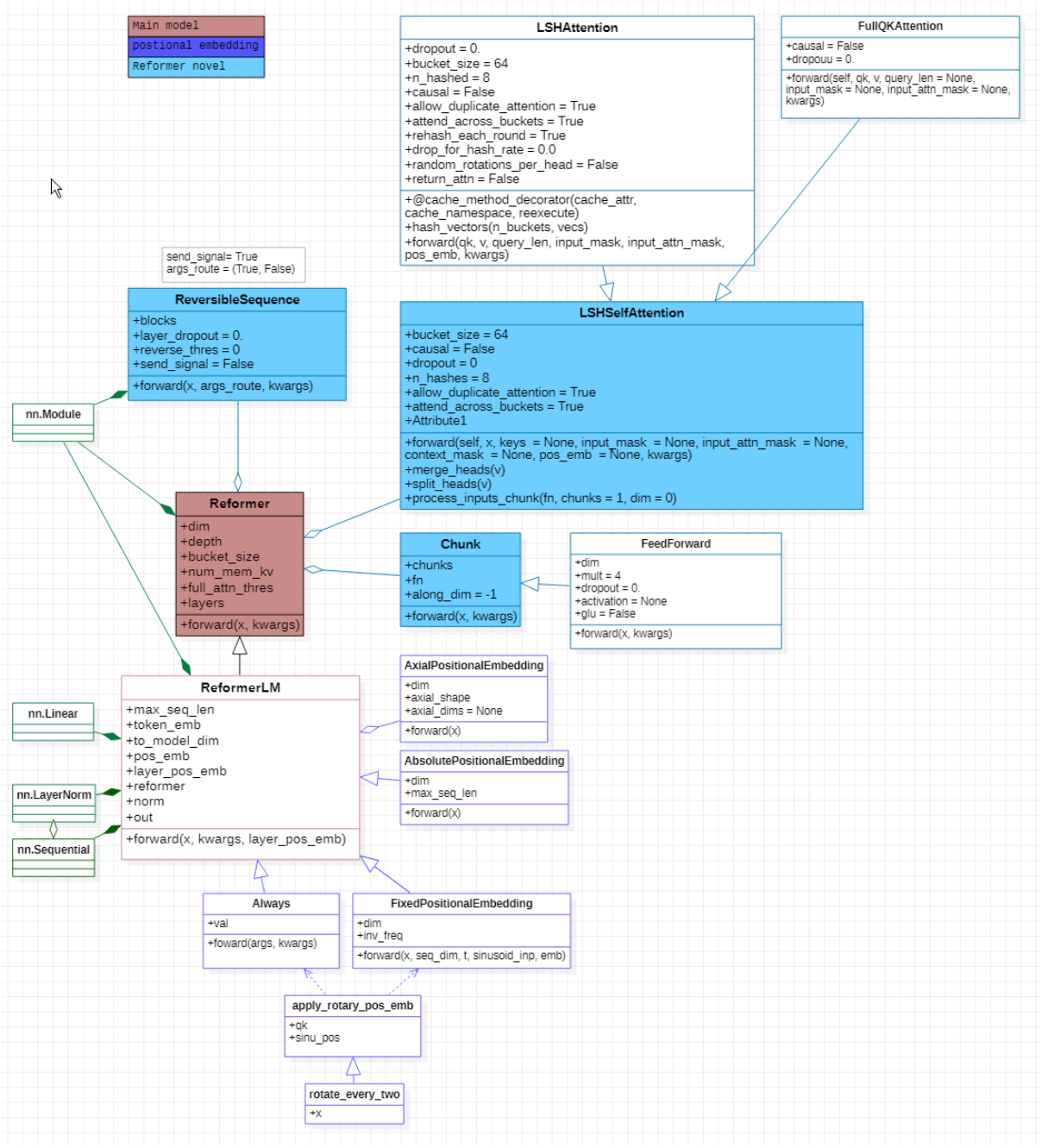
Reformer

Author	Nikita Kitaev
Created time	@2021년 4월 14일 오전 7:19
Date	
Link	https://arxiv.org/pdf/2001.04451.pdf
Organization	Google Research
Presented @	ICLR 2020
Property	LSH attention, Reversible Network for FFN, Chunking FFN
Published	
Remarks	
Tags	
Tags 1	
Tags 1 1	
Type	https://blog.pingpong.us/reformer-review/

Licidrain code : https://github.com/lucidrains/performer-pytorch/tree/968d3340a6a6d0cfd5bc208974bec85aa270e071

UML as an overview of Reformer code



LSHSelfAttention

- notations
  - b, t, e = *x.shape,
  - h = self.heads,
  - dh = dim_head

    - m = self.num_mem_kv
    - l_h = self.n_local_attn_heads
- attention function을 고를 수 있게 되어있음
    - `attn_fn = self.lsh_attn if not use_full_attn else self.full_attn`

LSHAttention

<forward>

- n_buckets = seqlen // self.bucket_size
- buckets = self.hash_vectors(n_buckets, qk, key_namespace=depth, fetch=is_reverse, set_cache=self.training)

<hash_vectors>

- vectors 를 가지고
    1. dropped_vecs : dropped_vec를 만듬
    2. rotated_vecs :
        - rotations_shape은 random_rotations_per_head가 True이면 batch_size만큼으로 시작 아니면 1로 시작.
        - 이 shape으로 radom_rotation이라는 torch.randn를 만듬.
        - dropped_vecs와 random_rotation으로 아인슈타인서메이션을 통해 batch matrix multiplication 시행.
- 이렇게 얻어진 rotated_vecs들은 torch.cat([rotated_vecs, -rotated_vecs], dim=-1) 작업
- rotated_vecs를 가지고 map each item to the top self.n_hashes buckets 함.
  `buckets size [batch size, seq_len, buckets]`

FullQKAttention

simple full attention

Chunk

- FeedForward Layer에 대하여...
- 만약 chunk 수가 1이면 그냥 바로 해당 function 리턴
- 아니면 chunks = x.chunk(self.chunks, dim = self.dim)
    - torch.cat([self.fn(c, **kwargs) for c in chunks], dim = self.dim)
        - input을 chunkify and return every function with each chunk

---