Institute of Business Administration
Introduction to Text Analytics
Assignment 03 – Assessment

Name: ZUHA AQIB **ID: 26106**

Report each experiment's detail and scores for k = 5, 9, and 13. You are required to perform ten experiments for each 'k' (number of clusters). Please set random seed value to your ERP ID for each K-Means clustering experiment.

PLEASE NOTE

This table is incomplete due to size and the full table is attached as an excel sheet with this submission.

*The first two entries in the table are provided for reference only. Hence, the scores do not interpret anything and have been entered randomly. Replace these entries while submitting.

k (Number of clusters)	Vectorizer Type and Details	vector_size	window	Epochs Count	CBoW/Skipgram OR DM/DBoW	Silhouette Score	WSS Score
5	Word2Vec	200	3	50	CBOW	0.20428638	5.18647337
	Word2Vec	200	10	50	CBOW	0.26089746	4.396201611
	Word2Vec	200	8	50	CBOW	0.2590102	4.453023434
	Word2Vec	200	10	50	CBOW	0.21531789	5.157343864
	Word2Vec	200	10	5	CBOW	0.3297802	0.029046454
	Word2Vec	300	3	50	Skipgram	0.10897815	2.091296673
	Word2Vec	400	3	50	Skipgram	0.102759175	1.874755621
	Word2Vec	425	3	50	Skipgram	0.11921752	1.963598371
	Word2Vec	400	15	50	Skipgram	0.10061162	1.196420431
	Word2Vec	400	15	25	Skipgram	0.15037157	0.637758613
	Word2Vec	400	15	5	Skipgram	0.3181979	0.072357133
	Doc2Vec	300	3	50	DBOW	0.23340887	1.556313276
	Doc2Vec	400	3	50	DBOW	0.2322252	1.213774204
	Doc2Vec	500	3	50	DBOW	0.24864852	0.980379701
	Doc2Vec	600	3	50	DBOW	0.24542189	0.814524651
	Doc2Vec	700	3	50	DBOW	0.24686135	0.717621684
	Doc2Vec	800	3	50	DBOW	0.24619788	0.624525785
	Doc2Vec	1000	3	50	DBOW	0.25864682	0.522559702
9	Word2Vec	250	3	50	CBOW	0.15616488	3.46835947
	Word2Vec	200	10	50	CBOW	0.20203397	3.459095955
	Word2Vec	200	8	50	CBOW	0.1600107	4.450608253
	Word2Vec	200	10	50	CBOW	0.17876591	4.168611526
	Word2Vec	200	10	5	CBOW	0.16110314	0.0217437
	Word2Vec	300	3	50	Skipgram	0.09239725	1.636084676
	Word2Vec	400	3	50	Skipgram	0.093137525	1.729730368
	Word2Vec	425	3	50	Skipgram	0.09071183	1.821123481
	Word2Vec	400	15	50	Skipgram	0.10140439	1.035103917
	Word2Vec	400	15	25	Skipgram	0.13093129	0.644830048
	Word2Vec	400	15	5	Skipgram	0.22330284	0.046959497

	Doc2Vec	300	3	50	DBOW	0.12699512	1.295376539
	Doc2Vec	400	3	50	DBOW	0.14082325	0.983258665
	Doc2Vec	500	3	50	DBOW	0.14187236	0.790132999
	Doc2Vec	600	3	50	DBOW	0.14582942	0.656357229
	Doc2Vec	700	3	50	DBOW	0.15012994	0.580757856
	Doc2Vec	800	3	50	DBOW	0.13678437	0.484199941
	Doc2Vec	1000	3	50	DBOW	0.15270704	0.407283843
13	Word2Vec	200	3	50	CBOW	0.130651	3.157839298
	Word2Vec	200	10	50	CBOW	0.16116615	3.137774467
	Word2Vec	200	8	50	CBOW	0.1698051	3.298812628
	Word2Vec	200	10	50	CBOW	0.16661488	2.868229628
	Word2Vec	200	10	5	CBOW	0.14250503	0.014597037
	Word2Vec	300	3	50	Skipgram	0.09102542	1.520768881
	Word2Vec	400	3	50	Skipgram	0.09856355	1.436671734
	Word2Vec	425	3	50	Skipgram	0.09491624	1.485663533
	Word2Vec	400	15	50	Skipgram	0.094204195	0.924631417
	Word2Vec	400	15	25	Skipgram	0.117109165	0.560105383
	Word2Vec	400	15	5	Skipgram	0.18925893	0.035685554
	Doc2Vec	300	3	50	DBOW	0.07994233	1.181271911
	Doc2Vec	400	3	50	DBOW	0.0969785	0.899154246
	Doc2Vec	500	3	50	DBOW	0.0936995	0.714732409
	Doc2Vec	600	3	50	DBOW	0.09148358	0.5942083
	Doc2Vec	700	3	50	DBOW	0.09751208	0.512354314
	Doc2Vec	800	3	50	DBOW	0.09121083	0.459784329
	Doc2Vec	1000	3	50	DBOW	0.117727555	0.365352154

