Transformers: State-of-the-Art Natural Language Processing

By

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, ´ Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, Alexander M. Rush

Hugging Face, Brooklyn, USA

What this paper is about

Hugging Face **Transformers** library

https://github.com/huggingface/transformers
https://huggingface.co/docs/transformers

Outline

Introduction

Introduction

- Transformers [1] has become the dominant architecture for NLP, surpassing CNN and RNN
  - Scales with training data and model size
  - Facilitate parallel training
  - Captures long-range sequence features
- The Transformer architecture is particularly conducive to pretraining on large text corpora, leading to major gains in accuracy on downstream tasks
  - Text classification
  - Language understanding
  - Machine translation
  - Coreference resolution
  - Commonsense inference
  - Summarization

[1] Vaswani, Ashish, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones et al. "Tensor2tensor for neural machine translation." *arXiv preprint arXiv:1803.07416* (2018).

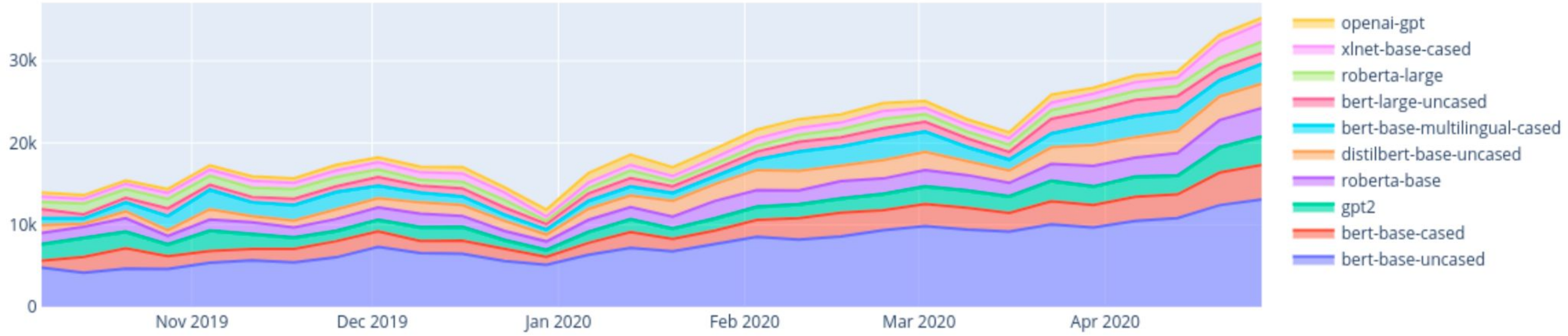
Introduction > Transformers Library

- Transformers is a library dedicated to support Transformer-based architectures and facilitate the distribution of pretrained models.
- Maintained by engineers and researchers at Hugging Face with over 400 external contributors
- Designed for both research and production
  - Easy to read, extend and deploy
  - Wide variety of pretrained models in a centralized model hub
- Released under Apache 2.0 license and
- Available in Github: https://github.com/huggingface/ transformers
- Detailed documentation and tutorials area available on Hugging Face website: https://huggingface.co/docs/transformers

-

Average daily downloads of pretrained models

2019 Oct - 2020 May



Related Work

Related Work

- Transformers library was inspired by
  - Tensor2tensor [1]
  - BERT [2]
  - Concept of providing easy caching for pretrained models stemmed from AllenNLP [3]

[1] Vaswani, Ashish, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones et al. "Tensor2tensor for neural machine translation." *arXiv preprint arXiv:1803.07416* (2018).
[2] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).
[3] Gardner, Matt, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. "Allennlp: A deep semantic natural language processing platform." *arXiv preprint arXiv:1803.07640* (2018).

Related Work

- Transformers library is closely related to
  - Fairseq [4]
  - OpenNMT [5]
  - Texar [6]
  - Megantron-LM [7]
  - Marian NMT [8]

[4] Ott, Myle, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. "fairseq: A fast, extensible toolkit for sequence modeling." *arXiv preprint arXiv:1904.01038* (2019).
[5] Klein, Guillaume, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. "Opennmt: Open-source toolkit for neural machine translation." *arXiv preprint arXiv:1701.02810* (2017).
[6] Hu, Zhiting, Haoran Shi, Bowen Tan, Wentao Wang, Zichao Yang, Tiancheng Zhao, Junxian He et al. "Texar: A modularized, versatile, and extensible toolkit for text generation." *arXiv preprint arXiv:1809.00794* (2018).
[7] Shoeybi, Mohammad, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. "Megatron-lm: Training multi-billion parameter language models using model parallelism." *arXiv preprint arXiv:1909.08053* (2019).
[8] Junczys-Dowmunt, Marcin, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide et al. "Marian: Fast neural machine translation in C++." *arXiv preprint arXiv:1804.00344* (2018).

