A Compendium of Thoughts: **Conundrum of Model Selection**

Nowadays, there is an abundance of pre-trained and fine-tuned models that are available in popular hubs, such as Hugging Face or Mosaic ML. Ideally, you seek a model that's fine-tuned for specific task which you're looking for. Other criteria for a right fit model are how well it will be integrated into your downstream without too much engineering. Most AI and ML projects fail due to the fact that they neglect to consider the transition points between MLOps and the overall service, as well as the integration process with necessary stopgaps and essential business logic. This is a complex topic and deserves a deep dive some other day.

When considering which model you should pick, consider the following dimensions:

- Fit
- Speed
- Complexity
- Size

With all the buzz surrounding ChatGPT and Generative AI, if you'd like to impress your executives, you may opt to showcase the versatility of GPT3/4, comparing it to a Swiss Army Knife. If you go this route, be prepared to utilize it as a service or have H100/A100 lying around for fine-tuning. Even with a pay-as-you-go model, a top-tier cloud provider may charge around $27.20/hr. Keeping costs aside, a savvy cyber or risk governance will throw a curveball at you if you opt for Open AI as a service based on an information leak.

Let's take the simple task of name entity recognition (NER) which is a basic NLP task. As mentioned earlier, you will use a foundational fine-tuned model for a specific task. My idea is to take a simple task and provide an empirical way for you to come to a conclusion about which model is right for you. I have provided code via EmpricalModelSelection repo for this specific case. You can adopt this methodology for your own task.

Before diving into methodology, I want to address the following question: can I use OpenAI (ChatGPT) for simple tasks like NER? The answer is yes, by using a zero-shot learning and tools/library such as LangChain or spaCy. We create a simple schema and create an extraction chain and run. You can refer to openAINER.ipynb. If you want to refine the results with prompts, I would recommend you take a look at this PromptNER paper. This provides example for prompts including "Conll," which we'll be using as a base dataset. I also came across Promptify, which is a library that provides a simple API for prompting.

To compare models, let us pick two fine-tuned models from Hugging Face. We'll build a transformer from scratch using Keras. We use "Conll2003" data for validation of our models. There are a couple of reasons why we chose this dataset. One major reason is that this dataset is simple, does not have lengthy, complicated sentences, and has been used for benchmarking for a couple of decades. The second and most important reason is that fine-tuned models from Hugging Face for NER task were using the same dataset. For the custom transformer model, we

build vocabulary from the base dataset and train the model using the same dataset. We will carve out 3,250 records as a validation portion.

We use Hugging Face NER pipeline with dslim/bert-base-NER and dbmdz/bert-large-cased-finetuned-conll03-english fine-tuned models. From the size point of view, bert-base-NER has 12 layers and around 107 million parameters whereas bert-largecased-finetuned-conll03-english has 24 layers and around 332 million parameters, which is a considerable footprint. On the contrary, a single block custom transformer model with 6 attention heads is 1.4 million parameters. When we compare the amount of time taken to infer 3,250 records, the bert-base-NER took around 2 minutes and 20 seconds while the bert-largecased-finetuned-conll03-english took nearly 6 minutes. One thing you need to consider is the use of pipeline which makes the process very clean and easy to maintain; this comes with some additional overhead. The custom transformer took just around 3 seconds. This is very quick, but we must develop and train from scratch. This will not be a challenge for a simple task like NER, but if it's a complex task such as question-answer or multi-modal then there is always a place for complex pre-trained or fine-tuned models. I want you to pay attention to bert-largecased model even though fine-tuned on the same dataset is not performing in the desired manner. One reason is its extensive vocabulary and the way in which it parses causes issues to align to the standard conll03 output. The poor results could also stem from the fact that the model is not fine-tuned properly to begin with (you can further explore aggregation_strategy of the pipeline and custom consolidate entities to get better results). So, you cannot simply assume that models are fine-tuned properly as you will not have any good metrics enforced in the Hugging Face hub. Anyone can upload a model claiming it is fine-tuned.

In conclusion, it is necessary to assess whether the chosen model is optimal for the task and has undergone fine-tuning. Compare its performance to that of a custom model and consider the trade-off between engineering complexity and ease of maintenance. Evaluate the significance of inference response time and establish a clear model life cycle strategy. These elements play a critical role in making the correct decision for selecting the appropriate model.