Learning data representations in unsupervised learning

Maziar Moradi Fard

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Maziar MORADI FARD

Thèse dirigée par **Eric GAUSSIER**

préparée au sein du **Laboratoire d'Informatique de Grenoble** dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Apprentissage de représentations de données dans un apprentissage non-supervisé

learning data representations in unsupervised learning

Thèse soutenue publiquement le **24 novembre 2020**, devant le jury composé de :

Monsieur ERIC GAUSSIER
PROFESSEUR DES UNIVERSITES, UNIVERSITE GRENOBLE ALPES, Directeur de thèse
Madame AMER-YAHIA SIHEM
PROFESSEUR DES UNIVERSITES, UNIVERSITE GRENOBLE ALPES, Présidente
Monsieur JULIEN VELCIN
PROFESSEUR DES UNIVERSITES, UNIVERSITE LYON 2, Examinateur
Monsieur LEBBAH MUSTAPHA
MAITRE DE CONFERENCES HDR, UNIVERSITE SORBONNE PARIS NORD, Rapporteur
Monsieur MARC TOMMASI
PROFESSEUR DES UNIVERSITES, UNIVERSITE DE LILLE, Rapporteur

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisors Prof. Eric Gaussier and Thibaut Thonet for the continuous support of my Ph.D study and related research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis.

Also, I would like to thank my fellow labmates especially Prof. Massih-Reza Amini for all the supports they provided during my Ph.D and all the fun we had together.

Last but not the least, I would like to thank my family: my parents and sisters for supporting me spiritually throughout writing this thesis and my life in general. Without their support, I could not imagine my self in this situation.

Abstract

Due to the great impact of deep learning on variety fields of machine learning, recently their abilities to improve clustering approaches have been investigated. At first, deep learning approaches (mostly Autoencoders) have been used to reduce the dimensionality of the original space and to remove possible noises (also to learn new data representations). Such clustering approaches that utilize deep learning approaches are called Deep Clustering. This thesis focuses on developing Deep Clustering models which can be used for different types of data (e.g., images, text). First we propose a Deep k-means (DKM) algorithm where learning data representations (through a deep Autoencoder) and cluster representatives (through the k-means) are performed in a joint way. The results of our DKM approach indicate that this framework is able to outperform similar algorithms in Deep Clustering. Indeed, our proposed framework is able to truly and smoothly backpropagate the loss function error through all learnable variables.

Moreover, we propose two frameworks named SD2C and PCD2C which are able to integrate respectively seed words and pairwise constraints into end-to-end Deep Clustering frameworks. In fact, by utilizing such frameworks, the users can observe the reflection of their needs in clustering. Finally, the results obtained from these frameworks indicate their ability to obtain more tailored results.

Résumé

En raison du grand impact de l'apprentissage profond sur divers domaines de l'apprentissage automatique, leurs capacités à améliorer les approches de clustering ont récemment été étudiées. Dans un premier temps, des approches d'apprentissage profond (principalement des autoencodeurs) ont été utilisées pour réduire la dimensionnalité de l'espace d'origine et pour supprimer les éventuels bruits (également pour apprendre de nouvelles représentations de données). De telles approches de clustering qui utilisent des approches d'apprentissage en profondeur sont appelées deep clustering. Cette thèse se concentre sur le développement de modèles de deep clustering qui peuvent être utilisés pour différents types de données (par exemple, des images, du texte). Tout d'abord, nous proposons un algorithme DKM (Deep k-means) dans lequel l'apprentissage des représentations de données (via un autoencodeur profond) et des représentants de cluster (via k-means) est effectué de manière conjointe. Les résultats de notre approche DKM indiquent que ce modèle est capable de surpasser des algorithmes similaires en Deep Clustering. En effet, notre cadre proposé est capable de propager de manière lisse l'erreur de la fonction de coût à travers toutes les variables apprenables.

