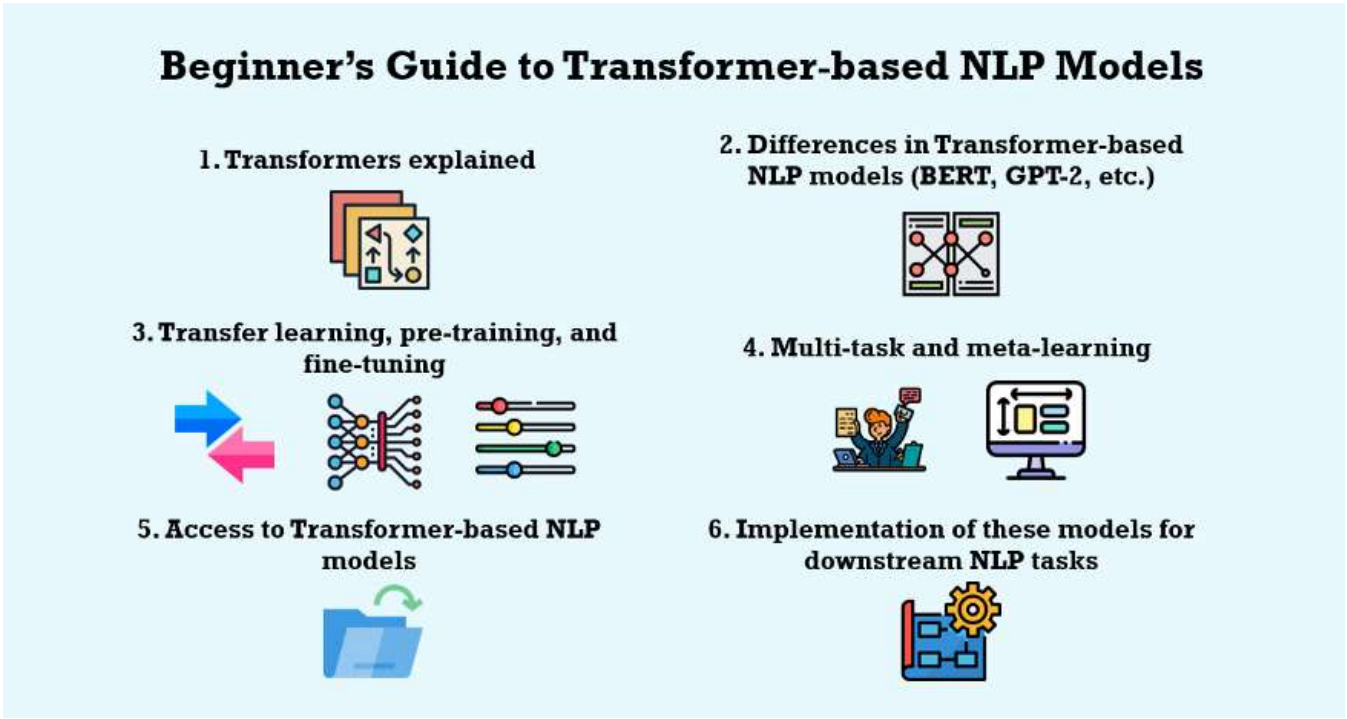
Julia Nikulski · Follow

May 11, 2021 · 11 min read ★



Graphic created by Julia Nikulski displaying the topics covered in the beginner's guide on how to use transformer-based NLP models. Icons made by Becris, Freepik, ultimatearm, monkik, and Eucalyp from Flaticon.

HANDS-ON TUTORIALS

How to Use Transformer-based NLP Models

A beginner's guide to understanding and implementing BERT, GPT-2, and other NLP models for downstream tasks

A few months ago, I started working on a project which involved text classification. I had **previously only worked with basic NLP techniques** to prepare text data and applied simple ML algorithms for classification. However, I was aware of the state-of-

Actually, let me rephrase this part to depict my state of knowledge at the time accurately: I knew there had been a significant buzz around the release of GPT-3 and how well this model could generate text on its own. And someone told me this model could be used for text classification. **How exactly could I "take this model" and make it predict labels for my data? I had no idea.**

There are great blog posts out there that explain the technical intricacies of Transformers, that contrast the many language understanding and language generation models currently available, and that provide tutorials and code on implementing specific NLP tasks. However, **I lacked basic information to understand the bigger picture of how to use these NLP models.**

This article, therefore, provides **a high-level overview that communicates the basics** around **how these models work, where you can access them, and how you can apply them to your own dataset and NLP task**. It is aimed at people familiar with NLP basics but new to more complex language models and transfer learning. To get more details beyond the basics, I provide links and references to academic papers, blog posts, and code walk-throughs.