Related Work

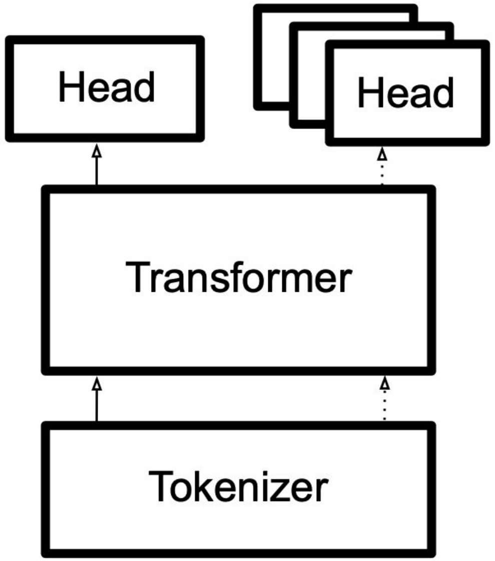
- Easy to use user facing other libraries
    - NLTK [9]
    - Stanford CoreNLP [10]
    - Spacy [11]
    - AlenNLP [3]
    - Flair [12]
    - Stanza [13]

[9] Loper, Edward, and Steven Bird. "Nltk: The natural language toolkit." *arXiv preprint cs/0205028* (2002).
[10] Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. "The Stanford CoreNLP natural language processing toolkit." In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations, pp. 55-60. 2014.
[11] Honnibal, Matthew, and Ines Montani. "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing." To appear 7, no. 1 (2017): 411-420.
[3] Gardner, Matt, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. "Allennlp: A deep semantic natural language processing platform." *arXiv preprint arXiv:1803.07640* (2018).
[12] Akbik, Alan, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. "FLAIR: An easy-to-use framework for state-of-the-art NLP." In Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics (demonstrations), pp. 54-59. 2019.
[13] Qi, Peng, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. "Stanza: A Python natural language processing toolkit for many human languages." arXiv preprint arXiv:2003.07082 (2020).

