Clustering Companies With Text

Gao Haocheng He Junjie

April 2020

Abstract

The aim of this project is to develop an effective model to cluster the companies within financial market, so that the group systematic risk and intra-group similarity can be better explained by the new clustering structure, compared with official classification by industries.

We tried revenue catagories text and introduction text disclosed in the annual reports as inputs, BERT, word2vec, dot2vec, Latent semantic indexing with different layers as company vector processing schemes, and greedy cosine-similarity, k-Means, Gausian Mixture Model, and Deep Embedding for Clustering model as clustering schemes. this working paper introduces how we impletemented these models and discusses briefly on the results.

Keywords: BERT, word2vec, dot2vec, Latent semantic indexing, cosine similarity, k-Means, Gausian Mixture Model, Deep Embedding for Clustering

1 The Vectorizing Models

2 The Clustering Models

2.1 Greedy Cosine Similarity

For company vector as inputs, in order to derive the firm-to-firm network representation of our industries, we use the vectors V_i and V_j for a pair of firms i and j to calculate the company cosine similarity or the firm pairwise similarity score as follows:

$$Company\,Cosine\,Similarity_{\,i,j} = (V_i \cdot V_j) \tag{1}$$

The network representation of firms is fully described by an $(N_t \cdot N_t)$ square matrix M_t (i.e., a network), where an entry of this matrix for row i and column j is the Company Cosine Similarity for firms i and j defined above. The large number of words used in business descriptions ensures that the matrix Mt is not sparse and that its entries are unrestricted real numbers in the interval [0, 1].

To process product-wise revenue classification text, an additional step is conducted before computing company cosine similarity. In this case, each product text vector is fed to the model, first a product pairwise similarity score is calculated as follows:

$$Product\,Cosine\,Similarity_{\,ai,bj} = (P_{ai} \cdot P_{bj}) \tag{2}$$

where $Product\,Cosine\,Similarity_{\,ai,bj}$ is the score of P_{ai}, the i^{th} producth of company a, and P_{bj}, j^{th} producth of company b. Therefore we can calculate the firm pairwise similarity score as follows:

$$Company\,Cosine\,Similarity_{\,a,b} = \sum_{i=1,j=1}^{n,n} Product\,Cosine\,Similarity_{\,ai,bj} w_{ai} w_{bj} \tag{3}$$

where $w_{ai}\, w_{bj}$ are the proportions of product i in company a's revenue and that of product j in company b's revenue, respectively.

The next stage of clustering [1] is conducted by taking the subsample of N single-segment firms. Then the industry classifications is initialized to have N industries, with each of the N firms residing within its own one-firm industry. Then is pairwise similarity for each unique pair of industries j and k, which is denoted as $I_{j,k}$.
To reduce the industry count to N-1 industries, we take the maximum pairwise industry similarity as follows:

$$\max_{j,k,j\neq k} I_{j,k} \tag{4}$$

The two industries with the highest similarity are then combined, reducing the industry count by one. This process is repeated until the number of industries reaches the desired number. Importantly, when two industries with m_j and m_k firms are combined, all industry similarities relative to the new industry must be recomputed. For a newly created industry l, for example, its similarity with respect to all other industries q is computed as the average firm pairwise similar- ity for all firm pairs in which one firm is in industry l and one in industry q as follows:

$$I_{l,q} = \sum_{x=1}^{m_t} \sum_{y=1}^{m_q} \frac{S_{x,y}}{m_l \cdot m_q} \tag{5}$$

Here, $S_{x,y}$ is the firm-level pairwise similarity between firm x in industry l and firm y in industry q.

2.2 k-Means

k-means clustering is a method of vector quantization, aiming to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster

centers or cluster centroid), serving as a prototype of the cluster. Given a set of observations $\{x_1, x_2, ..., x_n\}$, where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k sets $S = \{S_1, S_2, ..., S_k\}$ so as to minimize the inter cluster variance:

$$\arg\min_{s} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2 \tag{6}$$

2.3 Gaussian Mixture Model

A mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of certain different distributions with unknown parameters. One can think of mixture models as generalizing k-Means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent factor. The probability density is defined as a linear function of densities of all these K distributions:

$$p(X) = \sum_{k=1}^{K} \phi_k f_k(X|\mu_k, \Sigma_k) \tag{7}$$

In the formula, the distribution function of k^{th} cluster is characterized by distribution function f_k with weights ϕ_k, mean μ_k and covariance matrix Σ_k. For Gaussian Mixture Model, the probability density functions of all clusters are assumed to be Gaussian distribution.

