WHAT IS ByT5?

- Token free models, that operate directly on raw text (bytes or characters).
- Byte-level model by definition only requires 256 embeddings.
- It trains encoder decoder models that extend to generative tasks
- The models work directly with UTF-8
- It explores the effect of model scale, training models beyond 10 billion parameters.(ByT5- XXL)
- Migrating word representations, out of sparse vocab matrix and into dense network layers, allows the model to generalize more effectively.
- Token free models can process any text sequence.

- **Tokenization (used by LLM models)** = A piece of text is converted into a sequence of tokens by a tokenizer. It uses a fixed vocabulary of words, so we cannot process an out of vocabulary word. Sol- is to map all unknown words to <UNK> token but then it cannot distinguish between the different out of vocab words.
- **Subword Tokenizer(used by LLM models)** = It decomposes words into smaller subwords.
  E.g. 'Doghouse' gets divided into 2 tokens 'dog' and 'house'. It causes lot of typos and mispredictions and doesn't deal well with characters.

Byt5 Design

- It is based on the mT5 model which was trained on mC4
- It is a general purpose pre-trained text to text model covering 100+ languages similar to mT5
- It has 5 sizes- small, base, large, XL,XXL
- Useful for tasks operating on short to medium length

Changes made to mT5

- Dispense with the SentencePiece vocab and feeds UTF-8 bytes directly in to the model without any text preprocessing.
- Bytes are embedded to the model hidden size with a vocab of 256
- 3 additional IDs reserved for special tokens: - padding, end of sentence, unused token <UNK>

Changes made in pretraining

MT5

1. Uses “Span corruption” pretraining objective (Spans of tokens in unlabeled text data are replaced with a single “sentinel” ID and the model must fill in the missing spans)
2. It uses an average span length of 3 subword tokens.
3. It uses a “balanced” architecture (for encoder and decoder)
4. Sequence length= 1024
5. Train steps= 1 million
6. Token batches = 2^20 tokens

Byt5

1. Instead of adding 100 new tokens for sentinels, it reuses the final 100 byte IDs.
2. It uses a mean mask span length of 20 bytes as it finds masking longer byte spans more valuable
3. Works best by decoupling the depth of the encoder and decoder stacks.
4. Byte- level models benefit significantly from a heavier encoder, therefore encoder depth to 3 times that of decoder
5. Similar to encoder only models like BERT
6. Since all bytes are not legal according to the UTF-8 standard, illegal bytes in the models output are dropped
7. ByT5 model hidden size(d_{model}) and feed forward dimensionality (d_{ff}) to be parameter matched with mT5, and maintains a ratio of 2.5 between d_{ff} and d_{model}.
8. Rest is same as mT5 from point 4 onwards

Why is Byt5 better?

- ByT5 is better than traditional LLMs like GPT-3, T5, and mT5 because it processes text at the byte level instead of relying on tokenization, which often leads to information loss—especially in rare words, multilingual text, or noisy inputs. Models like GPT-3 and mT5 depend on subword vocabularies, which can fragment or omit important details. In contrast, ByT5 reads raw bytes directly, making it more robust, language-agnostic, and better suited for diverse and real-world data.

- The SoftMax and vocab output matrices in the mT5 base model amount to about 256 million parameters which is about 66% of the total parameters. Byte level models let us allocate these parameters elsewhere like to add new layers or to widen the existing layers

- Data efficiency- during pretraining, it is exposed to 4 times less actual text than a mT5 model

- Higher robustness to noise across tasks and languages(Experiment on synthetic noise) – (1) Characters are randomly dropped (2) Characters are randomly repeated (3) Characters are Capitalized and padded with a space (4) Characters are randomly assigned a case. ByT5 performs better

- Correct prediction of non alpha numeric characters. T5 wasn't able to predict characters like ">" or "|" and was creating typos like "FF" or "pouu" while ByT5 made the correct predictions of characters ">" and words like " pout"

- It outperforms mT5 in several scenarios like
  a. model sizes under 1 billion parameters
  b. on generative tasks,
  c. on multilingual tasks,
  d. on word-level tasks sensitive to spelling and/or pronunciation,
  e. in the presence of various types of noise

Drawbacks

- Byte sequences are longer than token sequences
  E.g. : A word like "hello" might be 1 token in a subword tokenizer.But in byte-level, it's 5 separate tokens: h, e, l, l, o.

Solution - Convolutions with pooling or adaptive computation time

- Byte sequences increases the (tokenized) sequence length of a given piece of text, resulting in a significantly higher computational cost.

Solution - Use easy to measure quantities like parameter count and FLOPs
  - Not all FLOPs are equal
  - Real-world cost of a model depends on the hardware it runs on
  - E.g. For byte level encoder-decoder model, if the decoder is particularly large, autoregressive sampling becomes comparatively expensive, due to the increased length of byte sequences.(Encoder runs in parallel and is fast but the decoder runs step by step - predicts one byte, waits , predicts the next byte)
    Solution - Map an input token to its corresponding vector representation in the vocab matrix which is free to do(in terms of FLOPs), since it can be implemented by addressing a particular row in memory

Reference

- Xue, L., Barua, A., Constant, N., Al-Rfou, R., Narang, S., Kale, M., Roberts, A., & Raffel, C. (2022). ByT5: Towards a Token-Free Future with Pre-trained Byte-to-Byte Models. *Transactions of the Association for Computational Linguistics*, *10*, 291–306. https://doi.org/10.1162/tacl_a_00461