De plus, nous proposons deux modèles nommés SD2C et PCD2C qui sont capables d'intégrer respectivement des mots d'amorçage et des contraintes par paires dans des approches de Deep Clustering de bout en bout. En utilisant de telles approches, les utilisateurs peuvent observer le reflet de leurs besoins en clustering. Enfin, les résultats obtenus à partir de ces modèles indiquent leur capacité à obtenir des résultats plus adaptés.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Artificial intelligence and more specifically machine learning have drawn a lot of attentions these days. The number of submitted papers in machine learning conferences are increasing rapidly. Online courses of machine learning are advertised on the internet frequently which reflects how much this field of study is becoming more popular. Nowadays, even Hollywood is attracted towards machine learning and the possible ways that it will influence our future. But these attractions raise the questions that why machine learning is becoming more popular and what is machine learning? To answer the latter question, we provide the standard definition proposed by Tom Mitchell. He defined machine learning as: *"the field of study which is concerned with the question of how to construct computer programs that automatically improve with experience".* To answer the first question, we provided below a few examples of the usage of machine learning which reflects its importance.

Machine learning algorithms have been able to inspire our daily lives in a variety of ways. Discriminating benign tumors from malignant ones is a medical example of using machine learning algorithms. Nearly 93% web traffic comes through search engines and machine learning is used by them to identify spams, low-quality contents, etc. Recent breakthroughs in mobile industry is formed by the help of machine learning as well. Identifying finger prints to unlock cell phones in the most secure and fastest way is obtained through machine learning. Recognizing the face to unlock the mobile phone where a lot of constraints such as light, angle, changes in the face (wearing eye glasses), etc. are present, is also one of the recent advances in mobile industry. Converting speech to text is one of the sensational abilities that

machine learning has provided as well. Moreover, machine learning algorithms have been used for protecting smart grids from cyberattacks. Thus, due to increasing impact of machine learning on our daily lives, performing scientific research in this area seems to be essential. Fundamentally, machine learning algorithms can be divided into three categories: 1) supervised, 2) semi-supervised, and 3) unsupervised learning.

In the case of supervised learning, the input data is coupled with its corresponding output (target) values. For instance, in the case of document classification, the categories (e.g., politics, scientific, etc.) pertaining to each data point are provided in advance which allows machine learning algorithms to discriminate different classes from each other. Semi-supervised learning embraces partially labeled datasets where the goal is to use labeled data points to categorize unlabeled ones. Unsupervised learning algorithms are used when no target information is available/required. In this thesis, we concentrate on clustering and Autoencoders.

Different from supervised and semi-supervised learning tasks, in clustering, the target values are not provided and the goal is to identify and discriminate different categories that exist in the data. Clearly, due to the lack of target values in clustering, this task is more difficult to deal with compared to supervised learning tasks.

So far, variety of algorithms have been proposed in the domain of clustering but probably the most well-known and widely used one is k-means [1].

k-means starts with randomly generated cluster representatives and tries to update them based on each observation of data points. To cluster a data point, the distance between the data point and each of the cluster representatives shall be computed. Then the data point will be assigned to the nearest cluster representative. There are several benefits regarding k-means including: 1) producing interpretable clusters (hard assignments), 2) easy to understand, 3) easy to implement, 4) efficient for large datasets, etc. These benefits helped k-means to be widely used in different areas.

However, there are some disadvantages regarding k-means: 1) k-means is an ill-posed problem, 2) k-means is performed in the original space of the data while this space can be noisy and contain redundant features, 3) subjectivity is a phenomenon that exists in k-means and standard k-means can not deal with it. To address the second problem, learning new data representations seems to be essential. Autoencoders are one potential candidate for learning new data representations. Indeed, representations learned through an Autoencoder can be fed to k-means algorithm to be clustered.

Predominantly, the representation learning phase is independent from

the clustering phase. In this case, after training an Autoencoder, k-means is applied on the new learned representations. Obviously, since these two phases are entirely disconnected, there is no guarantee that the new learned representations are adequate for clustering.

In order to combine these two disjoint phases into one joint phase where data representations (obtained through an Autoencoder) and k-means cluster representatives are updated simultaneously, a new objective function for k-means shall be defined– since k-means objective function is not differentiable and gradients can not be computed.