Library Design

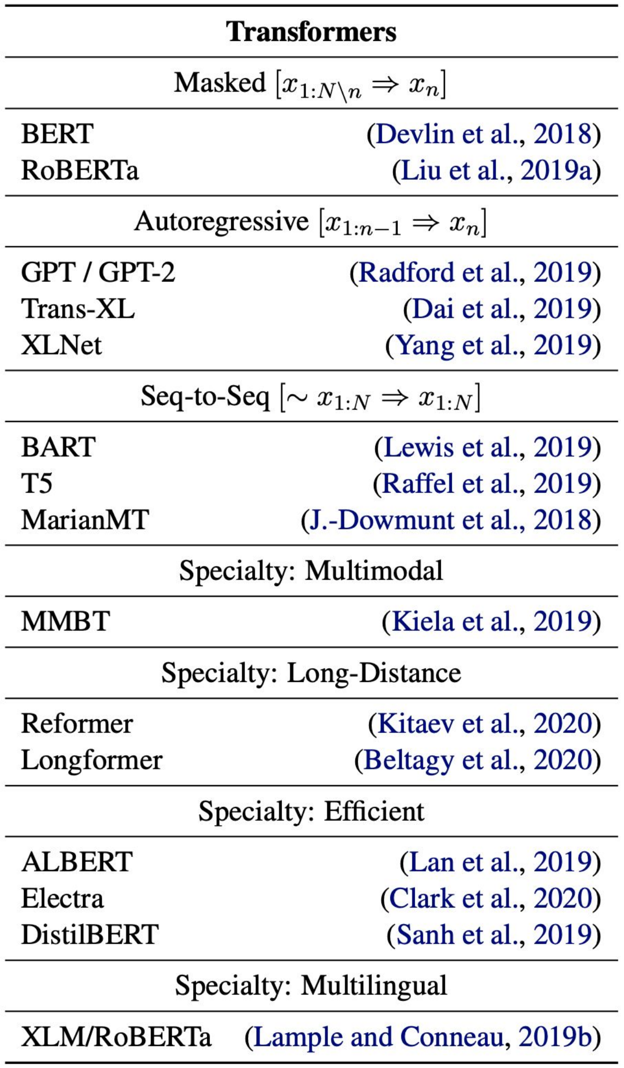
Library Design

- Designed to mirror the standard NLP machine learning model pipeline
  - Process data
  - Apply a model
  - Make predictions
- Every model in the library is defined by 3 building blocks
  - A Tokenizer
    - Converts raw text to sparse index encodings
  - A Transformer
    - Transforms sparse indices to contextual embeddings
  - A Head
    - Uses contextual embeddings to make a task-specific prediction



Library Design > Transformers

- Same multi-headed attention core
- Some significant differences in
  - Positional representation
  - Masking
  - Padding
  - Use of seq2seq design



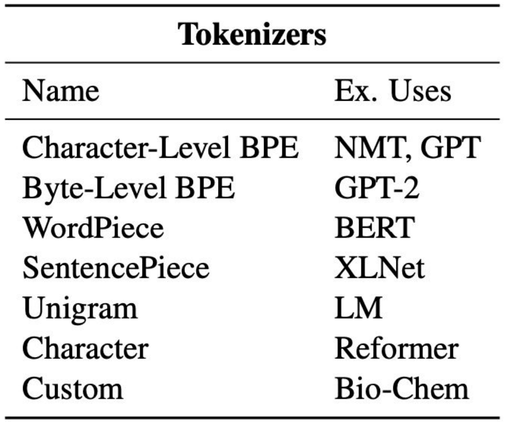
Transformers	
Masked $[x_{1:N\setminus n} \Rightarrow x_n]$	
BERT	(Devlin et al., 2018)
RoBERTa	(Liu et al., 2019a)
Autoregressive $[x_{1:n-1} \Rightarrow x_n]$	
GPT / GPT-2	(Radford et al., 2019)
Trans-XL	(Dai et al., 2019)
XLNet	(Yang et al., 2019)
Seq-to-Seq $[\sim x_{1:N} \Rightarrow x_{1:N}]$	
BART	(Lewis et al., 2019)
T5	(Raffel et al., 2019)
MarianMT	(J.-Dowmunt et al., 2018)
Specialty: Multimodal	
MMBT	(Kiela et al., 2019)
Specialty: Long-Distance	
Reformer	(Kitaev et al., 2020)
Longformer	(Beltagy et al., 2020)
Specialty: Efficient	
ALBERT	(Lan et al., 2019)
Electra	(Clark et al., 2020)
DistilBERT	(Sanh et al., 2019)
Specialty: Multilingual	
XLM/RoBERTa	(Lample and Conneau, 2019b)

Library Design > Transformers

- Models follow a hierarchy of abstraction
- Base class implements the model's computation graph
- Encoding is projected on the embedding matrix
- Self-attention layers are applied in a series
- Final encoder hidden states are generated
- Base class is specific to each model
- Base class closely follows the model's original implementation
- Users can easily dissect the inner workings of each architecture
- Each model is implemented in a single file
- Implementation in a single file enables ease of extensibility

Library Design > Tokenizers

- The library is available at https://github.com/huggingface/tokenizers
- Tokenizers are a critical aspect of the library for NLP tasks
- Tokenizer classes inherit from a common base class
- Tokenizers can be instantiated from pretrained models or configured manually
- Tokenizers store vocabulary token-to-index map for the model
- Tokenizers handle encoding and decoding of input sequences
- Tokenization process is specific to each model
- Tokenizers can be modified by users
- Interfaces allow users to add additional token mappings
- Users can add special tokens like classification or separation tokens
- Vocabulary can be resized by users