1. What is a Transformer?

A Transformer is **a type of neural network architecture** developed by Vaswani et al. in 2017 [1]. Without going into too much detail, this model architecture consists of a **multi-head self-attention mechanism combined with an encoder-decoder structure**. It can achieve SOTA results that outperform various other models leveraging recurrent (RNN) or convolutional neural networks (CNN) both in terms of evaluation score (BLEU score) and training time.

A **key advantage of a Transformer** over other neural network (NN) structures is that a **longer-distanced context around a word** is considered in a more computationally efficient way [1, 2]. The phrase "making…more difficult" would be recognized in the sentence "making the registration or voting process more difficult" even though "more difficult" is a rather distant dependency of "making" [1]. The computation of the relevant context around a word can be done in parallel, saving significant training resources.

Initially, the Transformer was developed for NLP tasks and applied by Vaswani et al. to machine translation [1]. However, it has been adapted for image recognition and other areas too. For more details on the exact mechanisms and functioning of the Transformer, take a look at these **in-depth explanations**:

- How Transformers Work by Giuliano Giacaglia
- Illustrated Guide to Transformers- Step by Step Explanation by Michael Phi
- What is a Transformer by Maxime
- The Illustrated Transformer by Jay Alammar

Photo by Markus Spiske on Unsplash

2. How do NLP models leverage the Transformer architecture, and what makes them different?

The Transformer model structure has largely replaced other NLP model implementations such as RNNs [3]. Current SOTA **NLP models use the Transformer architecture in part or as a whole**. The GPT model only uses the decoder of the Transformer structure (unidirectional) [2], while BERT is based on the Transformer encoder (bidirectional) [4]. T5 utilizes an encoder-decoder Transformer structure very similar to the original implementation [3]. These general architectures also differ in the number and dimension of the elements that comprise an encoder or decoder (i.e., the number of layers, the hidden size, and the number of self-attention heads they employ [4]).

Aside from these variations in model structure, the **language models also diverge in the data and tasks they used for pre-training**, which the following paragraph will elaborate on further. For more details on the exact model architectures and their pre-training design, you can refer to the academic papers introducing these models:

- [2]: GPT

Photo by Tim Mossholder on Unsplash

3. What are transfer learning, pre-training, and fine-tuning?

Many Transformer-based NLP models were specifically created for transfer learning [3, 4]. **Transfer learning** describes an approach where a model is first pre-trained on large unlabeled text corpora using self-supervised learning [5]. Then it is minimally adjusted during fine-tuning on a specific NLP (downstream) task [3]. Labeled datasets for specific NLP tasks are usually relatively small. Training a model only on such a small dataset without pre-training would lower results compared to its pre-trained version [2]. The same pre-trained model can be used to fine-tune various NLP downstream tasks, including text classification, summarization, or question answering.

Various unlabeled data sources can be utilized for **pre-training**. They can be entirely unrelated to the data or task during fine-tuning as long as the dataset is large enough. GPT-2 was pre-trained using 40 GB of text [6]. Consequently, pre-training is very time-

[2] which predicts the next word in a sentence, BERT was trained on Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) [4]. The RoBERTa model replicated the BERT model architecture but changed the pre-training using more data, training for longer, and removing the NSP objective [7].

The model checkpoints of the pre-trained models serve as the starting point for **fine-tuning**. A labeled dataset for a specific downstream task is used as training data. There are several different fine-tuning approaches, including the following:

1. Training the entire model on the labeled data.
2. Training only higher layers and freezing the lower layers.
3. Freezing the entire model and training one or more additional layers added on top.

Two additional types of fine-tuning (adapter layers and gradual unfreezing) are explained in [3]. No matter the approach, **a task-specific output layer usually needs to be attached to the model**. This layer is unnecessary if you only fine-tune the model on a new text dataset, but the learning objective stays the same. This could be the case if you use GPT-2 on a specific text dataset for language generation.

Several research papers introducing the pre-trained Transformer-based models also ran fine-tuning experiments to demonstrate their transfer learning performance. **BERT**, for example, was trained on 11 NLP tasks **fine-tuning all parameters in the entire model and feeding outputs to a task-specific output layer**.



Photo by Jess Bailey on Unsplash

4. Are there approaches other than transfer learning?

Multi-task learning is intended to increase the model generalization across tasks and domains [8]. The same model can perform various downstream NLP tasks without fine-tuning or otherwise changing the model parameters or architecture [6]. GPT-2 was trained on a large and diverse text dataset using only a language modeling objective. It reached SOTA results for various NLP tasks without being fine-tuned on those tasks. Only a few examples were provided during inference time to help the model understand what task was requested [6].

Meta-learning is particularly well suited for situations where a task-specific dataset only has a few samples (low-resource tasks). It enables models to develop a good initialization and a broad set of skills that rapidly adapt to new or previously seen tasks during inference [9, 10]. A key difference between multi-task learning and meta-learning is that the performance of multi-task learning is skewed towards tasks with large datasets. Meta-learning adapts well to any task [10]. Similar to multi-task learning, no fine-tuning is necessary for meta-learning models.

inference time to the model [2].

