Question 1

Part A

1.

Process:

First, I use **grid search** to find suitable C, I make C values = [0.001, 0.01, 0.1, 1, 10, 100] to try one value for every tenth digit. Then I get the relative accuracy as follows:

C: 0.001 acc: 0.7798

C: 0.01 acc: 0.8298

C: 0.1 acc: 0.8608

C: 1 acc: 0.8892

C: 10 acc: 0.897

C: 100 acc: 0.8928

best C: 10

Then, I find that the best C is between 10 and 100, so I try another set of C values = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100] and get relative accuracy as follows:

C: 10 acc: 0.897

C: 20 acc: 0.8954

C: 30 acc: 0.8958

C: 40 acc: 0.8942

C: 50 acc: 0.8938

C: 60 acc: 0.8938

C: 70 acc: 0.8936

C: 80 acc: 0.8928

C: 90 acc: 0.8928

C: 100 acc: 0.8928

best C: 10

Then, I find that the best C is between 10 and 20, so I try another set of C values = [10, 11, 12, 13, 14, 15, 16, 17, 18, 19] and get relative accuracy as follows:

C: 10 acc: 0.897

C: 11 acc: 0.8966

C: 12 acc: 0.896

C: 13 acc: 0.896

C: 14 acc: 0.896

C: 15 acc: 0.896

C: 16 acc: 0.8964

C: 17 acc: 0.8962

C: 18 acc: 0.8956

C: 19 acc: 0.8954

best C: 10

Answer:

Therefore, it's suitable to use grid search and choose **C values = [0.001, 0.01, 0.1, 1, 10, 100]** to be our range of values.

This range is reasonable because:

①. Easy and fast: It's a very straightforward way for us to initialize C and make adjustments according to the results just like what I have done above.

②. Wide range: There is one value for every tenth digit. Although C is uncertain, this range provide wide enough range for us.

Grid search is appropriate because:

①. Straightforward: We can easily see the results using grid search and make adjustments accordingly.

②. Wide coverage: Grid search ensures that all possible hyperparameter combinations are tried within a given parameter range.

2.

10 is the best C to single digit precision, if we want to make it more precise, we can try [10, 10.1, 10.2, 10.3, 10.4, 10.5, 10.6, 10.7, 10.8, 10.9] and so on.

3.

The accuracy on test set is 0.9055

Part B

1.

I tune vector_size and windows, the ranges of them are vector_size = [100, 200, 300] and windows = [5, 10, 20] and get following results:

Table 1: Question 1 Part B

Vector Size	Window	Best C	Accuracy
100	5	100	0.8418
100	10	1	0.849
100	20	10	0.851
200	5	100	0.8594
200	10	10	0.8594
200	20	100	0.8592
300	5	100	0.8634
300	10	100	0.8674
300	20	100	0.8652

Vector size defines the dimensionality of word vectors and normally it's from 100 to 300 according to experience. **The range of vector_size is reasonable because:**

① Wide range: it's wide enough from 100 to 300, this covers most possibilities so it can help us estimate the ideal value and we can make adjustments accordingly.

② The value spacing is appropriate: We get one number every 100 so we can clearly see the difference every 100, which can help us approach ideal value

The window size defines the extent to which we pay attention to context and normally it's from 5 to 20. **The range of windows is suitable because:**

① Wide range: It's wide enough for us to see most possibilities and make adjustments accordingly.

② Avoided some extreme values: We didn't choose values which too big or too small, which avoid redundant computation.

Grid search method is used and it's appropriate because:

① Various combinations and possibilities: Grid search makes different combinations between vector size and windows to check whether the combination is the best, this can help people to clearly see more possibilities.

② Provide tangible numbers: Grid search can provide relative performances for different combinations, which can help us make further adjustments on parameters.

2.

The best performing values are: Vector Size = 300 and Window = 10

3.

The accuracy on test set is 0.8714. This result is smaller than the result in part A, which means the TF-IDF method is more efficient than the Word2vec method this is probably because:

① TF-IDF focuses on capturing important words to make predictions and Word2vec just focuses on context words, which means Word2vec's focus is narrower than TF-IDF's. This may result in Word2vec does not pay attention to emotions beyond the context of words

② Word2vec may not learn enough of the fine-grained semantic relationships of words while TF-IDF doesn't need much learning than Word2vec.

Question 2

TF-IDF

1.

I have tried K = 2, K = 3 and K = 4.

When K = 2, we just assume that people's reviews are either positive or negative.

When K = 3, we assume that people's reviews have three sentiments: positive, negative, and neutral.

