Code Work

팀 데컬코마니

1. 크롤링 과정

별도 파일 첨부

1. Feeder.py \_decal\_topic\_sent\_sample\_append() 함수
2. *def* \_decal\_topic\_sent\_sample\_append(*self*, *text*, *length*, *text\_file*):
3. sentences = split\_text\_into\_sentences(*text*)
4. sentences\_tokenized = [*self*.tokenizer.morphs(sent) for sent in sentences if sent is not '']
5. begin\_index = random.randrange(0, len(sentences\_tokenized)-2) if len(sentences\_tokenized) > 2 else 0
6. selected\_sents = sentences\_tokenized[begin\_index:]
8. folders = [folder for folder in  Path(*self*.data\_dir\_path).glob('\*')]
9. for folder in folders:
10. try:
11. topic\_folder = [files for files in Path(folder).rglob('\*.txt')]
12. file\_path = [file for file in Path(folder).glob(*text\_file*)]
13. if file\_path[0] in topic\_folder:
14. topic\_path = topic\_folder
15. topic\_path.remove(file\_path[0])
16. break
17. except:
18. continue
19. token\_ids = *self*.tokenizer.convert\_tokens\_to\_ids([token for sent in selected\_sents for token in sent])
20. while len(token\_ids) < *length*:
21. appending\_text = random.choice(topic\_path).read\_text(*encoding*='utf-8')
22. token\_ids += *self*.tokenizer.convert\_tokens\_to\_ids(*self*.tokenizer.morphs(appending\_text))
23. return token\_ids[:*length*]
24. Sampler.py top\_p\_and\_k\_logits() 함수
25. *def* top\_p\_and\_k\_logits(*logits*, *p*, *k*, *temperature*):
26. if *k* == 0:
27. # no truncation
28. return *logits*
29. *def* \_top\_k(*log*):
30. values, \_ = tf.nn.top\_k(*log*, *k*=*k*)
31. min\_values = values[:, -1, tf.newaxis]
32. return tf.where(
33. *log* < min\_values,
34. tf.ones\_like(*log*, *dtype*=*logits*.dtype) \* -1e10,
35. *log*,
36. )
38. with tf.variable\_scope('top\_p\_logits'):
39. logits\_sort = tf.sort(*logits*, *direction*='DESCENDING')
41. probs\_sort = tf.nn.softmax(logits\_sort/*temperature*)
42. probs\_sums = tf.cumsum(probs\_sort, *axis*=1, *exclusive*=True)
43. logits\_masked = tf.where(probs\_sums < *p*, logits\_sort, tf.ones\_like(logits\_sort)\*1000) # [batchsize, vocab]
44. # 누적확률까지 맞으면 logits\_sort 유지, 아니면 1000으로 변경.
45. # 그다음 이걸 reduce\_min -> 최소 확률 찾음
46. min\_logits = tf.reduce\_min(logits\_masked, *axis*=1, *keepdims*=True) # [batchsize, 1]
47. # min\_logit보다 크거나 같아야 남겨줌.
48. top\_p\_result = tf.where(
49. *logits* < min\_logits,
50. tf.ones\_like(*logits*, *dtype*=*logits*.dtype) \* -1e10,
51. *logits*,
52. )
53. #갯수 카운트용
54. for\_count = tf.where(
55. top\_p\_result==tf.constant([-1e10], *dtype*=top\_p\_result.dtype),
56. tf.zeros\_like(top\_p\_result, *dtype*=top\_p\_result.dtype),
57. top\_p\_result
58. )
59. cnt = tf.count\_nonzero(for\_count)
60. return tf.cond(
61. tf.less\_equal(cnt,*k*),
62. *lambda*: top\_p\_result,
63. *lambda*: tf.cond(
64. tf.equal(*k*, 0),
65. *lambda*: *logits*,
66. *lambda*: \_top\_k(top\_p\_result),
67. )
68. )