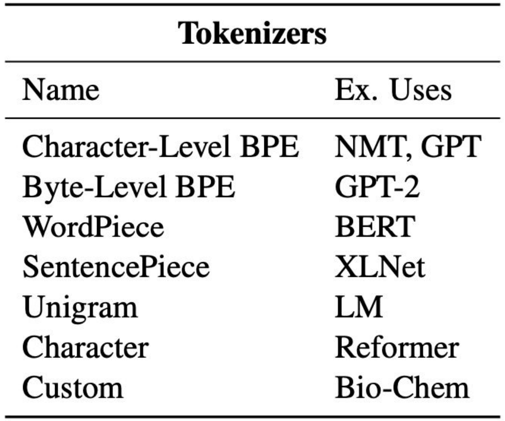


Tokenizers

Name	Ex. Uses
Character-Level BPE	NMT, GPT
Byte-Level BPE	GPT-2
WordPiece	BERT
SentencePiece	XLNet
Unigram	LM
Character	Reformer
Custom	Bio-Chem

Library Design > Tokenizers

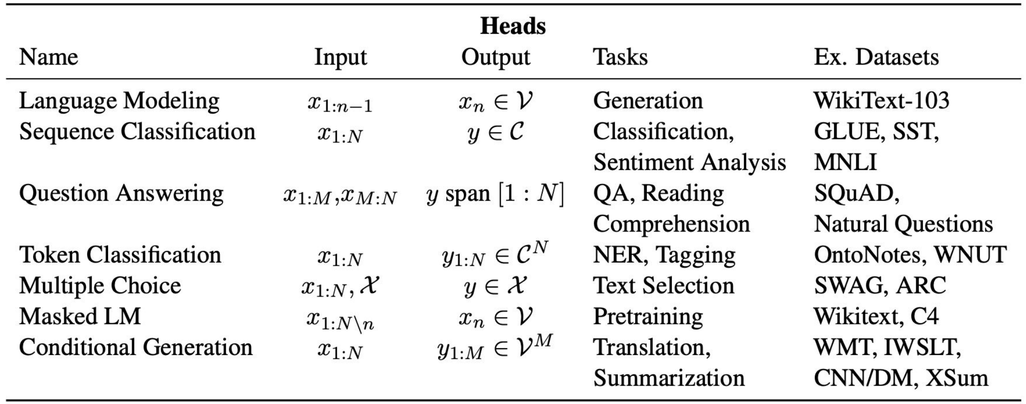
- Token type indices can be implemented for sequence classification
- Maximum length sequence truncating can be performed by tokenizers
- Truncating takes into account model-specific special tokens
  (Most pretrained Transformer models often have a maximum
  sequence length)
- Python-based tokenization can be slow for training on large datasets
- Transformers recently switched to using a highly-optimized tokenization
  library by default
- The library is written in Rust
- Rust implementation speeds up tokenization during training and deployment



Tokenizers

Name	Ex. Uses
Character-Level BPE	NMT, GPT
Byte-Level BPE	GPT-2
WordPiece	BERT
SentencePiece	XLNet
Unigram	LM
Character	Reformer
Custom	Bio-Chem