Photo by Jazmin Quaynor on Unsplash

5. Where can you access these models?

Most of the Transformer-based models developed by researchers are open source (except GPT-3 for now). You can find their model implementations on GitHub (e.g., for BERT). However, if you want to apply a model to your own NLP task, you need the **model checkpoint**. You can access these via the **transformers library provided by Hugging Face**. This library gives you access to more than 32 pre-trained SOTA models. It provides an API that allows you to comfortably integrate the models into your code using PyTorch or TensorFlow.

Hugging Face is used by organizations such as Google, Microsoft, and Facebook. However, **its transformers library is great for beginners** as well. It has extensive documentation and various Google Colab notebooks providing example implementations of text classification using DistilBERT, for example. They also walk you through more general examples of how to train and fine-tune a model

the documentation explains which models work with which task-specific pipeline. Hugging Face even created a **zero-shot classification pipeline**. It takes any text input and number of labels and returns the probability for each of those labels. It is not necessary that any of those labels were seen during fine-tuning (hence the zero-shot name).

Photo by Danielle MacInnes on Unsplash

6. How can you apply these models to your NLP task?

If you know what NLP task you want to conduct, you will need to **determine which Transformer-based model would be best suited for this purpose**. GPT-2, for example, was pre-trained and designed for language generation with its unidirectional architecture, making it most useful for language generation tasks. However, it is also possible to fine-tune it for sequence classification, for example.

You can **consult the original academic papers of these models to get an idea of what tasks they were fine-tuned for and where they performed particularly well.** You can also check the Hugging Face transformers docs that elaborate on the various

right away (see the pipeline objects of Hugging Face). Aside from the 32+ pre-trained models provided by Hugging Face, its community provides more than 9,000 model specifications.

You will **have to be familiar with neural networks (NN) and deep learning at least a little** to fine-tune a model in most cases. Remember, you usually add a task-specific NN layer to the pre-trained model and implement a training process. Moreover, you need to turn your dataset into the right data structure to be compatible with your training process. If you haven't worked with NNs yet, I recommend you read up on them here:

- Machine Learning for Beginners: An Introduction to Neural Networks by Victor Zhou
- Intro to Deep Learning on Kaggle
- First neural network for beginners explained (with code) by Arthur Arnx

To **get started on a specific task and fine-tuning your model**, take a look at these example implementations for various downstream NLP tasks:

- Fine Tuning GPT-2 for Magic the Gathering Flavour Text Generation by Richard Bownes
- Fine Tuning a T5 transformer for any Summarization Task by Priya Dwivedi
- Fine-Tuning BERT with HuggingFace and PyTorch Lightning for Multilabel Text Classification | Dataset by Venelin Valkov

Photo by Nick Fewings on Unsplash

Conclusion

Leveraging Transformer-based NLP models seems daunting at first. Many articles have been published that explain the concepts related to these models in great detail. However, **before I dive into those details, I like to get a broader overview.** This high-level overview of the most important elements relating to the models, the training process, and applying it to your own NLP use case hopefully elucidated how to use them. For my text classification task, I ended up using the Longformer model and fine-tuned it using PyTorch Lightning.

Do you want to get started with Transformer-based NLP models, but you don't have a specific project idea yet? Why don't you check out my **5-step guide on how to come up with unique data science project ideas**:

5 Steps to Develop Unique Data Science Project Ideas

A guide that helps you identify new and unique data projects that are worth your while

towardsdatascience.com

References to academic papers

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. ArXiv:1706.03762 [Cs]. http://arxiv.org/abs/1706.03762

[2] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). *Improving language understanding by generative pre-training*. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[3] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. ArXiv:1910.10683 [Cs, Stat]. http://arxiv.org/abs/1910.10683

[4] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ArXiv:1810.04805 [Cs]. http://arxiv.org/abs/1810.04805

[5] Aßenmacher, M., & Heumann, C. (2020). On the comparability of Pre-trained

https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[7] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. ArXiv:1907.11692 [Cs]. http://arxiv.org/abs/1907.11692

[8] Worsham, J., & Kalita, J. (2020). Multi-task learning for natural language processing in the 2020s: Where are we going? *Pattern Recognition Letters*, *136*, 120–126. https://doi.org/10.1016/j.patrec.2020.05.031

[9] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). Language Models are Few-Shot Learners. *ArXiv:2005.14165 [Cs]*. http://arxiv.org/abs/2005.14165

[10] Dou, Z.-Y., Yu, K., & Anastasopoulos, A. (2019). Investigating Meta-Learning Algorithms for Low-Resource Natural Language Understanding Tasks. *ArXiv:1908.10423 [Cs]*. http://arxiv.org/abs/1908.10423