Analysis & Interpretation:

- **Identify which embedding technique resulted in the best clustering.**
- **Discuss how different choices of hyperparameters impacted the results.**
- **Compare the performance of word2vec and doc2vec embeddings with those used in previous assignment (Assignment 02)**

So first of all, word2vec and doc2vec is MUCH better than the previous ones as our best results in the last assignment was wss=52, and sil=0.56, but here my first run was sil=0.20 and wss=5.18. Thus just before beginning we can see that this is much better.

Starting off, I applied the same preprocessing of the BEST that I found in the last assignment: unigrams, lemmatization, stopwords removed, tokenized with data cleaned.

Then I ran my code with multiple values for one variable (vectorsize, windowsize, epochs) and rest were kept constant – this ran in a loop to find the BEST variable value before moving to tune the next.
In this I ran almost 5-10 loops for each variable, sometimes additional runs just to find the breakpoint value where it has the highest SIL and lowest WSS. Thus I have only shown the BEST of all 5-10 iterations, making total 30 iterations for each cluster k.

So thus I started with CBOW and SKIPGRAM. I started with base values of each variable and tuned from vectorsize to windowsize to epochs.

1	[5, 'word', 'CBOW', 200, 3, 50, 0.20428638, 5.186473369598389]	k=9
2	[5, 'word', 'CBOW', 200, 10, 50, 0.26089746, 4.3962016105651855]	k=9
3	[5, 'word', 'CBOW', 200, 8, 50, 0.2590102, 4.453023433685303]	no elbow
4	[5, 'word', 'CBOW', 200, 10, 50, 0.21531789, 5.157343864440918]	no elbow
5	[5, 'word', 'CBOW', 200, 10, 5, 0.3297802, 0.029046453535556793]	no elbow
6	[13, 'word', 'Skipgram', 300, 3, 50, 0.09102542, 1.5207688808441162]	no elbow
7	[13, 'word', 'Skipgram', 400, 3, 50, 0.09856355, 1.4366717338562012]	no elbow
8	[13, 'word', 'Skipgram', 425, 3, 50, 0.09491624, 1.4856635332107544]	k=9
9	[13, 'word', 'Skipgram', 400, 15, 50, 0.094204195, 0.9246314167976379]	k=9
10	[13, 'word', 'Skipgram', 400, 15, 25, 0.117109165, 0.5601053833961487]	no elbow
11	[5, 'word', 'Skipgram', 400, 15, 5, 0.3181979, 0.07235713303089142]	k=9

In WORD2VEC we noticed that skipgram was much much better than cbow as it started with a much better WSS but a very bad bad SIL. Skipgram used more vector size – I ran it multiple times to see if it worked better for more than 400 however it decreased SIL and increased WSS. Window size it used more larger however CBOW used lesser, and in both I noticed the smaller the epochs, the better. The values improved drastically.
The best value we found was with CBOW however as it had more SIL and lesser WSS.

