UIUC CS410 Text Information Systems
Technology Review
Huggingface Transformers

Anthony Petrotte
adp12@illinois.edu

Recent developments in machine learning and deep learning architectures have opened up a range of possibilities for analyzing and drawing valuable insights from data. In conjunction with the vast amount of publicly available information on the internet, one of the most powerful tools to come from these developments has been advanced techniques for natural language processing. The value here is inherent in the fact that human beings use textually based information to communicate, express themselves and store a wide array of information. Although this information is usually quite easy for the human mind to interpret, the ambiguity of textual information creates unique challenges for creating accurate and repeatable computational processes to interpret text.

New and unique challenges of course warrant new and unique solutions, and in 2017 a revolutionary neural network architecture called the Transformer was proposed by Vaswani et al [1]. This architecture has quickly become the defacto standard for natural language processing, taking the place of many models such as Recurrent Neural Networks [2][4]. In short, the reason for this lies in the recurrent nature of RNNs making them harder to train in parallel and the issue of vanishing gradients, both issues that the Transformer architecture does not have [2][4].

Anyone with a fairly modern computer and the will to do so could create an implementation of most modern neural network architectures using open-source libraries such as Keras and Pytorch. However, apart from the extensive amount of work this could entail, particularly if one's goals were not to create the model itself but were instead to want to use a model for a downstream task such as text classification, implementing these models comes with a huge catch: training the model.

Training NLP models is particularly expensive both computationally and financially. The most advanced (and subsequently accurate) NLP models tend to be trained using stacks of state-of-the-art processors coupled with advanced parallel frameworks on millions if not billions of parameter values [5]. This kind of processing power does not come cheap. For example, the cost of a single training run of a large parameter variant of Google's T5 model are estimated at well above a million dollars [3]. To reiterate for emphasis, that cost is for a single training run on one specific model. This creates a huge barrier for many companies (and certainly most individuals) to implement accurate NLP models for various applications.

This is where Huggingface comes to the rescue of the average researcher, data scientist, or starving CS student. Anyone with experience coding, particularly in Python or R, knows the power of a good importable code library. Without well documented and standardized libraries to accomplish complicated computational tasks, advancements in data science would be stunted, even the most prestigious CS educations would become disjointed and therefore hold less utility, and modern analytical research would be plunged back into the dark ages of non-reproducibility. The same open-source principle may hold true for the power of accurate NLP, and Huggingface, despite its silly name, would appear to be taking up the mantle for this task.

Huggingface, in short, provides an expansive library of popular NLP models in an easy to use API. Although that in itself is an amazing leap for the open-source learning community, it really is just the tip of the iceberg. Firstly, I acknowledge that many NLP architectures can potentially be found pretrained by their respective authors online, but the implementation of these models into existing code could be an extensive extra amount of work. When further considering issues with compatibility, you may end up with poor implementations of a good model or even using a model not right for the project simply due to its availability. Huggingface's Transformer library on the other hand was designed to be as fast and easy to use as possible, minimizing the time to implement NLP solutions into code. They do this via an API to a huge selection of models that are all pre-trained via their respective sources. These are not imitations of state-of-the-art models either, they are in many cases the actual models themselves provided by the official authors of the architecture [b].

For example, Google's T5 model mentioned earlier to express the large cost of creating modern NLP models is one of the many included in Huggingface's accessible models. In fact, the most recent version of Huggingface(4.14.2 at time of writing) has over 80 accessible pretrained state-of-the-art models. The models themselves can be accessed in a way similar to selecting the parameters of a function. This is actually quite the feat considering these models were created at different times with different compatibilities. This does mean that in some cases the code of the model coming out of Huggingface does not perfectly match the code of the original source in order to ensure the seamlessness of the library. However, the Huggingface team has made it a point to keep the code "as close to the original code base as possible". This discrepancy between the original model's code seems insignificant when you consider that all of the models in the library can use the same resource functions and classes from the library. This in a sense standardizes the models under a singular roof and similar language structure, making it many factors easier to compare model outputs, change architectures when necessary and adapt to the changing needs of a complex project [b].

