yliu3233@gatech.edu
Yi Chieh Liu

1. *Solution.*

   (a) Same.
   $$P(store|computer) = P(monitor|computer) = 0.4$$

   (b) d=0.5
   $$P(W_i|W_{i-1}) = \frac{max(C(w_{i-1}, w_i) - d, 0)}{C(w_{i-1})} + \lambda(w_{i-1})P_{continue}$$
   For the bigram computer store
   $$\frac{max(C(w_{i-1}, w_i) - d, 0)}{C(w_{i-1})} = \frac{3.5}{10} = 0.35$$
   $$\lambda(w_{i-1}) = \frac{0.5}{10} \times 3 = 0.15$$
   $$P_{continue} = \frac{3}{9}$$
   $$P_{KN}(store|computer) = 0.40$$
   For the bigram computer monitor
   $$\frac{max(C(w_{i-1}, w_i) - d, 0)}{C(w_{i-1})} = \frac{3.5}{10} = 0.35$$
   $$\lambda(w_{i-1}) = \frac{0.5}{10} \times 3 = 0.15$$
   $$P_{continue} = \frac{2}{9}$$
   $$P_{KN}(monitor|computer) = 0.383$$
   Yes, since the word "store" is more likely to appear in all the text.

   (c) d=0.1
   $$P_{KN}(store|computer) = 0.40$$
   $$P_{KN}(monitor|computer) = 0.3967$$
   The probability of two bigrams become similar since we decrease the d. However, the probability of computer store still higher than computer monitor.

■

2. *Solution.*

(a) The following table are the TF, IDF and the TF-IDF weights.

TF(t,d)	Doc1	Doc2	Doc3
car	1.447158031	0.6989700043	1.397940009
insurance	0.6020599913	1.278753601	0
auto	0	1.531478917	1.477121255
best	1.176091259	0	1.255272505

	IDF(t)
car	0.2218487496
insurance	0.5228787453
auto	0.3010299957
best	0.09691001301

TF(t,d) x IDF(t)	Doc1	Doc2	Doc3
car	0.3210501998	0.1550656215	0.310131243
insurance	0.3148043728	0.6686330784	0
auto	0	0.4610210918	0.4446578049
best	0.1139750192	0	0.1216484748

(b) The cosine similarity of Doc1, Doc2, Doc3 is as follows:

$$Cosinesimilarity(Doc1, Doc2) = 0.6786147103$$

$$Cosinesimilarity(Doc2, Doc3) = 0.5509130366$$

$$Cosinesimilarity(Doc1, Doc3) = 0.0.4401322123$$

■

yliu3233@gatech.edu
Yi Chieh Liu

3. *Solution.*

(a) The 3(a) and 3(b) is as follows:

(a)

$$CE(y, \hat{y}) = -\sum_i \log\left(\frac{\exp(u_{w_o}^T \cdot v_{w_i})}{\sum_{w \in Vocab} \exp(u_w^T \cdot v_{w_i})}\right)$$

$$= -\sum_i \log\left(P(O = w_o \mid I = w_i)\right)$$

(b)

$$\frac{\partial CE(y, \hat{y})}{\partial v_{w_i}} = \sum_{w' \in Vocab}\left(\frac{u_{w'}^T \exp(u_{w'}^T \cdot v_{w_i})}{\sum_{w \in Vocab} \exp(u_w^T \cdot v_{w_i})}\right) - u_{w_o}^T$$

$$= \sum_{w' = Vocab}\left(u_{w'}^T \cdot \hat{y}_{w'}\right) - U \cdot y$$

$$= U \cdot \hat{y} - U y$$

(b) The 3(c) is as follows:

(c)

$$\frac{\partial CE(y, \hat{y})}{\partial u_{w_o}} = \sum_{w \in Vocab} \frac{v_{w_i} \cdot \exp(u_w^T \cdot v_{w_i})}{\sum_{w \in Vocab} \exp(u_w^T \cdot v_{w_i})} - v_{w_i}$$

for $w_o = O$ $\quad \frac{\partial CE}{\partial u_{w_o}} = v_{w_I} \cdot \hat{y} - v_{w_I}$

$w_o \neq O$ $\quad \frac{\partial CE}{\partial u_{w_o}} = -v_{w_I}$

(c) The size of our word vocabulary means that our skip-gram model has a tremendous number of weights, all of which would be updated slightly by every samples. With negative sampling, we are able to randomly select k words (k is smaller than the number of sample). Hence, the loss will only be propagated back for them and therefore only the weights corresponding to them will be updated. The technique could overcome the gradient vanishing and make the loss decrease more critically.

■

4. *Solution.*

(a) The code has been uploaded to the Gradescope.

(b) Yes, my character-level generation begin with F and word-level generation begin with In. The reason of the tendency is that F and In often appear in the first character/word in the training data, which make the range of their probabilities is wide and more likely to be selected.

(c) Both of the results are not satisfying. The perplexity are both infinity since there are unseen word in the test data. The model could not handle the unknown issue and we should implement smoothing methods.

(d) I exhaustively search the n, k, λ, n from 1 to 10, k for 0.0001, 0.001, 0.01, 0.1, 1, 10, λ from increasing to decreasing. The results is as follows

	N	K	Lambda	Perplexity
Character - shakespeare sonnets	7	0.0001	0.33, ...0.33	5.677816019041828
Word - Shakespeare sonnets	2	0.001	[0.2, 0.3, 0.5]	2936.348970275774
Word - val-e	2	0.0001	[0.3, 0.2, 0.5]	706.86972453515

(e) The paragraph generated by the RNN model is as follows:
"Whis in thou sweet his barder thou lives thine old, And say, age ore to be were eyes, rand still' In o"
Compared to the n-gram model, the rnn model with the lowest weight is more fluent and have some grammar concepts. The perplexity is 2.0491368770599365, which is less than the n-gram model.

■