In this thesis we propose a Deep k-means framework where a new differntiable objective function for k-means is proposed. This differentiability enables gradients to be computed. Indeed, through our joint Deep Clustering framework, Autoencoder parameters (weights and biases) and k-means cluster representatives can be updated through backpropagation in a truly joint way. Our Deep k-means framework is evaluated on various kinds of data including image and text. The results indicate that our proposed framework outperforms the methods in Deep Clustering which share the most similarities to our framework.

As mentioned earlier, another important aspect of clustering is subjectivity. Indeed, while in standard clustering no side information is used, users might be interested in providing additional information to influence the clustering. In fact, additional information can help clustering to obtain better results. In case of document clustering, additional information can take the form of: 1) pairwise constraints and 2) seed words. In the earlier case, the user provides additional information about pairs of documents as must-link and cannot-link constraints (indicating respectively whether the documents in the pair are coming from the same cluster or not). In the latter case, the user may provide seed words for each category of documents.

Seed words correspond to sets of words which are able to characterize and define each cluster. For instance, {'player', 'coach', 'stadium'} can be considered as descriptive seed words for a sports cluster while {'politician', 'congress', 'president'} are able to describe politics. In this thesis we discuss our two individual frameworks which are able to include constraints. The framework which is able to include pairwise constraints is called Pairwise-Constrained Deep Document Clustering (PCD2C) and the one which is able to include seed words is called Seed-guided Deep Document Clustering (SD2C). Indeed in both PCD2C and SD2C frameworks, the algorithms are able to bias data representations based on the constraints provided by the user.

To evaluate our PCD2C and SD2C frameworks, we have used constraints

obtained from an automatic constraints selection procedure and several human constraints selection experiments. In the earlier approach, constraints are extracted automatically from the data. In the latter approach, we carefully designed several experiments where users were asked to participate and provide constraints for document clustering. Indeed, we tried to replicate a real world case scenario which enables us to obtain more accurate analysis regarding the performance of our proposed PCD2C and SD2C frameworks in such cases.

The results from both automatic and human constraints selection indicate that our proposed PCD2C and SD2C frameworks are able to include constraints in an end-to-end Deep Clustering framework. Finally, it is shown that in most cases, constraints help SD2C and PCD2C to obtain better results compared to when no constraints are used.

The organization of the thesis is as follows: in Chapter 2 we discuss the basics of clustering (more specifically k-means) and Autoencoders. Chapter 3 includes related works in the domain of Deep Clustering and constrained clustering. Our Deep k-means framework is proposed in Chapter 4 and finally Chapter 5 contains our proposed PCD2C and SD2C frameworks.

CHAPTER 2

Definitions and Notations

Clustering is one of the essential tasks in machine learning where the goal is to find existing patterns in unlabeled data points and group them into several clusters based on their similarities or dissimilarities. Up to now, several works tried to improve clustering algorithms from different perspectives. One of the most important aspects of improving clustering algorithms is reducing the dimensions of the underlying data by learning new data representations which are adequate for clustering. Indeed, the underlying data might include redundant and noisy features which can reduce the performance of the clustering. Using Autoencoders is a possible solution to learn new data representations and also helps to reduce the dimensions of the data to avoid standard issues in machine learning such as curse of dimensionality. Since clustering and Autoencoders are the two most important aspects of this thesis, in this chapter we review their basic concepts.

2.1 Clustering

Learning patterns from unlabeled data can be much more challenging than the case in which true labels are available. Clustering is one way to learn patterns from unlabeled data and can be defined in many different ways [2] but the standard definition is to group data points with similar patterns into the same cluster [3]. One can define several similarities or dissimilarities such as Euclidean distance, cosine distance, etc. Previously, several clustering algorithms with different perspectives towards grouping data points have been proposed but since the focus of the thesis is towards the k-means

algorithm, we only discuss this algorithm and Fuzzy C-means in details.