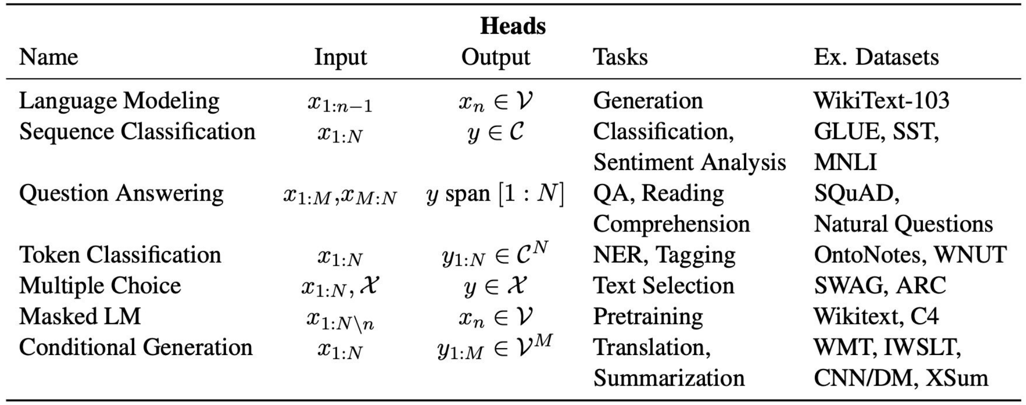
Library Design > Heads



		Heads		
Name	Input	Output	Tasks	Ex. Datasets
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$	Generation	WikiText-103
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$	Classification, Sentiment Analysis	GLUE, SST, MNLI
Question Answering	$x_{1:M}, x_{M:N}$	y span $[1:N]$	QA, Reading Comprehension	SQuAD, Natural Questions
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$	NER, Tagging	OntoNotes, WNUT
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$	Text Selection	SWAG, ARC
Masked LM	$x_{1:N \setminus n}$	$x_n \in \mathcal{V}$	Pretraining	Wikitext, C4
Conditional Generation	$x_{1:N}$	$y_{1:M} \in \mathcal{V}^M$	Translation, Summarization	WMT, IWSLT, CNN/DM, XSum

- Each Transformer model can be paired with ready-implemented heads for different tasks
- These heads are implemented as wrapper classes on top of the base class
- Heads add a specific output layer and optional loss function to the Transformer's contextual embeddings
- Head classes follow a naming pattern: **XXXForSequenceClassification**, where **XXX is the model name**
- Heads can be used for adaptation (fine-tuning) or pretraining
- Some heads, like conditional generation, support additional functionality such as sampling and beam search

Library Design > Heads



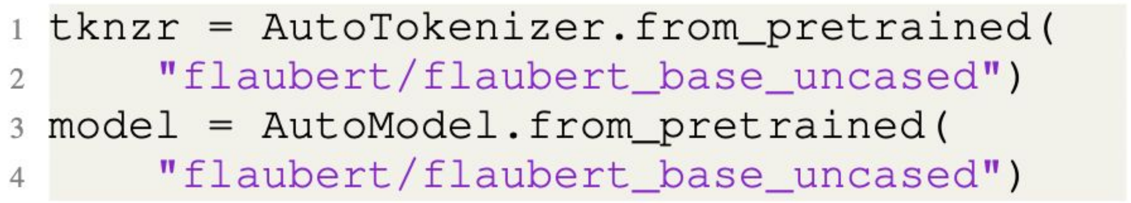
	Heads			
Name	Input	Output	Tasks	Ex. Datasets
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$	Generation	WikiText-103
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$	Classification, Sentiment Analysis	GLUE, SST, MNLI
Question Answering	$x_{1:M}$,$x_{M:N}$	y span $[1:N]$	QA, Reading Comprehension	SQuAD, Natural Questions
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$	NER, Tagging	OntoNotes, WNUT
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$	Text Selection	SWAG, ARC
Masked LM	$x_{1:N\setminus n}$	$x_n \in \mathcal{V}$	Pretraining	Wikitext, C4
Conditional Generation	$x_{1:N}$	$y_{1:M} \in \mathcal{V}^M$	Translation, Summarization	WMT, IWSLT, CNN/DM, XSum

- Pretrained models come with released heads used for their own pretraining objectives
- For example, BERT has language modeling and next sentence prediction heads
- These released heads facilitate easy adaptation of the pretrained models
- Users can also utilize the same core Transformer parameters with different heads for fine-tuning
- The library includes a collection of examples demonstrating each head on real problems
- Examples show how a pretrained model can be adapted with a specific head for achieving state-of-the-art results

Community Model Hub

Community Model Hub

- Facilitate easy use and distribution of pretrained models
- The Model Hub makes it simple for any end-user to access a model for use with their own data
- This hub now contains 2,097 user models, both pretrained and fine-tuned, from across the community
- To upload a model, any user can sign up for an account and use a command-line interface to produce an archive consisting a **tokenizer, transformer,** and **head**
- Just 2 lines of code for load the uploaded models



```
tknzr = AutoTokenizer.from_pretrained(
    "flaubert/flaubert_base_uncased")
model = AutoModel.from_pretrained(
    "flaubert/flaubert_base_uncased")
```

Deployment

Deployment

- Models in the library are available in both PyTorch and TensorFlow frameworks
- There is interoperability between the two frameworks
- Models trained in one framework can be saved through standard serialization
- Saved model files can be seamlessly reloaded in the other framework
- This allows for easy switching between frameworks throughout the model's lifecycle
- Switching can be done during training, serving, and other stages

