Deep Learning Dimensionality Reduction for Text

1. Data Preprocessing for Graph Autoencoder:

1.1. Convert a collection of raw documents to a matrix of ***tf-idf*** features:

tf-idf (t, d, D) = tf (t, d) × idf (t, D) *tf-idf (t, d, D) =tf (t, d) ×idf (t, D)*

t: *defines the terms;*

d: *defines each document;*

D: *defines the collection of documents.*

Formula for ***idf*** *(Inverse Document Frequency):*

$$idf(t, D) = log \frac{|D|}{1 + |\{d \in D : t \in d\}|}$$

D: inferring to a document space.

1.2. Scale input vectors individually to unit norm by compute ***Euclidean norm*** (***ℓ²-norm***) of the tf-idf matrix for each feature:

$$\| x \|_2 = \sqrt{\left(\sum_{i=1}^{n} \boldsymbol{x}_i{}^2\right)} = \sqrt{\boldsymbol{x}_1{}^2 + \boldsymbol{x}_2{}^2 + \cdots + \boldsymbol{x}_i{}^2}$$

Table 1: *data samples*

	F_1	F_2	F_3	F_4
$Sample_1$	-1	0	-2	1
$Sample_2$	0	1	2	1
$Sample_3$	1	0	2	2

Here we have the unit norm of $\boldsymbol{F_1}$ elements:

$$z = F1 = \| x \|_2 = \sqrt{(-1)^2 + (0)^2 + (1)^2} = \sqrt{1+0+1} = \sqrt{2} = 1.4$$

$$x = \frac{x}{z} = \frac{-1}{1.4} = -0.71$$

Table 2 : *feature normalized*

	F_1	F_2	F_3	F_4
$Sample_1$	-0.70710678	0.	-0.57735027	0.40824829
$Sample_2$	0.	1.	0.57735027	0.40824829
$Sample_3$	0.70710678	0.	0.57735027	0.81649658

It normalized each column sample. Unit norm means the squared elements sum for each feature is **1**.

Here we have sum of $\boldsymbol{F_1}$ squared elements :

$$(-0.71)^2 + (0.)^2 + (0.71)^2 = 0.5 + 0. + 0.5 = 1.0$$

Table 3: sum of squared elements

	F_1	F_2	F_3	F_4
$Sample_1$	0.5	0.	0.33333333	0.16666667
$Sample_2$	0.	1.	0.33333333	0.16666667
$Sample_3$	0.5	0.	0.33333333	0.16666667
Sum	1.	1.	1.	1.

1.3. Compute pairwise distances between observations in n-dimensional space:

1.3.1. Computes the ***Cosine Distance*** between vectors ***u*** and ***v***:

$$Cos\ (\theta)\ = 1 - \frac{u\,.\,v}{\|\ u\ \|_2\ \ \|\ v\ \|_2}$$

```
Cos (θ) = [1.13608276, 1.40824829, 0.55555556]
```

1.3.2. Convert the vector to an ***n*** by ***n*** distance matrix.

$$v\left[\binom{1}{2} - \binom{n-i}{2} + (j - i - 1)\right]$$

Table 4: Cosine Similarity Result

	$Sample_1$	$Sample_2$	$Sample_3$
$Sample_1$	0.	1.13608276	1.40824829
$Sample_2$	1.13608276	0.	0.55555556
$Sample_3$	1.40824829	0.55555556	0.

1.3.3. Compute the ***Correlation Distance*** between vectors ***u*** and ***v***:

$$Cor\ (\theta) =\ 1 - \frac{(u - \bar{u})\,.\,(v - \bar{v})}{\|\ (u - \bar{u})\ \|_2\ \|\ (v - \bar{v})\ \|_2}$$

```
Cor (θ) = [0.58385853, 1.07028092, 1.8331475]
```

1.3.4. 1.3.2. Convert the vector to an ***n*** by ***n*** distance matrix, see 1.3.2.

Table 5: Correlation Similarity Result

	$Sample_1$	$Sample_2$	$Sample_3$
$Sample_1$	0.	0.58385853	1.07028092
$Sample_2$	0.58385853	0.	1.8331475
$Sample_3$	1.07028092	1.8331475	0.

