Another desirable feature of self-attention is that it creates a representation for each token that is dependent on its surrounding tokens. This makes the representation of each token context aware, such that the representation of the word “apple” (fruit) is different from “apple” (computer manufacturer). This feature is not novel about the transformer architecture and previous architectures such as ELMo also used contextualized representations. Updating the token representations with self-attention and feed-forward networks is then repeated across several layers or “blocks” to produce a rich encoding which is combined with the decoder inputs. These layers are similar for the encoder and decoder part of the Transformer architecture and we will have a closer look at their inner workings in Chapter 3. Abandoning recurrence and replacing it with self-attention and feed-forward networks also greatly improves the computational efficiency of transformer models. Research into the scaling laws of deep learning models has revealed that larger models trained on more data in many cases yield better results. The scalability of transformers enables the full exploitation of the scaling laws which has started a scaling race of NLP models. But scaling models comes at the price of requiring large amounts of training data. When working on practical applications of NLP we usually do not have access to large amounts of textual data to train such large models on. A final piece was missing to get the transformer revolution started: transfer learning. Transfer Learning in NLP It is common practice in computer vision to use transfer learning to train a convolutional neural network like ResNet on one task and then adapt or fine-tune it on a new task, thus making use of the knowledge learned in the original task. Architecturally, this usually works by splitting the model in terms of a body and head, where the head is a task-specific network. During pretraining, the weights of the body learn broad features of the source domain, and it is these weights which are used to initialize the new model for the new task. Compared to traditional supervised learning, this approach typically produces high-quality models that can be trained much more efficiently on a variety of downstream tasks, and with much less labeled data. A comparison of the two approaches is shown in Figure 1-7 Hugging Face Transformers: Bridging the Gap Applying a novel machine learning architecture to a new task can be a complicated undertaking, and usually involves the following steps: Implement the model architecture in code, typically in PyTorch or TensorFlow; Load the pretrained weights (if available) from a server; Preprocess the inputs, pass them through the model, and apply some task□specific post□processing; Implement data loaders and define loss functions and optimizers to train the model. Each of these steps requires custom logic for each model and task. Traditionally, research teams that publish a new model will often release some of the code (but not always!) along with the model weights. However, this code is rarely standardized and requires days of engineering to adapt to new use-cases. This is where the Transformers library came to the NLP practioner’s rescue: it provides a standardized interface to a wide range of transformer models as well as code and tools to adapt these models to new use-cases. The library integrates all major deep learning frameworks such as TensorFlow, PyTorch, or JAX and allows you to switch between them. In addition it provides task□specific “heads” so you can easily fine-tune transformers on down-stream tasks such as classification, named entity recognition, question-answering etc. together with modules to train them. This reduces the time for a practitioner to train and test a handful of models from a week to a single afternoon! See for yourself in the next section where we show that with just a few lines of code, the Transformers library can be applied to tackle some of the most common NLP applications that you’re likely to encounter in the wild.

I had better say something here about this question of age, since it is particularly important for mathematicians. No mathematician should ever allow himself to forget that mathematics, more than any other art or science, is a young man's game. To take a simple illustration at a comparatively humble level, the average age of election to the Royal Society is lowest in mathematics. We can

naturally find much more striking illustrations. We may consider, for example, the career of a man who was certainly one of the world's three greatest mathematicians. Newton gave up mathe□matics at fifty, and had lost his enthusiasm long before; he had recognized no doubt by the time he was forty that his greatest creative days were over. His greatest idea of all, fluxions and the law of gravitation, came to him about 1666 , when he was twentyfour—'in those days I was in the prime of my age for invention, and minded mathematics and philosophy more than at any time sine'. He made big discoveries until he was nearly forty (the 'elliptic orbit' at thirty-seven), but after that he did little but polish and perfect. Galois died at twenty-one, Abel at twenty-seven, Ramanujan at thirty□three, Riemann at forty. There have been men who have done great work a good deal later; Gauss's great memoir on differential geometry was published when he was fifty (though he had had the fundamental ideas ten years before). I do not know an instance of a major mathematical advance initiated by a man past fifty. If a man of mature age loses interest in and abandons mathematics, the loss is not likely to be very serious either for mathematics or for himself. On the other hand the gain is no more likely to be substantial: the later records of mathematicians are not particularly encouraging. Newton made a quite competent Master of the Mint (when he was not quarrelling with anybody). Painlevé was a not very successful Premier of France. Laplace's political career was highly discreditable, but he is hardly a fair instance since he was dishonest rather than incompetent, and never really 'gave up' mathematics. It is very hard to find an instance of a first-rate mathematician who has abandoned mathematics and attained first-rate distinction in any other field. There may have been young men who would have been first-rate mathematician if they had stuck in mathematics, but I have never heard of a really plausible example. And all this is fully borne out by my very own limited experience. Every young mathematician of real talent whom I have known has been faithful to mathematics, and not form lack of ambition but from abundance of it; they have all recognized that there, if anywhere, lay the road to a life of any distinction. There is also what I call the 'humbler variation' of the standard apology; but I may dismiss this in a very few words. (2) 'There is nothing that I can do particularly well. I do what I do because it came my way. I really never had a chance of doing anything else.' And this apology too I accept as conclusive. It is quite true that most people can do nothing well. If so, it matters very little what career they choose, and there is really nothing 1 Pascal seems the best more to say about it. It is a conclusive reply, but hardly one likely to be made by a man with any pride; and I may assume that none of us would be content with it.