Algorithms Implemented

Basic requirement

The following two algorithms satisfy the basic requirement of the assignment. These were run using (1) TF-IDF features and (2) Bag-of-words (BOW) features.

1. *K-means*
2. *K-means++*

Novel algorithm

The following additional algorithm was developed to get richer and more informative features.

Word-cluster based K-means algorithm (WCK-means)

The main intuition behind this approach is that many words are very similar to each other. But using a representation like BOW or TF-IDF does not take this similarity into account thus leading to a more sparse representation of documents.

The idea now is to identify clusters of similar words. Then use these word-clusters as features for the documents.

Following this intuition, I follow these steps:

1. A simple representation of each word in the vocabulary can be found by using the document-term matrix shown in figure 1. Each row in this matrix is the representation of the corresponding word.

2. Next, I apply the K-means algorithm to find word clusters using the features described in the previous step. I thus find k word-clusters $C = \{c_i\}_{i=1}^{k}$.

3. Each feature i for a document d_j is now a word-cluster c_i. The value of this feature d_j^i is the frequency of c_i in the document. This value is computed as the maximum of the frequencies of all words in the cluster c_i present in d_j. Formally,

$$d_j^i = max(\{|w_l|\}_{l=1}^{|c_i|}) \forall w_l \in c_i$$

   Here, $|w_l|$ = frequency of word w_l in the document d_j.

   This can be seen as the *max-pooling* operation performed across all words in a cluster. The most representative word in a cluster will be the one with the highest frequency and would be representative of the semantic score of that cluster.

4. Now a matrix (X) similar to the one in figure 1 can be formed but this time with word-clusters instead of words. Each column of this matrix will be the bag-of-clusters representation of the corresponding document. This column is augmented further by

	doc-1	doc-2	doc-3	...doc-j...	doc-n
word-1	freq. of word-1 in doc-1	...	...	...	...
word-2	...	...	...	...	...
word-3	...	...	...	...	...
...	...	...	...	...	...
word-i	...	...	...	freq. of word-i in doc-j	...
...	...	...	...	...	...
word-m	...	...	...	...	...

Figure 1: Term document matrix

Model	Training accuracy	Validation accuracy	Online efficiency	Offline efficiency
K-means+BOW	8.42%	8.46%	32.63 s	1.6×10^{-5} s
K-means+TF-IDF	11.61%	10.84%	0.9 s	2.4×10^{-6} s
K-means(++)+BOW	8.44%	8.46%	**0.18 s**	8.4×10^{-7} s
K-means(++)+TF-IDF	8.39%	8.37%	0.43 s	9.9×10^{-7} s
WCK-means	**18.51%**	**18.24%**	126.06 s	5.7×10^{-6} s
WCK-means(++)	13.31%	13.44%	92.8 s	9.3×10^{-7} s

Table 1: Results

the IDF score of each cluster which can be easily computed from X. This gives us a TF-IDF equivalent representation of each document in terms of word-clusters.

5. The K-means algorithm is now run on these word-cluster based features.

Results and Analysis

The results are shown in table 1. Hyperparameters are tuned for all reported algorithms and are reported in table 2. We see that word-cluster based algorithms give the best performance

Model	vocab. size	# word-clusters
K-means+BOW	2048	-
K-means+TF-IDF	256	-
K-means(++)+BOW	128	-
K-means(++)+TF-IDF	64	-
WCK-means	-	64
WCK-means(++)	-	32