When K = 4, we assume that people's reviews have four sentiments: positive, neutral to positive, neutral to negative, and negative.

These K values are reasonable because K means how many clusters or sentiments there are for movie reviews, all settings of K make sense in classifying movies in reality (Strong interpretability). Besides, all movie reviews can be classified into at least one cluster. Below is the result:

Table 2: Result for Different K (TF-IDF)

K	max_iter	seed	Result
2	5	40	Iteration Total Distance

			0	48458.14
			1	42324.32
			2	42170.35
			3	42114.6
			4	42096.09
3	5	40	Iteration	Total Distance
			0	48147.92
			1	42281.22
			2	42102.78
			3	42001.77
			4	41974.19
4	5	40	Iteration	Total Distance
			0	47693.44
			1	42229.53
			2	42085.06
			3	42012.86
			4	41924.81

2.

We choose K = 2 and seed = 40, 100 or 200 to see the difference, the result is as follows:

Table 3: Result for Different Seed (TF-IDF)

K	max_iter	seed	Result	
2	5	40	Iteration	Total Distance
			0	48458.14
			1	42324.32
			2	42170.35
			3	42114.6
			4	42096.09
2	5	100	Iteration	Total Distance
			0	48847.22
			1	42382.56
			2	42219.82
			3	42123.86
			4	42100.13
2	5	200	Iteration	Total Distance
			0	48614.98
			1	42209.5
			2	42111.4
			3	42096.07
			4	42092.89

From Table 3, we can see that the total distance doesn't change much after 5 iterations with different seeds, which means that the seed doesn't have an obvious influence on the results. This is probably because the value of seed doesn't change much.

3.

From Table 2, we can see that K = 4 is the best one after 5 iterations. And as the K value increases, the total distance of the cluster becomes closer and closer after 5 iterations, which means that the classification is getting better and better and the improvement of K value can better fit the model.

Besides, it does not mean that the larger the K value is after each iteration, the better the cluster effect will be. This is probably because there exist suitable K values for different iterations (a bigger K doesn't mean a better cluster).

Word2vec

1.

I have tried K = 5, K = 6 and K = 7.

When K = 5, we assume that people's reviews have 5 sentiments: happy, sad, angry, fear and surprise.

When K = 6, we assume that people's reviews have 6 sentiments: happy, sad, angry, fear, surprise and hate.

When K = 7, we assume that people's reviews have 7 sentiments: happy, sad, angry, fear, surprise, hate and pity.

These K values are reasonable because they are all the normal sentiments in real life, the results are more representative. Besides, the setting of K value covers most of people's sentiments in real life.

Table 4: Result for Different K (Word2vec)

K	max_iter	seed	Result	
5	5	40	Iteration	Total Distance
			0	562.43
			1	433.22
			2	416.46
			3	411.62
			4	409.59

6	5	40	Iteration	Total Distance
			0	585.3
			1	420.68
			2	409.5
			3	405.9
			4	404.2
7	5	40	Iteration	Total Distance
			0	597.42
			1	409.54
			2	401.67
			3	398.37
			4	396.74

2.

We choose K = 5 and seed = 40 or 1000 or 10000, below is the result:

Table 5: Result for Different Seeds (Word2vec)

K	max_iter	seed	Result	
5	5	40	Iteration	Total Distance
			0	562.43
			1	433.22
			2	416.46
			3	411.62
			4	409.59
5	5	1000	Iteration	Total Distance
			0	632.18
			1	433.16
			2	418.71
			3	414.09
			4	411.93
5	5	10000	Iteration	Total Distance
			0	692.91
			1	423.16
			2	415.05
			3	412.15
			4	410.65

From Table 5, we can see that seed doesn't influence much on the result after 5 iterations. This means that selections of initial centroid may influence little to the result.

3.

From Table 4, we can see that K = 7 is the best one after 5 iterations, but it's not always the best one after each iteration (it's the worst one after the first iteration). This illustrates that a bigger K can fit better for the model, but there may be a suitable K for each iteration.

Question 3

Different try will give different results, one is as below:

Validation loss: 2.3822

Argmax: John Barrin

Random:

Sopefta Kosta

Bahle Toxenal

Stent Pryriss

Duum Makhefolason

Jak Amumnyar

Beell Terceler

BetcFo Taises

Jak Tofalle

Indn Fr

Bvond Ronlakas

Below is the visualization that we use inside cosine and K = 5, max_iter = 5, seed = 10000. It takes me too much time to get this result and it will take longer if I use cosine_distance that I write because computer problem. Hope this can help.