2.4 Deep Embedding for Clustering

The Deep Embedding for Clustering (DEC) model is built upon the Stacked Autoencoder (SAE) model. Autoencoder is a kind of unsupervised learning structure that owns three layers: input layer, hidden layer, and output layer. The process of an autoencoder training consists of two parts: encoder and decoder. Encoder is used for mapping the input data into hidden representation, and decoder is referred to reconstructing input data from the hidden

representation. Then the structure of SAEs is stacking autoencoders into hidden layers by an unsupervised layer-wise learning algorithm and then fine-tuned by a supervised method. The structure of SAE is illustrated in Figure 1.

After greedy layer-wise training, we concatenate all encoder layers followed by all decoder layers, in reverse layer-wise training order, to form a deep autoencoder and then finetune it to minimize reconstruction loss. The final result is a multilayer deep autoencoder with a bottleneck coding layer in the middle. We then discard the decoder ayers and use the encoder layers as our initial mapping between the data space and the feature space, as shown in Figure 2.[3]

Following [2], we then add a new clustering layer to iteratively refine the clusters by learning from their high confidence assignments with the help of an auxiliary target distribution. Specifically, the model is trained by matching the soft assignment to the target distribution. To this end, we define our objective as a Kullback-Leibler (KL) divergence loss between the soft assignments q_i and the auxiliary distribution p_i as follows:

$$L = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}} \tag{8}$$

where the soft assignments q_i is defined as follows:

$$q_{ij} = \frac{(1 + ||z_i - \mu_j||^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'}(1 + ||z_i - \mu_{j'}||^2/\alpha)^{-\frac{\alpha+1}{2}}} \tag{9}$$

where z_i are the embedding vectors of each companies, and u_j are the centroids of each group j. Following [2], we set $\alpha = 1$ and the auxiliary distribution p_i as:

$$p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}} \tag{10}$$

The overall structure and the hyper-parameters are shown as in Figure 3. The training scheme is:

1. Pretrain the full SAE model and get the weights;

2. Pretrain a baseline machine learning classifier (we use k-Means in this model);

3. Construct the DEC model and load pretrain weights;

4. Initialize the clustering layer to the k-Means centroids;

5. Train the DEC model.

3 Data and Results

The input data contains two parts: 1. the revenue breakdown by different products stated in the annual reports, including the text and the sales figures as of 31 December 2018; 2. the descriptive text about the operartion, the main products of the company; 3. the daily stock price of the corresponding companies in 2019. The total sample size is 3801.

We evaluate each model by the following method: for each clustering of carried out by each model, feed each the time series return of each company with respect to the mean return of the group in a naive linear model, then calculate the average regression R square, which serves as the indicator of the performance of the models.

The results are summarized in Table 1

4 Discussions

5 Conclusion

Tables and Figures

Embedding	Clustering Models					
	official	greedy (prodduct-based)	greedy (company-based)	k-Means	GMM	DEC
BERT	0.3791					0.3425 1 pooling layer
Bag of Words						0.4139
dot to vec						0.4240 300 length
LSI						0.4100 500 length