Then after we found a decent enough value for both, we moved to DOC2VEC. For DBOW:

12	[5, 'doc', 'DBOW', 300, 3, 50, 0.23340887, 1.5563132762908936]	k=9
13	[5, 'doc', 'DBOW', 400, 3, 50, 0.2322252, 1.2137742042541504]	no elbow
14	[5, 'doc', 'DBOW', 500, 3, 50, 0.24864852, 0.9803797006607056]	no elbow
15	[5, 'doc', 'DBOW', 600, 3, 50, 0.24542189, 0.8145246505737305]	no elbow
16	[5, 'doc', 'DBOW', 700, 3, 50, 0.24686135, 0.7176216840744019]	k=9
17	[5, 'doc', 'DBOW', 800, 3, 50, 0.24619788, 0.624525785446167]	no elbow
18	[5, 'doc', 'DBOW', 1000, 3, 50, 0.25864682, 0.5225597023963928]	no elbow
19	[5, 'doc', 'DBOW', 1500, 3, 50, 0.25769895, 0.3495648205280304]	no elbow
20	[5, 'doc', 'DBOW', 2500, 3, 50, 0.26216635, 0.2140878140926361]	no elbow
21	[5, 'doc', 'DBOW', 4000, 3, 50, 0.2695049, 0.13980528712272644]	k=9
22	[5, 'doc', 'DBOW', 4000, 3, 50, 0.2695049, 0.13980528712272644]	k=9
23	[5, 'doc', 'DBOW', 4000, 3, 50, 0.2695049, 0.13980528712272644]	k=9

Here we saw that DBOW was getting better and better with vector size, I did nearly 50 iterations checking from 100 vector size to 4000 – and still it was just getting better, that's when I had to stop.
But this gave us another thing to analyse – no matter what window size I kept, there was no difference:

22	Doc2Vec	4000	3	50	DBOW	0.2695049	0.139805287
22	Doc2Vec	4000	5	50	DBOW	0.2695049	0.139805287
22	Doc2Vec	4000	7	50	DBOW	0.2695049	0.139805287
22	Doc2Vec	4000	10	50	DBOW	0.2695049	0.139805287
22	Doc2Vec	4000	12	50	DBOW	0.2695049	0.139805287
22	Doc2Vec	4000	15	50	DBOW	0.2695049	0.139805287
22	Doc2Vec	4000	20	50	DBOW	0.2695049	0.139805287

See here that whatever window size from 3-20 was kept, the values did not change.
And this was the same for epochs, but they were VERY LONG and slow. They took a lot of time:

23	Doc2Vec	4000	3	50	DBOW	0.2695049	0.139805287
23	Doc2Vec	4000	3	100	DBOW	0.31483832	3.069660664
23	Doc2Vec	4000	3	150	DBOW	0.30241734	116.249939
23	Doc2Vec	4000	3	200	DBOW	0.08463991	1741.776855
23	Doc2Vec	4000	3	250	DBOW	0.05159686	4313.291992

And the values were very haphazard and not good.
This further strengthened our hypothesis that the more the epochs, the worser

Then came DM:

24	Doc2Vec	2500	3	50	DM	0.31171334	0.13287279
24	Doc2Vec	2750	3	50	DM	0.31764877	0.122057475
24	Doc2Vec	3000	3	50	DM	0.31404993	0.111240186
24	Doc2Vec	3250	3	50	DM	0.3087002	0.100687139
24	Doc2Vec	3500	3	50	DM	0.31351027	0.093864843
24	Doc2Vec	3750	3	50	DM	0.31418225	0.087562226
24	Doc2Vec	4000	3	50	DM	0.31878072	0.084688254

Just after the first grid search we can see that it is WAY better than any of the others – it gave the lowest WSS and highest SIL.
But now that vector size is so high, even after changing all values of window size and epochs, there was no change and only deterioration,

```
[5, 'doc', 'DM', 4000, 3, 50, 0.31878072,
0.08468825370073318]
[5, 'doc', 'DM', 4000, 3, 50, 0.31878072,
0.08468825370073318]
```

k=9

k=9

k	Vectorizer Type	vector_size	window	Epochs Count	Vectorizer Name	Silhouette Score	WSS Score
5	Doc2Vec	4000	3	50	DM	0.318781	0.084688
5	Doc2Vec	4000	5	50	DM	0.40597	0.178194
5	Doc2Vec	4000	7	50	DM	0.435287	0.337264
5	Doc2Vec	4000	10	50	DM	0.494364	0.594451
5	Doc2Vec	4000	12	50	DM	0.523626	0.63316
5	Doc2Vec	4000	15	50	DM	0.49827	0.798213
5	Doc2Vec	4000	20	50	DM	0.514435	0.811576
9	Doc2Vec	4000	3	50	DM	0.200868	0.060477
9	Doc2Vec	4000	5	50	DM	0.321749	0.087757
9	Doc2Vec	4000	7	50	DM	0.388037	0.140837
9	Doc2Vec	4000	10	50	DM	0.445877	0.231961
9	Doc2Vec	4000	12	50	DM	0.452999	0.278377
9	Doc2Vec	4000	15	50	DM	0.443924	0.296356
9	Doc2Vec	4000	20	50	DM	0.469326	0.294408
13	Doc2Vec	4000	3	50	DM	0.128698	0.054968
13	Doc2Vec	4000	5	50	DM	0.235948	0.069309
13	Doc2Vec	4000	7	50	DM	0.338308	0.085057
13	Doc2Vec	4000	10	50	DM	0.387119	0.129843
13	Doc2Vec	4000	12	50	DM	0.389599	0.147211
13	Doc2Vec	4000	15	50	DM	0.438907	0.155927
13	Doc2Vec	4000	20	50	DM	0.450675	0.16765