The unification makes instantiating a state-of-the-art model a simple function call and each of the models requires only three standard classes to use. It's so easy that behind the scenes, the necessary model will be downloaded and all needed caching and loading of associated data is handled. This makes actually using the model just as fast as promised, and utilizing the AutoTokenizer feature allows you to get the corresponding tokenization scheme and tokenize the data correctly for the instantiated model. This actually means you can go from model instantiation to a usable output from that model with a few lines of code [b].

In addition to using pre-trained models, you can also easily fine tune the models to better fit your specific use case using built in training functions. This is a much need addition considering a model, even if suited for a task in its architecture and training corpus cannot always perfectly fit each task. However, as valuable as this is, it does not allow for expansion of the provided architecture, simply to tune it. Loss of the modularity of the architecture is the sacrifice you pay for speed and ease of use. That being said, Huggingface has also made it a point to provide the internal structure of the models, including the hidden-states and attention weights. Given this, if someone were to want to build on the model's architecture, this would still be possible, just not within the confines of the Huggingface library. After adjusting the model though, using the simplicity of Huggingface's interface would still be just as easy using the new model. The built in functions allow you to actually save models locally and instantiate models from the library or locally just as easily, adding a whole layer of utility [b].

Although there are other API libraries available that certainly compete with Huggingface in the pre-trained NLP model API field such as spaCy or Standford's Stanza, Huggingface creates a hugely unique value with a community driven approach to NLP models. Huggingface encourages those who have expanded and fine-tuned models for different uses to share those models in their ever expanding list of community-models. This allows for unprecedented access to specific NLP models for different tasks and the ability to share and compare those models with peers online, further speeding up the process of implementing complex NLP models into code. At the time of this writing, the community-models library has 19,900 different models available.

Lastly, I would like to mention that Huggingface's value stands out in not just the amount of models in the library, but in the breadth of the provided resources and the sheer quality of their documentation. The documentation in particular is plush with in depth examples and guides for almost every function and even every model. There are also guides to assist in model selection to ensure correct selection for specific use cases.

In the last 5 years, the accuracy and scale of NLP architectures have exploded [2], creating new and exciting ways to leverage the most plentiful data source that humanity has accumulated. It should be relieving that the online open-source community has been made available with these cutting edge models and that despite the cost to create and train [3], the authors of the models have made available many models(though obviously not all) to the greater community to expand the collective knowledge of humanity. Huggingface is paving the way for faster and easier implementations of NLP in code and almost completely eliminates the barriers previously inhibiting the majority of companies and individuals from leveraging advanced NLP techniques. In fact, over 5,000 companies are reportedly using Huggingface, including major names like Microsoft, Google, and Facebook [a]. Huggingface (despite its name) has been propelled to a unique position in the NLP space, and is setting itself up to be a vital hub for major efficiencies throughout society that NLP can provide.

References

[a] Huggingface. https://huggingface.co/.
Accessed: 11-07-2021

[b] Huggingface Transformers. https://huggingface.co/transformers/.
Accessed: 11-07-2020

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin,
“Attention Is All You Need,”
arXiv:1706.03762 [cs], Dec. 2017. arXiv: 1706.03762.

[2] A. Gillioz, J. Casas, E. Mugellini and O. A. Khaled,
"Overview of the Transformer-based Models for NLP Tasks,"
2020 15th Conference on Computer Science and Information Systems (FedCSIS), 2020, pp. 179-183, doi: 10.15439/2020F20

[3] O. Sharir, B. Peieg, Y. Shoham,
"The Cost Of Training NLP Models,"
arXiv:2004.08900 [cs.CL], Apr. 2020. arXiv:2004.08900v1

[4] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, A. Rush,
"Transformers: State-of-the-Art Natural Language Processing"
arXiv:1910.03771 [cs.CL], Jul. 2020. arXiv:1910.03771v5

[5] V. Sanh, L. Debut, J. Chaumond, T. Wolf
"DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter,"
arXiv:1910.01108 [cs.CL] Oct. 2019. arXiv:1910.01108v4