Deployment

- Each framework in the library has deployment recommendations
- In PyTorch, models are compatible with TorchScript, an intermediate representation
- TorchScript allows models to be run efficiently in Python or in high-performance environments like C++
- Fine-tuned models can be exported to production-friendly environments and run through TorchServing
- TensorFlow has several serving options within its ecosystem
- TensorFlow serving options can be used directly for deployment

Deployment

- Library can export models to intermediate neural network formats
- Models can be converted to the Open Neural Network Exchange format (ONNX) for deployment
- ONNX format allows the model to be run in a standardized interoperable manner
- Conversion to ONNX format can lead to significant speed improvements
- Experiments in collaboration with the ONNX team optimized BERT, RoBERTa, and GPT-2 models
- Using the ONNX intermediate format, nearly a 4x speedup was achieved on the model
- The team is also experimenting with other promising intermediate formats such as JAX/XLA and TVM

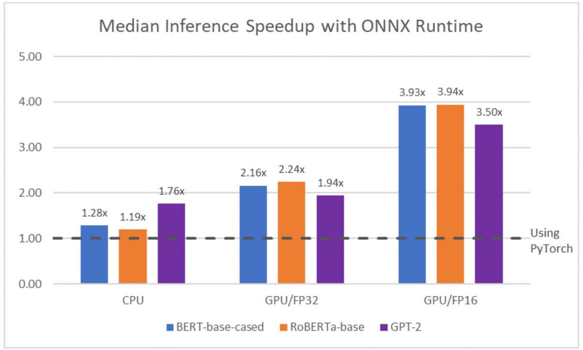


Figure 4: Experiments with *Transformers* inference in collaboration with ONNX.

Deployment

- With the increasing use of Transformers in various NLP applications, deployment to edge devices like phones or home electronics is becoming crucial
- Models can be converted to CoreML weights using adapters for embedding them within iOS applications
- This enables on-the-edge machine learning on iOS devices
    - Code : https://github.com/huggingface/ swift- coreml- transformers
- Similar methods can be used for deploying models on Android devices

Conclusion

Conclusion

- As Transformers and pretraining become more prominent in NLP, accessibility to these models is crucial for researchers and end-users
- Transformers is an open-source library and community that aims to make large-scale pretrained models easily accessible
- The library enables users to build, experiment, and deploy models in downstream tasks with state-of-the-art performance
- Transformers has gained significant organic traction since its release
- It provides core infrastructure and continues to facilitate access to new models

References

[1] Vaswani, Ashish, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones et al. "Tensor2tensor for neural machine translation." *arXiv preprint arXiv:1803.07416* (2018).
[2] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).
[3] Gardner, Matt, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. "Allennlp: A deep semantic natural language processing platform." *arXiv preprint arXiv:1803.07640* (2018).
[4] Ott, Myle, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. "fairseq: A fast, extensible toolkit for sequence modeling." *arXiv preprint arXiv:1904.01038* (2019).
[5] Klein, Guillaume, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. "Opennmt: Open-source toolkit for neural machine translation." *arXiv preprint arXiv:1701.02810* (2017).
[6] Hu, Zhiting, Haoran Shi, Bowen Tan, Wentao Wang, Zichao Yang, Tiancheng Zhao, Junxian He et al. "Texar: A modularized, versatile, and extensible toolkit for text generation." *arXiv preprint arXiv:1809.00794* (2018).
[7] Shoeybi, Mohammad, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. "Megatron-lm: Training multi-billion parameter language models using model parallelism." *arXiv preprint arXiv:1909.08053* (2019).
[8] Junczys-Dowmunt, Marcin, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide et al. "Marian: Fast neural machine translation in C++." *arXiv preprint arXiv:1804.00344* (2018).
[9] Loper, Edward, and Steven Bird. "Nltk: The natural language toolkit." *arXiv preprint cs/0205028* (2002).
[10] Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. "The Stanford CoreNLP natural language processing toolkit." In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pp. 55-60. 2014.
[11] Honnibal, Matthew, and Ines Montani. "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing." To appear 7, no. 1 (2017): 411-420.
[12] Akbik, Alan, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. "FLAIR: An easy-to-use framework for state-of-the-art NLP." In Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics (demonstrations), pp. 54-59. 2019.
[13] Qi, Peng, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. "Stanza: A Python natural language processing toolkit for many human languages." arXiv preprint arXiv:2003.07082 (2020).

Q n A

Thank you…