Table 2: Hyperparameter setting

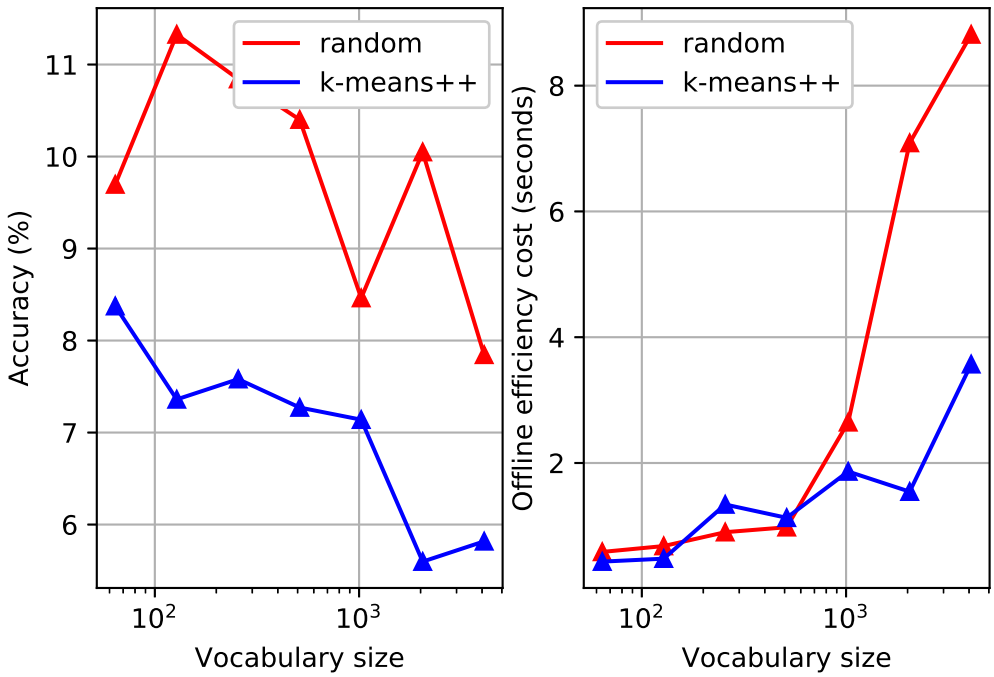

Figure 2: (left) effect of vocabulary size on accuracy (right) effect of vocabulary on training time (x-axis is on a log-scale)

in terms of prediction accuracy. This shows that the word clusters solve the problem of sparsity to a large extent while keeping the semantic information intact.

We can also observe that the K-means++ algorithm doesn't perform very well in terms of the prediction accuracy. This comes as a surprise and more investigations need to be made as to why. One possible explanation can be that the K-means++ algorithm gets stuck in the local-minima quickly given the fact the features are still not quite dense. But this is just a speculative argument.

One thing to note is that K-means++ gives the best performance in terms of Online efficiency. Thus it converges very fast. On the other hand, word-cluster based algorithms take a lot more time to run because of the two clustering steps.

Figure 2 shows the effect of vocabulary size on the accuracy and offline efficiency cost of k-means (random initialization) and k-means++. The trend shows that almost for all vocabulary sizes random initialization does better than k-means++. Also note that the offline efficiency cost for k-means++ is very low establishing the fact that it converges very fast.

I also compare the performance of the word-cluster based approach with vanilla K-means in terms of performance variation with feature size. This is shown in figure 3. Note that the features for vanilla K-means are the individual words and that for the WCK-means approach

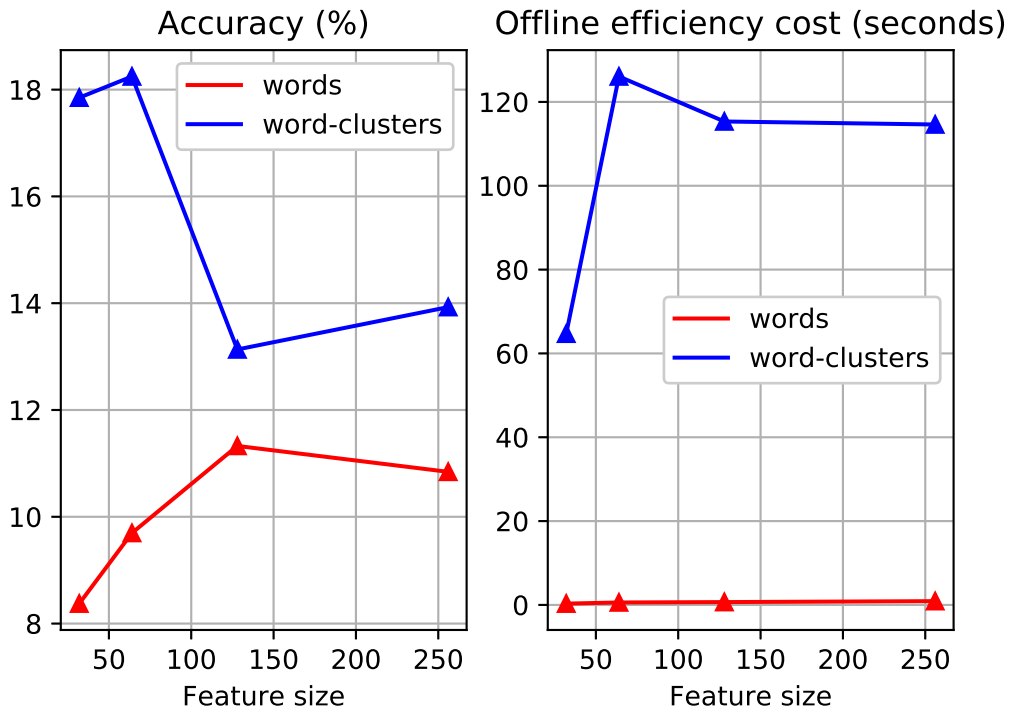

Figure 3: (left) effect of feature-vector size on accuracy (right) effect of feature-vector size on training time

are the word-clusters. In terms of accuracy, the word-cluster based approach does better than word-based K-means for all feature sizes. Another important thing to note is that the offline efficiency cost of the WCK-means approach is extremely high when compared to the word-based K-means even for a feature size < 50. This is one of the drawbacks of the word-cluster based approach.