2. Graph Auto-Encoder:

2.1. Define layers:

Use the geometric progression to define the hidden layers:

$$mhL = k^1 , k^2 , k^3 , \ \ldots \ ,k^n \qquad , (if \ \ k^n < size \ of \ input)$$

$$nhl \ = z(\sqrt{lenght(mhL)})$$

$$hl = mhl[-nhl]$$

k: *embedded code*

z: *integer values*

mhL: *maximum hidden layers*

nhl: *number of hidden layers proposed*

hl: *hidden layers proposed*

E.g. here we have hidden layers for 150 samples for 2D goal embedded:

$$mhL = 2^2 , 2^3 , \ \ldots \ ,k^n \qquad , (if \ \ 2^n < 150)$$

$$mhL = 2^2 , 2^3 , 2^4 , 2^5 , 2^6$$

$$nhl \ = z(\sqrt{mhL}\,) = 2$$

$$hl = mhl[-nhl] = mhl[-2]$$

$$hl = [2^5 , 2^6]$$

Graph Autoencoders for 150 samples:

$$150 — 128 — 64 — 2 — 64 — 128 — 150$$

2.2. Optimizer:

2.2.1. Optimize with the Adam algorithm.

$$m_0 := 0ext(Initializeinitial1stmomentvector)$$

$$v_0 := 0ext(Initializeinitial2ndmomentvector)$$

$$t := 0ext(Initializetimestep)$$

$$t := t + 1$$

$$lr_t := extlearning\_rate * \sqrt{1 - beta_2^t}/(1 - beta_1^t)$$

$$m_t := beta_1 * m_{t-1} + (1 - beta_1) * g$$

$$v_t := beta_2 * v_{t-1} + (1 - beta_2) * g * g$$

$$variable := variable - lr_t * m_t/(\sqrt{v_t} + \epsilon)$$

The Adam algorithm parameters:

Beta 1: The exponential decay rate for the first moment estimates:

β1= 0.9

Beta 2: The exponential decay rate for the second moment estimates.

β2 = 0.999

Learning rate: The learning rate is used which is indifferent to the error gradient:

Learning rate = 0.001

Epsilon: we set the epsilon value to the default value.

Epsilon = 1e-08

2.2.2. Compute gradients of loss to return a list of (gradient, variable) pairs where "gradient" is the gradient for "variable".

2.2.3. Apply gradients to gradients of loss to return an Operation that applies gradients.

2.3. Lost function:

$$|| \, Input - Output \, || = \sum_i (Input_i + Output_i)$$

2.4. Layers:

2.4.1. Activation function:

Use sigmoid function:

$$f(x) = \frac{1}{1 + e^{-(x)}}$$

2.4.2. Weight Initialization:

Consider a *L* layer neural network, which has *L-1* hidden layers and 1 input and output layer each.

2.4.2.1. Weights:

$$W^{[l]} = \frac{1}{(size \ of \ layer \ L)^{1/2}}$$

2.4.2.2. Biases:

$$b^{[l]} = \frac{1}{(size\ of\ layer\ L - 1)^{\ 1/2}}$$

2.5. Normalization layers:

Normalizes an input layer by mean and variance, and applies a scale (gamma γ) and an offset (beta β):

$$\frac{\gamma\ (x - \mu)}{\sigma} + \beta$$

γ initializer: *0.1*

β initializer: *1.0*

2.6. Build deep autoencoder:

Our function minimize the value by following the formula:

Table 6: Model Architecture

Build deep autoencoder and define all layers and span of variables.
▪ *Input is normalized matrix similarity (n×n).* ▪ *S = size of input* ▪ *HL = number of hidden layers* ▪ *L = number of layers* ▪ $\Finv = S^{1/ HL}$

***For** epoch = 1 to* $\Finv$
    ***For** i = 1 to S*
        Train the autoencoder with L number of data along with backpropagation strategy.
        Obtain the cost value.
End

Obtain the code or embedded data after optimization.
Run k-means algorithm on embedded data.

