Part 1

(a) m을 사용하면 단순히 각 단계의 gradient를 사용하는 것이 아니라 이전 gradient와 평균을 취하기 때문에 변동성이 줄어든다. 변동이 적으면 학습 과정에서 parameter가 더 일관된 방향으로 update가 되어서 더 빠르고 안정적으로 최적화된다.

(b) 모델 parameter 중에서 크기가 작은 parameter들이 더 크게 update된다.
큰 gradient를 가진 parameter들의 update를 줄여서 overfitting을 방지할 수 있다.
작은 gradient를 가진 parameter들의 update를 크게 해서 학습의 균형을 맞춘다.

(c) $E[d_i] = 0 \times p_{drop} + 1 \times (1 - p_{drop}) = 1 - p_{drop}$

$E[(d \odot h)_i] = (1 - p_{drop}) h_i$

$\gamma (1 - p_{drop}) h_i = h_i$ 이 성립하려면 $\gamma = \frac{1}{1 - p_{drop}}$ 이어야 한다.

훈련 중 dropout은 모델의 overfitting을 방지하고 더 일반화된 모델을 학습할 수 있도록 한다.
그러나 test 중에는 dropout을 적용하지 않아야 한다.
만약 test 중에 dropout을 적용하면 매번 무작위로 일부 neuron이 비활성화되기 때문에 예측이 불안정해진다.

Part 2

(a)

Stack	Buffer	New dependency	Transition
[ROOT]	[I, attended, lectures, in, the, NLP, class]		Initial Configuration
[ROOT, I]	[attended, lectures, in, the, NLP, class]		SHIFT
[ROOT, I, attended]	[lectures, in, the, NLP, class]		SHIFT
[ROOT, attended]	[lectures, in, the, NLP, class]	attended→I	LEFT-ARC

[ROOT, attended, lectures] | [in, the, NLP, class] | | SHIFT
[ROOT, attended] | [in, the, NLP, class] | attended -> lectures | LEFT-ARC
[ROOT, attended, in] | [the, NLP, class] | | SHIFT
[ROOT, attended, in, the] | [NLP, class] | | SHIFT
[ROOT, attended, in, the, NLP] | [class] | | SHIFT
[ROOT, attended, in, the, NLP, class] | [] | | SHIFT
[ROOT, attended, in, the, class] | [] | class -> NLP | LEFT-ARC
[ROOT, attended, in, class] | [] | class -> the | LEFT-ARC
[ROOT, attended, class] | [] | class -> in | LEFT-ARC
[ROOT, attended] | [] | attended -> class | RIGHT-ARC
[ROOT] | [] | ROOT -> attended | RIGHT-ARC

(b) 2n
1. push n words to the stack with SHIFT
2. remove n words with the LEFT-ARC or RIGHT-ARC