = AutoModel.from_pretrained("bert-base-uncased")

>>> inputs = tokenizer("Hello world!", return_tensors="pt")
>>> outputs = model(**inputs)
```

or for TensorFlow:

```
>>> from transformers import AutoTokenizer, TFAutoModel

>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
>>> model = TFAutoModel.from_pretrained("bert-base-uncased")

>>> inputs = tokenizer("Hello world!", return_tensors="tf")
>>> outputs = model(**inputs)
```

The tokenizer is responsible for all the preprocessing the pretrained model expects, and can be called directly on one (or list) of texts (as we can see on the fourth line of both code examples). It will output a dictionary you can directly pass to your model (which is done on the fifth line).

The model itself is a regular `Pytorch nn.Module` or a `TensorFlow tf.keras.Model` (depending on your backend) which you can use normally. For instance, [this tutorial](#) explains how to integrate such a model in classic PyTorch or TensorFlow training loop, or how to use our `Trainer` API to quickly fine-tune the on a new dataset.

## Why should I use transformers?

---

### 1. Easy-to-use state-of-the-art models:

- High performance on NLU and NLG tasks.
- Low barrier to entry for educators and practitioners.
- Few user-facing abstractions with just three classes to learn.
- A unified API for using all our pretrained models.

### 2. Lower compute costs, smaller carbon footprint:

- Researchers can share trained models instead of always retraining.
- Practitioners can reduce compute time and production costs.
- Dozens of architectures with over 2,000 pretrained models, some in more than 100 languages.

### 3. Choose the right framework for every part of a model's lifetime:

- Train state-of-the-art models in 3 lines of code.
- Move a single model between TF2.0/PyTorch frameworks at will.
- Seamlessly pick the right framework for training, evaluation, production.

### 4. Easily customize a model or an example to your needs:

- Examples for each architecture to reproduce the results by the official authors of said architecture.
- Expose the models internal as consistently as possible.
- Model files can be used independently of the library for quick experiments.

## Why shouldn't I use transformers?

---

- This library is not a modular toolbox of building blocks for neural nets. The code in the model files is not refactored with additional abstractions on purpose, so that researchers can quickly iterate on each of the models without diving in additional abstractions/files.

- The training API is not intended to work on any model but is optimized to work with the models provided by the library. For generic machine learning loops, you should use another library.
- While we strive to present as many use cases as possible, the scripts in our [examples folder](#) are just that: examples. It is expected that they won't work out-of-the box on your specific problem and that you will be required to change a few lines of code to adapt them to your needs.

## Installation

---

### With pip

This repository is tested on Python 3.6+, PyTorch 1.0.0+ (PyTorch 1.3.1+ for [examples](#)) and TensorFlow 2.0.

You should install 🤗 Transformers in a [virtual environment](#). If you're unfamiliar with Python virtual environments, check out the [user guide](#).

First, create a virtual environment with the version of Python you're going to use and activate it.

Then, you will need to install at least one of TensorFlow 2.0, PyTorch or Flax. Please refer to [TensorFlow installation page](#), [PyTorch installation page](#) regarding the specific install command for your platform and/or [Flax installation page](#).

When TensorFlow 2.0 and/or PyTorch has been installed, 🤗 Transformers can be installed using pip as follows:

```
pip install transformers
```

If you'd like to play with the examples, you must [install the library from source](#).

### With conda

Since Transformers version v4.0.0, we now have a conda channel: `huggingface`.

🤗 Transformers can be installed using conda as follows:

```
conda install -c huggingface transformers
```

Follow the installation pages of TensorFlow, PyTorch or Flax to see how to install them with conda.

## Models architectures

**All the model checkpoints** provided by 🤗 Transformers are seamlessly integrated from the huggingface.co [model hub](#) where they are uploaded directly by [users](#) and [organizations](#).

Current number of checkpoints: models 4,709

🤗 Transformers currently provides the following architectures (see [here](#) for a high-level summary of each them):

1. **ALBERT** (from Google Research and the Toyota Technological Institute at Chicago) released with the paper [ALBERT: A Lite BERT for Self-supervised Learning of Language Representations](#), by Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, Radu Soricut.
2. **BART** (from Facebook) released with the paper [BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension](#) by Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov and Luke Zettlemoyer.
3. **BARThez** (from École polytechnique) released with the paper [BARThez: a Skilled Pretrained French Sequence-to-Sequence Model](#) by Moussa Kamal Eddine, Antoine J.-P. Tixier, Michalis Vazirgiannis.
4. **BERT** (from Google) released with the paper [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#) by Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova.
5. **BERT For Sequence Generation** (from Google) released with the paper [Leveraging Pre-trained Checkpoints for Sequence Generation Tasks](#) by Sascha Rothe, Shashi Narayan, Aliaksei Severyn.
6. **Blenderbot** (from Facebook) released with the paper [Recipes for building an open-domain chatbot](#) by Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M. Smith, Y-Lan Boureau, Jason Weston.
7. **CamemBERT** (from Inria/Facebook/Sorbonne) released with the paper [CamemBERT: a Tasty French Language Model](#) by Louis Martin\*, Benjamin Muller\*, Pedro Javier Ortiz Suárez\*, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah and Benoît Sagot.

8. **CTRL** (from Salesforce) released with the paper [CTRL: A Conditional Transformer Language Model for Controllable Generation](#) by Nitish Shirish Keskar\*, Bryan McCann\*, Lav R. Varshney, Caiming Xiong and Richard Socher.
9. **DeBERTa** (from Microsoft Research) released with the paper [DeBERTa: Decoding-enhanced BERT with Disentangled Attention](#) by Pengcheng He, Xiaodong Liu, Jianfeng Gao, Weizhu Chen.
10. **DialoGPT** (from Microsoft Research) released with the paper [DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation](#) by Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, Bill Dolan.
11. **DistilBERT** (from HuggingFace), released together with the paper [DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter](#) by Victor Sanh, Lysandre Debut and Thomas Wolf. The same method has been applied to compress GPT2 into [DistilGPT2](#), RoBERTa into [DistilRoBERTa](#), Multilingual BERT into [DistilmBERT](#) and a German version of DistilBERT.
12. **DPR** (from Facebook) released with the paper [Dense Passage Retrieval for Open-Domain Question Answering](#) by Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih.
13. **ELECTRA** (from Google Research/Stanford University) released with the paper [ELECTRA: Pre-training text encoders as discriminators rather than generators](#) by Kevin Clark, Minh-Thang Luong, Quoc V. Le, Christopher D. Manning.
14. **FlauBERT** (from CNRS) released with the paper [FlauBERT: Unsupervised Language Model Pre-training for French](#) by Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, Didier Schwab.
15. **Funnel Transformer** (from CMU/Google Brain) released with the paper [Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing](#) by Zihang Dai, Guokun Lai, Yiming Yang, Quoc V. Le.
16. **GPT** (from OpenAI) released with the paper [Improving Language Understanding by Generative Pre-Training](#) by Alec Radford, Karthik Narasimhan, Tim Salimans and Ilya Sutskever.
17. **GPT-2** (from OpenAI) released with the paper [Language Models are Unsupervised Multitask Learners](#) by Alec Radford\*, Jeffrey Wu\*, Rewon Child, David Luan, Dario Amodei\*\* and Ilya Sutskever\*\*.
18. **LayoutLM** (from Microsoft Research Asia) released with the paper [LayoutLM: Pre-training of Text and Layout for Document Image Understanding](#) by Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou.
19. **Longformer** (from AllenAI) released with the paper [Longformer: The Long-Document Transformer](#) by Iz Beltagy, Matthew E. Peters, Arman Cohan.



20. **LXMERT** (from UNC Chapel Hill) released with the paper [LXMERT: Learning Cross-Modality Encoder Representations from Transformers for Open-Domain Question Answering](#) by Hao Tan and Mohit Bansal.
21. **MarianMT** Machine translation models trained using [OPUS](#) data by Jörg Tiedemann. The [Marian Framework](#) is being developed by the Microsoft Translator Team.
22. **MBart** (from Facebook) released with the paper [Multilingual Denoising Pre-training for Neural Machine Translation](#) by Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, Luke Zettlemoyer.
23. **MPNet** (from Microsoft Research) released with the paper [MPNet: Masked and Permuted Pre-training for Language Understanding](#) by Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, Tie-Yan Liu.