3. Experimental Evaluation:

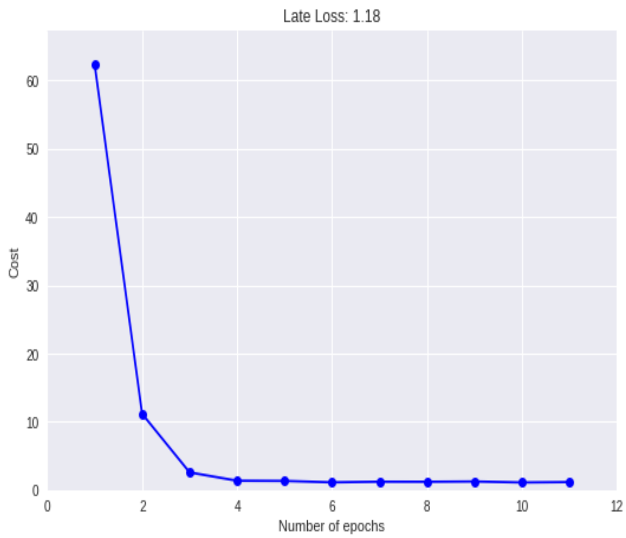
3.1. We experimentally test the models on the 6 groups of the 20 newsgroups datasets which already have been used by other. For each group, we selected 200 documents randomly. It means for our 6 groups we have 1,200 documents. After converting datasets to vector by TF-IDF, we compute normalized correlation on that.

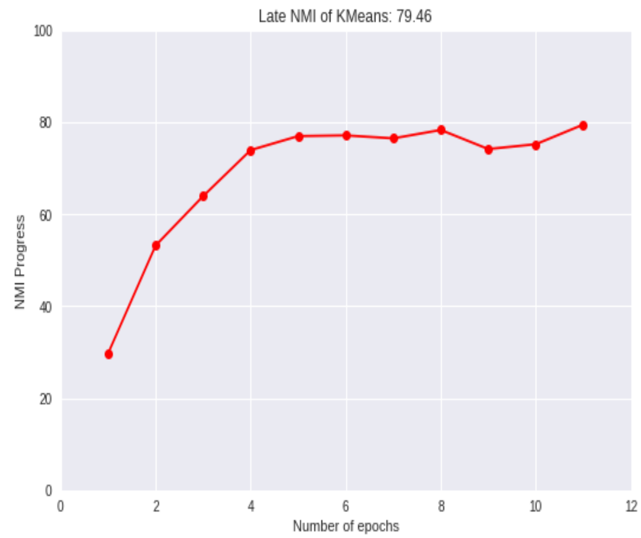
Table 7: Define layers

Dataset	Nodes
6NG	1200 – 1024 – 512 – 256 – 2

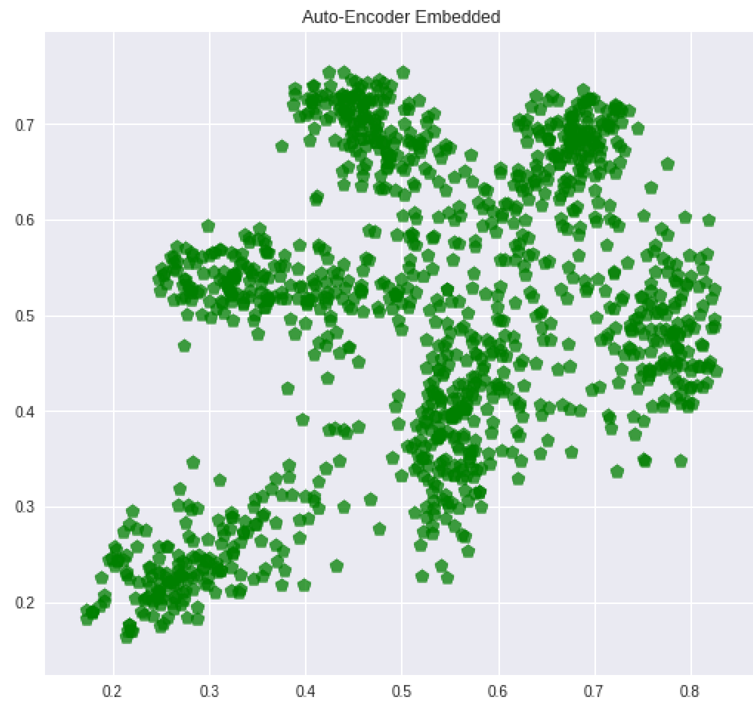
3.2. Results:

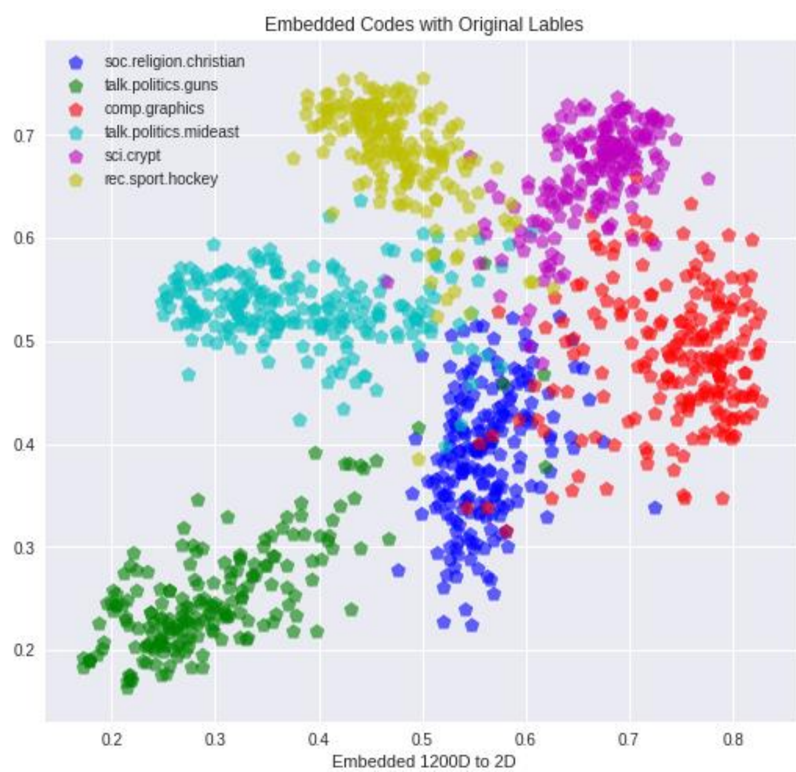
3.2.1. Progresses of the loss function and KMeans algorithm on the embedded code after each epoch:



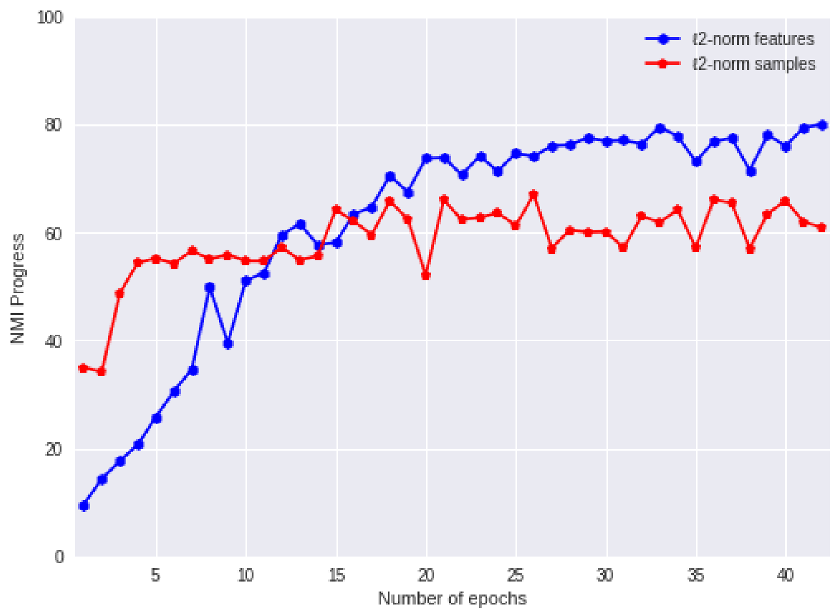


3.2.2. Final Embedded codes of Autoencoder (AE):

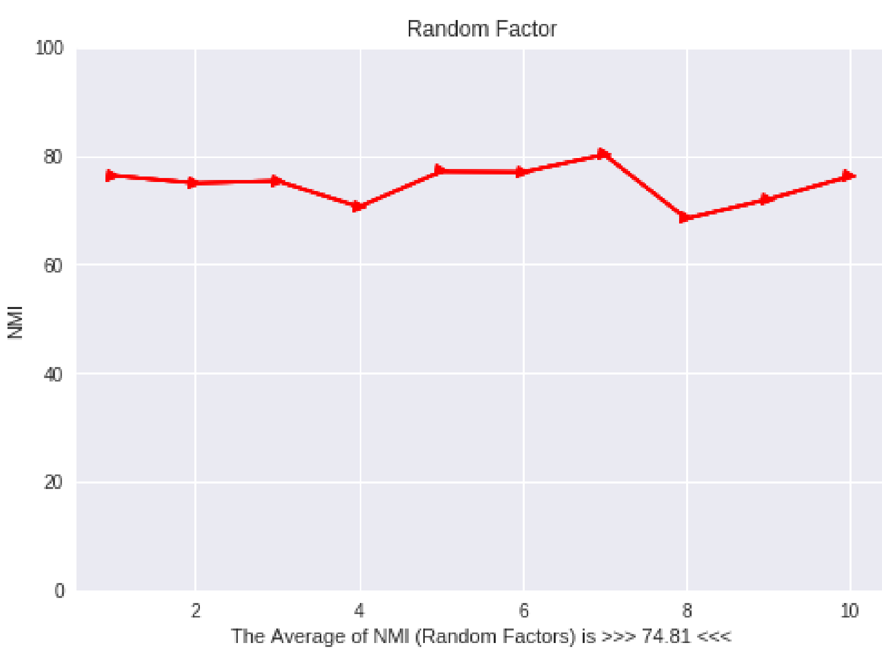




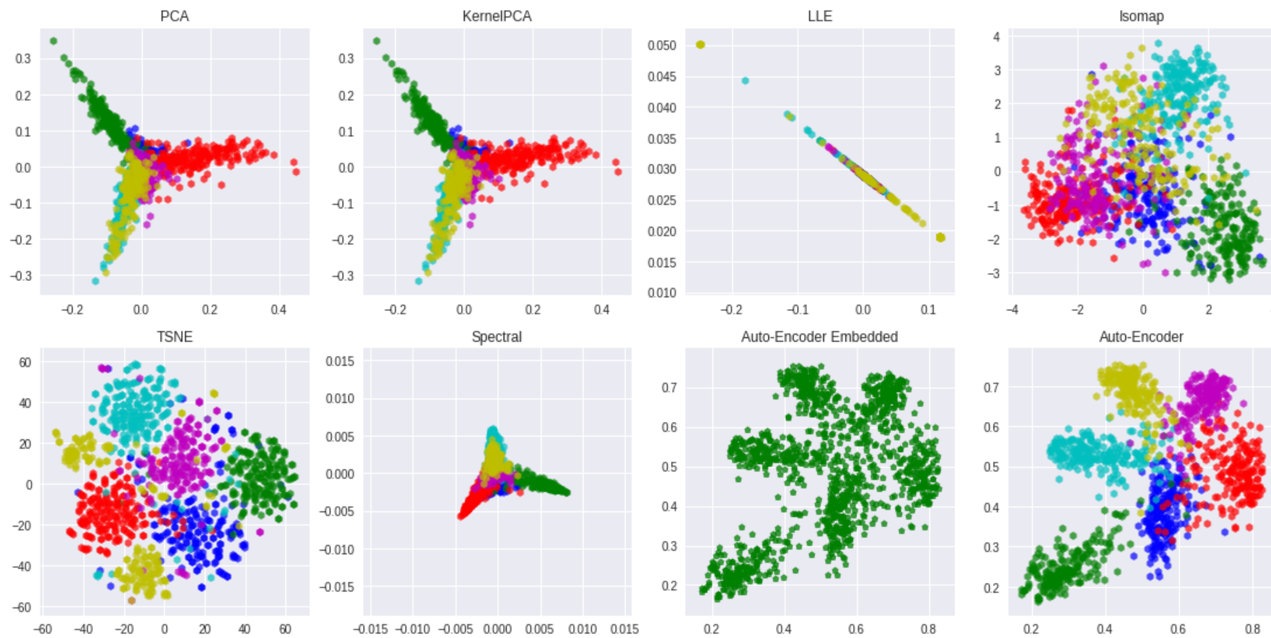
3.2.3. Unit norm on features vs samples:



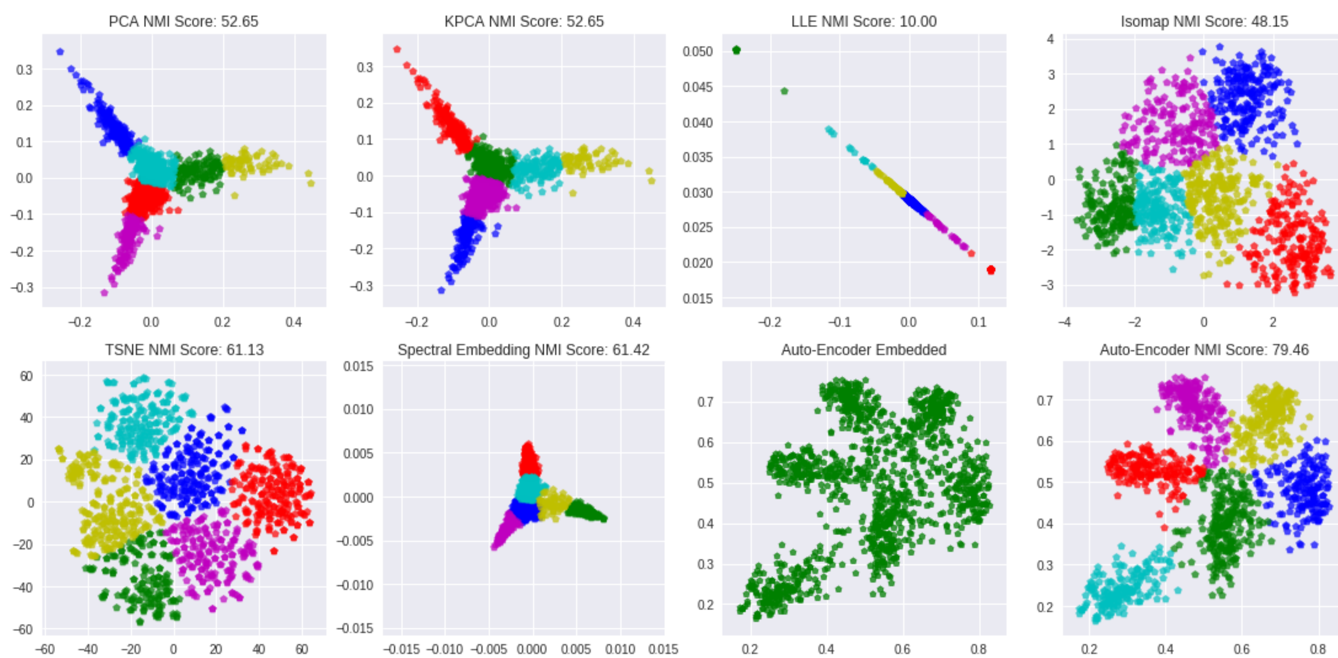
3.2.4. Performance of KMeans on 10 random factors of the model:



3.2.5. Performance of different techniques of dimensionality reduction on the 6 Groups of the 20 Newsgroups Dataset:



3.2.6. Performance of KMeans on different techniques of dimensionality reduction on the 6 Groups of the 20 Newsgroups Dataset:



3.2.7. Comprising with recent paper *(KATE: K-Competitive Autoencoder for Text, Yu Chen and Mohammed J. Zaki):*