2.1.1 k-means Clustering

k-means [1] is one of the most well-known and widely used Machine Learning algorithms. This algorithm starts with randomly generated cluster representatives and tries to update them based on each observation of data points. To cluster a data point, the distance between the data point and each of the cluster representatives shall be computed. Then the data point will be assigned to the nearest cluster representative. Afterwards, the selected cluster representative will be updated as the mean of all its assigned data points to optimize the so-called k-means objective function. In the remainder, x denotes an object from a set $\mathcal{X}$ of objects to be clustered. K denotes the number of clusters to be obtained, $\mathbf{r}_k \in \mathbb{R}^p$ the representative of cluster k, $1 \leq k \leq K$, and $\mathcal{R} = \{\mathbf{r}_1, \ldots, \mathbf{r}_K\}$ the set of representatives. We use the term *representative* rather than *centroid* here to emphasize the fact that in case of similarity functions, or dissimilarity functions different from the Euclidean distance, the representative of a cluster does not necessarily coincide with its centroid. The k-means objective function can be defined as:

$$\min_{\mathcal{R}} \sum_{x \in \mathbf{X}} ||x - c(x; \mathcal{R})||^2 \tag{2.1}$$

Where $c(x; \mathcal{R}) = \arg\min_{r \in \mathcal{R}} ||x - r||^2$ is the nearest cluster to x. Moreover, in [4], further advantages of k-means algorithms are mentioned such as:

- Producing interpretable clusters (hard assignments);
- Easy to understand;
- Easy to implement;
- Efficient for large datasets.

Due to several advantages regarding the k-means algorithm which we discussed above, we have used this algorithm in our proposed framework.

Despite many advantages with respect to k-means, still there are a few drawbacks that shall be taken into account. In [5] the initialization of the k-means has been discussed. Based on this survey, the performance of the k-means greatly depends on the initial cluster representatives. To address this problem k-means++ [6] has been proposed. In k-means, it is mandatory for all data points to be categorized in one of the clusters thus k-means can be sensitive towards noise and outliers [7]. Several other variations of k-means have been proposed to improve the performance of k-means. In [8]

distributed k-means has been proposed which decreases the runtime of k-means for large datasets. More discussions about k-means shortcomings and some proposed approaches for improving this algorithm have been discussed in [9].

Algorithm 1: k-means algorithm

1. Randomly initialize cluster representatives
2. Assign each data point to the closest cluster representative
3. Update the selected cluster representative as the average of all data points assigned to this cluster.
4. Repeat steps 2 and 3 until convergence
5. End

2.1.2 Fuzzy C-means

Fuzzy C-means (FCM) is an improved version of k-means algorithm which was proposed in [10]. In this algorithm, each data point can be assigned to more than one cluster. In other words, it introduces a new objective function which determines how much a data point belongs to each cluster. This function is called **membership function** and it can take values between 0 and 1. FCM can be formulated as below:

$$\begin{aligned} \min_{\mathcal{R}} \sum_{k \in K} \sum_{x \in \mathcal{X}} \mu(x, r_k)^m ||x - r_k||^2; 1 < m < \infty \\ \mu(x, r_k) = \left[\sum_{l=1}^{K} \left(\frac{||x - r_k||^2}{||x - r_l||^2} \right) \right]^{-1} \end{aligned} \tag{2.2}$$

Similarly to k-means algorithm which suffers from the initialization of cluster representatives, FCM suffers from this issue as well but several algorithms have been proposed to deal with this problem [11, 12].

Moreover, both k-mean and and FCM are performed in the original space of the data while it can be noisy and include redundant features. To mitigate these issues, one potential solution is utilizing Autonecoders which are able to learn new data representations.

In the following section, we discuss details of the Autoencoder used in our framework.