Table 1: The average R square performance of each model

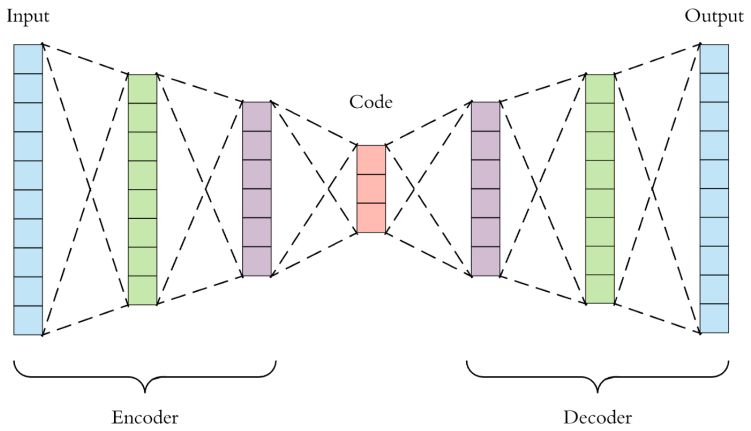

Figure 1: The Structure of a full SAE model, including encoder and decoder

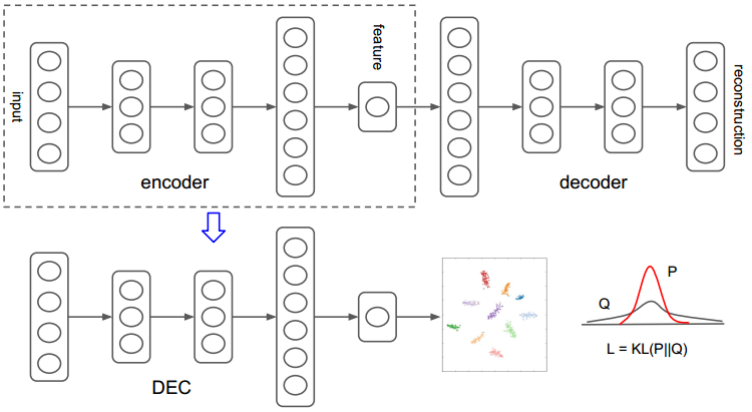

Figure 2: The Structure of an SAE model only considering encoder

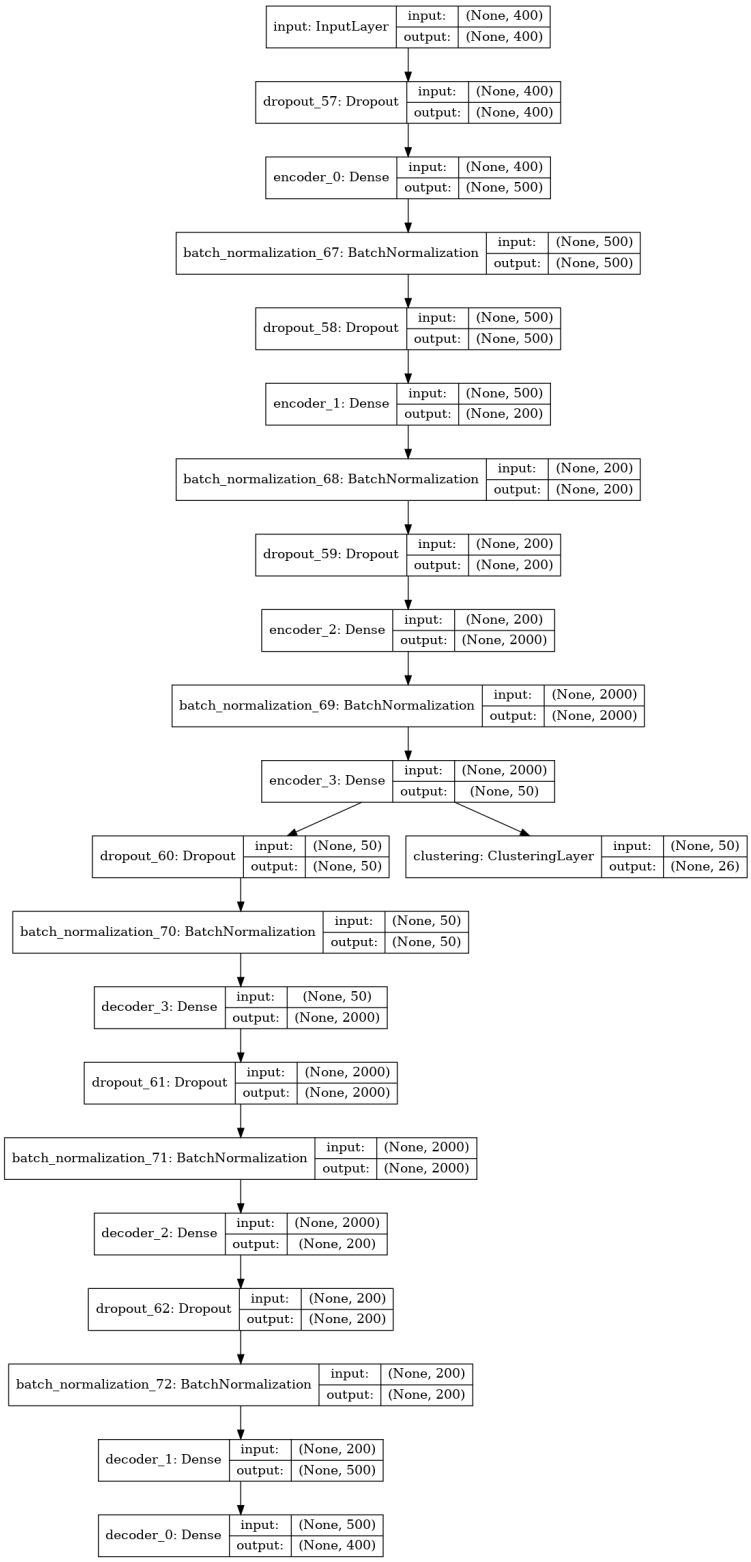

Figure 3: The Structure of the DEC model

References

[1] Gerard Hoberg and Gordon M Phillips. Text-based network industries and endogenous product differentiation. Working Paper 15991, National Bureau of Economic Research, May 2010.

[2] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 478–487, New York, New York, USA, 20–22 Jun 2016. PMLR.

[3] Jaime Zabalza, Jinchang Ren, Jiangbin Zheng, Huimin Zhao, Chunmei Qing, Zhijing Yang, Peijun Du, and Stephen Marshall. Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging. *Neurocomputing*, 185:1–10, 2016.