THUS

We come to our conclusion that

The best in both WSS and SIL was CBOW. It was indeed the fastest and less resource consuming whereas DM and DBOW gave nearly same result however took a huge vector size and much more computation time.

BEST IN EVERY CASE	k, vector_size, window_size, epoch, sil, wss	elbow curve
1	[5, 'word', 'CBOW', 200, 3, 50, 0.20428638, 5.186473369598389]	k=9
2	[5, 'word', 'CBOW', 200, 10, 50, 0.26089746, 4.3962016105651855]	k=9
3	[5, 'word', 'CBOW', 200, 8, 50, 0.2590102, 4.453023433685303]	no elbow
4	[5, 'word', 'CBOW', 200, 10, 50, 0.21531789, 5.157343864440918]	no elbow
5	[5, 'word', 'CBOW', 200, 10, 5, 0.3297802, 0.029046453535556793]	no elbow
6	[13, 'word', 'Skipgram', 300, 3, 50, 0.09102542, 1.5207688808441162]	no elbow
7	[13, 'word', 'Skipgram', 400, 3, 50, 0.09856355, 1.4366717338562012]	no elbow
8	[13, 'word', 'Skipgram', 425, 3, 50, 0.09491624, 1.4856635332107544]	k=9
9	[13, 'word', 'Skipgram', 400, 15, 50, 0.094204195, 0.9246314167976379]	k=9
10	[13, 'word', 'Skipgram', 400, 15, 25, 0.117109165, 0.5601053833961487]	no elbow
11	[5, 'word', 'Skipgram', 400, 15, 5, 0.3181979, 0.07235713303089142]	k=9
12	[5, 'doc', 'DBOW', 300, 3, 50, 0.23340887, 1.5563132762908936]	k=9
13	[5, 'doc', 'DBOW', 400, 3, 50, 0.2322252, 1.2137742042541504]	no elbow
14	[5, 'doc', 'DBOW', 500, 3, 50, 0.24864852, 0.9803797006607056]	no elbow
15	[5, 'doc', 'DBOW', 600, 3, 50, 0.24542189, 0.8145246505737305]	no elbow
16	[5, 'doc', 'DBOW', 700, 3, 50, 0.24686135, 0.7176216840744019]	k=9
17	[5, 'doc', 'DBOW', 800, 3, 50, 0.24619788, 0.624525785446167]	no elbow
18	[5, 'doc', 'DBOW', 1000, 3, 50, 0.25864682, 0.5225597023963928]	no elbow
19	[5, 'doc', 'DBOW', 1500, 3, 50, 0.25769895, 0.3495648205280304]	no elbow
20	[5, 'doc', 'DBOW', 2500, 3, 50, 0.26216635, 0.2140878140926361]	no elbow
21	[5, 'doc', 'DBOW', 4000, 3, 50, 0.2695049, 0.13980528712272644]	k=9
22	[5, 'doc', 'DBOW', 4000, 3, 50, 0.2695049, 0.13980528712272644]	k=9
23	[5, 'doc', 'DBOW', 4000, 3, 50, 0.2695049, 0.13980528712272644]	k=9
24	[5, 'doc', 'DM', 4000, 3, 50, 0.31878072, 0.08468825370073318]	k=9
25	[5, 'doc', 'DM', 4000, 3, 50, 0.31878072, 0.08468825370073318]	k=9

THE YELLOW HIGHLIGHTED IS BEST CASE FOR EACH VECTORIZATION TECHNIQUE