2.2 Autoencoders

Autoencoders are at the core of modern clustering (deep clustering) approaches. In this Chapter, we are not going to dive into deep clustering approaches but we rather give an overview of Autoencoders and their applications– more details regarding deep clustering approaches are discussed in Chapter 3. Autoencoders are a special kind of neural networks in which the objective of the network is to reconstruct the input in the output layer. These networks can be divided into two parts: 1) encoder and 2) decoder. The encoder is the first part of an Autoencoder where the data will be passed through several layers of neural networks and finally it will be embedded into a new space. This new space is called embedding space. Afterwards, the embedded input shall be fed to the decoder. The decoder is the mirrored version of the encoder in which the number of neurons in its final layer is equal to the number of dimensions of the input. Indeed, the final layer of an Autoencoder yields a vector which is aimed to reconstruct the input. Finally, after training an Autoencoder, the obtained embedded data (through encoder) can be used for variety of purposes (supervised or unsupervised tasks). The Autoencoder loss function takes the following form:

$$\mathcal{L}_{\text{rec}}(\mathcal{X}, g_\eta \circ f_\theta(\mathcal{X})) = \sum_{x \in \mathcal{X}} \delta^I(\mathbf{x}, g_\eta \circ f_\theta(\mathbf{x})) \tag{2.3}$$

where $\mathcal{X}$ is a set of input samples and θ and η are respectively the encoder and decoder parameters. δ^I shall be defined as dissimilarity function (e.g., Euclidean, Cosine, etc.). As mentioned earlier, an Autoencoder consists of two parts, an encoder and a decoder. The encoder is formulated as $f_\theta : \mathbb{R}^d \to \mathbb{R}^p$ and the decoder is formulated as $g_\eta : \mathbb{R}^p \to \mathbb{R}^d$ where $\mathbb{R}^d$ represents the input space and $\mathbb{R}^p$ represents the space in which learned data representations are to be embedded. Finally, an Autoencoder is formulated as $g_\eta \circ f_\theta(\mathbf{x})$ meaning that first the encoder projects the input in the embedding space and then the decoder tries to reconstruct the original data from the embedded representation. This loss function is trained by using backpropagation [13].

As Formula 2.3 implies, the Autoencoder loss function is optimized when the input is well reconstructed in the output of the Autoencoder. Indeed, while trying to reconstruct the input in the output layer, the embedding space ($\boldsymbol{f}_\theta(x)$) captures the most salient information about the input data. The transformed data in the embedding space is the input to both supervised [14] and unsupervised algorithms [15]. Autoencoders have been used in different fields of studies like medical image processing [16], image classification [14], image generation [17], speech recognition [18], speech-to-text translation [19],

etc.

Up to now, different kinds of Autoencoders such as Variational Autoencoders [20], Regularized Autoencoders [21, 22], etc. have been proposed. We have used undercomplete Autoencoders in our proposed framework which are one of the simplest versions of Autoencoders.

Undercomplete Autoencoders: In section 2.2 we discussed that Autoencoders consist of an encoder and a decoder (mirrored version of the encoder). If the dimension of the projected data in the embedding space is less than the dimension of the original input data, the Autoencoder is called **undercomplete**. In this case, the representations learned through the embedding layer are able to describe the original input in the most distinctive manner. In other words, not only the dimensions of the original data will be reduced (which can be helpful to get rid of redundant features), new distinctive representations from the original data will be obtained. In this case, one can use different types of neural network layers such as Feed Forward Neural Networks (FFNN's), Convolutional Neural Networks (CNN's) [23], etc. Figure 2.1 illustrates an example of undercomplete Autoencoders.

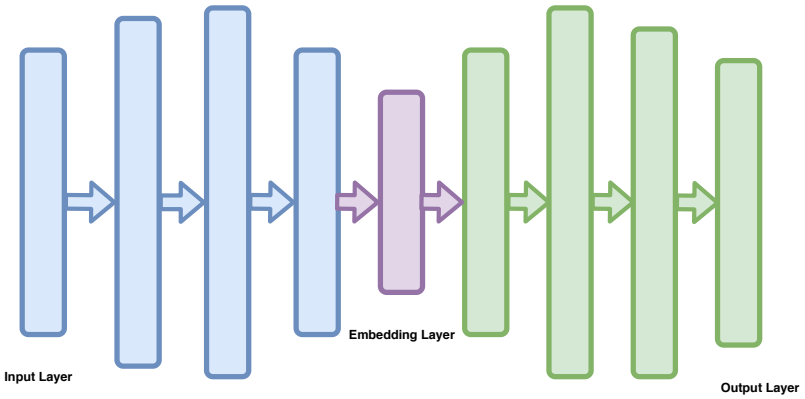


Figure 2.1: An example of Undercomplete Autoencoders. Blue layers represent the encoder and green layers represent the decoder while the purple layer represents the embedding layer.