24. **MT5** (from Google AI) released with the paper [mT5: A massively multilingual pre-trained text-to-text transformer](#) by Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, Colin Raffel.
25. **Pegasus** (from Google) released with the paper [PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization](#) by Jingqing Zhang, Yao Zhao, Mohammad Saleh and Peter J. Liu.
26. **ProphetNet** (from Microsoft Research) released with the paper [ProphetNet: Predicting Future N-gram for Sequence-to-Sequence Pre-training](#) by Yu Yan, Weizhen Qi, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang and Ming Zhou.
27. **Reformer** (from Google Research) released with the paper [Reformer: The Efficient Transformer](#) by Nikita Kitaev, Łukasz Kaiser, Anselm Levskaya.
28. **RoBERTa** (from Facebook), released together with the paper [Robustly Optimized BERT Pretraining Approach](#) by Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov. ultilingual BERT into [DistilmBERT](#) and a German version of DistilBERT.
29. **SqueezeBert** released with the paper [SqueezeBERT: What can computer vision teach NLP about efficient neural networks?](#) by Forrest N. Iandola, Albert E. Shaw, Ravi Krishna, and Kurt W. Keutzer.
30. **T5** (from Google AI) released with the paper [Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer](#) by Colin Raffel and Noam Shazeer and Adam Roberts and Katherine Lee and Sharan Narang and Michael Matena and Yanqi Zhou and Wei Li and Peter J. Liu.
31. **TAPAS** (from Google AI) released with the paper [TAPAS: Weakly Supervised Table Parsing via Pre-training](#) by Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno and Julian Martin Eisenschlos.




32. **Transformer-XL** (from Google/CMU) released with the paper [Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context](#) by Zihang Dai\*, Zhilin Yang\*, Yiming Yang, Jaime Carbonell, Quoc V. Le, Ruslan Salakhutdinov.
33. **XLNet** (from Facebook) released together with the paper [Cross-lingual Language Model Pretraining](#) by Guillaume Lample and Alexis Conneau.
34. **XLNet-ProphetNet** (from Microsoft Research) released with the paper [ProphetNet: Predicting Future N-gram for Sequence-to-Sequence Pre-training](#) by Yu Yan, Weizhen Qi, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang and Ming Zhou.
35. **XLNet-RoBERTa** (from Facebook AI), released together with the paper [Unsupervised Cross-lingual Representation Learning at Scale](#) by Alexis Conneau\*, Kartikay Khandelwal\*, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer and Veselin Stoyanov.
36. **XLNet** (from Google/CMU) released with the paper [XLNet: Generalized Autoregressive Pretraining for Language Understanding](#) by Zhilin Yang\*, Zihang Dai\*, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le.
37. Want to contribute a new model? We have added a **detailed guide and templates** to guide you in the process of adding a new model. You can find them in the [templates](#) folder of the repository. Be sure to check the [contributing guidelines](#) and contact the maintainers or open an issue to collect feedbacks before starting your PR.

To check if each model has an implementation in PyTorch/TensorFlow/Flax or has an associated tokenizer backed by the  Tokenizers library, refer to [this table](#)

These implementations have been tested on several datasets (see the example scripts) and should match the performances of the original implementations. You can find more details on the performances in the Examples section of the [documentation](#).

## Learn more

Section	Description
<a href="#">Documentation</a>	Full API documentation and tutorials
<a href="#">Task summary</a>	Tasks supported by  Transformers
<a href="#">Preprocessing tutorial</a>	Using the <code>Tokenizer</code> class to prepare data for the models

Section	Description
<a href="#">Training and fine-tuning</a>	Using the models provided by 🤗 Transformers in a PyTorch/TensorFlow training loop and the <code>Trainer</code> API
<a href="#">Quick tour: Fine-tuning/usage scripts</a>	Example scripts for fine-tuning models on a wide range of tasks
<a href="#">Model sharing and uploading</a>	Upload and share your fine-tuned models with the community
<a href="#">Migration</a>	Migrate to 🤗 Transformers from <code>pytorch-transformers</code> or <code>pytorch-pretrained-bert</code>

## Citation

We now have a [paper](#) you can cite for the 🤗 Transformers library:

```
@inproceedings{wolf-etal-2020-transformers,
  title = "Transformers: State-of-the-Art Natural Language Processing",
  author = "Thomas Wolf and Lysandre Debut and Victor Sanh and Julien Chaumond",
  booktitle = "Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing",
  month = oct,
  year = "2020",
  address = "Online",
  publisher = "Association for Computational Linguistics",
  url = "https://www.aclweb.org/anthology/2020.emnlp-demos.6",
  pages = "38--45"
}
```

## Releases 49

📦 **v4.1.1: TAPAS, MPNet, model parallelization, Sharded DDP, conda, multi-part downloads.** Latest  
yesterday

[+ 48 releases](#)

## Packages

No packages published

Used by 5.9k



Contributors 704



+ 693 contributors

Languages