Due to simplicity and great performance of uncercomplete Autoencoders, we have used them in our proposed framework which is discussed in Chapter 4. In Chapter 3 we discussed the usage of Autoencoders in other Deep Clustering frameworks.

2.3 Evaluation Metrics

In our experiments, the clustering performances of the different methods are evaluated with respect to three standard measures: Normalized Mutual Information (NMI), clustering accuracy (ACC) and the Adjusted Rand Index (ARI).

2.3.1 NMI

NMI is an information-theoretic measure based on the mutual information of the ground-truth classes and the clusters obtained from a clustering algorithm [24]. Formally, let $S = \{S_1, \ldots, S_K\}$ and $C = \{C_1, \ldots, C_K\}$ denote the ground-truth classes and the obtained clusters, respectively. S_i (resp. C_j) is the subset of data points from class i (resp. cluster j). Let N be the number of points in the dataset. The NMI is computed according to the following formula:

$$\mathrm{NMI}(C, S) = \frac{I(C, S)}{\sqrt{H(C) \times H(S)}} \tag{2.4}$$

where $I(C, S) = \sum_{i,j} \frac{|C_i \cap S_j|}{N} \log \frac{N|C_i \cap S_j|}{|C_i||S_j|}$ corresponds to the mutual information between the partitions C and S, and $H(C) = -\sum_i \frac{|C_i|}{N} \log \frac{|C_i|}{N}$ is the entropy of C.

2.3.2 ACC

Different from NMI, ACC measures the proportion of data points for which the obtained clusters can be correctly mapped to ground-truth classes, where the matching is based on the Hungarian algorithm [25]. Let s_i and c_i further denote the ground-truth class and the obtained cluster, respectively, to which data point x_i, $i \in \{1, \ldots, N\}$ is assigned. Then the clustering accuracy is defined as follows:

$$\mathrm{ACC}(C, S) = \max_{\phi} \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\{s_i = \phi(c_i)\} \tag{2.5}$$

where $\mathbb{I}$ denotes the indicator function: $\mathbb{I}\{\text{true}\} = 1$ and $\mathbb{I}\{\text{false}\} = 0$; ϕ is a mapping from cluster labels to class labels.

2.3.3 ARI

The third metric that we have used to evaluate different clustering algorithms performances is called ARI [26]. ARI counts the pairs of data points on which the classes and clusters agree or disagree, and is corrected for chance. Formally, ARI is given by:

$$\text{ARI}(C,S) = \frac{\sum_{ij}\binom{|C_i\cap S_j|}{2} - \binom{N}{2}^{-1}\sum_i\binom{|C_i|}{2}\sum_j\binom{|S_j|}{2}}{\frac{1}{2}\left(\sum_i\binom{|C_i|}{2}+\sum_j\binom{|S_j|}{2}\right) - \binom{N}{2}^{-1}\sum_i\binom{|C_i|}{2}\sum_j\binom{|S_j|}{2}} \tag{2.6}$$

2.4 Tools for Implementation

Python is the main programming language that we have used to implement our proposed methods and most of the baselines. For the methods that are based on DNN's, we have used Tensorflow for the implementation. Although we have re-implemented most of the baselines, for some others we have used the original code published by their authors (e.g., Java, Keras, etc.). More details about the implementations and libraries can be found in the following github link:

https://github.com/MaziarMF

CHAPTER 3

Related Work

3.1 Overview

In Chapter 2, we reviewed basics of clustering and Autoencoders and we discussed that Autoencoders can be used for learning new data representations in a Deep Clustering (DC) framework. Autoencoders are a special case of using DNNs in DC and standard DNN approaches have been utilized in DC frameworks as well. Indeed, the usage of standard DNNs in DC frameworks is quite similar to their usage in supervised learning tasks but the difference is that instead of obtaining class probabilities, cluster probabilities are obtained through a softmax layer (last layer).
In fact, we can categorize DC algorithms based on different aspects but we propose to categorize them based on how DNNs are being used in DC frameworks as suggested in [27]. Indeed, these approaches can be divided into two groups: 1) Autoencoder based approaches and 2) non-Autoencoder based approaches. In the DC frameworks where Autoencoders are used for learning data representations, the focus is to incorporate Autoencoder reconstruction loss in the final loss such that it can help the algorithm to obtain more suitable representations which are apt for clustering. On the other hand, non-Autoencoder based approaches which utilize standard DNNs in their frameworks try to embed the data points into m (number of layers) distinctive spaces and finally obtain the clustering probabilities. Later in this chapter, we review papers in both categories and provide more detailed information about them.

Additionally; in this Chapter, we discuss different kinds of constraints

which are applicable in standard clustering and we review some of the papers in this domain.

3.2 Deep Clustering

As mentioned earlier, DC approaches can be categorized based on different aspects (loss function, type of neural network, and etc.) but in this thesis we proposed to categorize them based on using Autoencoder. Indeed, these approaches can be divided into Autoencoder Based and non-Autoencoder based approaches. Following in this thesis, we discuss both approaches and review works in these domains.

3.2.1 Autoencoder Based Deep Clustering Approaches

The DC algorithms mostly use Autoencoders to learn new representations from raw data. Usually in these algorithms, the Autoencoder loss is added to the clustering loss to jointly learn data representations and cluster representatives through back propagation. In this case, to stabilize the algorithms, performing pretraining is essential [28]. Pretraining consists in training an Autoencoder which is followed by applying k-means on the learned representations (this process shall be performed in a disjoint way). Then, the learned Autoencoder parameters (e.g., weights and biases) and cluster representatives (through applying k-means on the learned representations) are used to initialize the Autoencoder parameters and cluster representatives before training a DC framework.

One of the most known algorithms in this category is Deep Clustering Network (DCN) [29]. This k-means based algorithm is one of the first algorithms in Autoencoder based DC algorithms. DCN first pretrains an Autoencoder then after initializing Autoencoder parameters and cluster representatives, optimizes Autoencoder loss and clustering loss in an alternative way. This algorithm claims to be k-means friendly meaning that the final representations are closed to a k-means friendly space. Since our proposed method is also based on k-means, later we will provide more details and analysis with respect to this algorithm. Preserving local structure of the data is a challenging task that many papers tried to address. Deep Embedding Network (DEN) [30] is one of these algorithms which first tries to pretrain an Autoencoder and then applies a locally preserving constraint. Improved Deep Embedding Clustering (IDEC) [31] is another approach which tries to preserve the local structure of the data. This soft clustering algorithm employs Autoencoders to learn representations for the clustering as well. Moreover, the clustering

loss contains Kullback Leibler (KL) divergence between the soft labels and the predicted labels. Finally, the total loss function is a combination of clustering loss and Autoencoder loss. IDEC obtained state-of-the-art results for different types of datasets (e.g., image, text and etc.). We have selected this algorithm as another baseline to compare our proposed framework with. Later in this chapter, details of this algorithm will be discussed as well.

Deep Multi-Manifold Clustering (DMC) [32] is a deep learning based multi-manifold clustering approach that proposes a loss function which contains three elements. The first element is the clustering loss which makes the data representations to be cluster friendly. The second term in the loss is the Autoencoder loss which helps to obtain new data representations and finally a locally preserving loss which makes the representations useful and meaningful. In addition to these approaches, in [33] Gaussian Mixture Model (GMM) [34] and k-means are used in a deep continuous framework where cluster representatives and network parameters are updated in a joint way. In this paper, a new relaxation funtion is introduced based on Alternating Direction of Multiplier Method (ADMM) [35]. A GMM based deep learning model is also proposed in [36]. In this paper, first it has been tried to decrease intra-cluster variance by obtaining more suitable representations and finally increase inter-cluster variance to make clusters more separable.

Figure 3.1 illustrates the general architecture of Autoencoder based DC approaches.

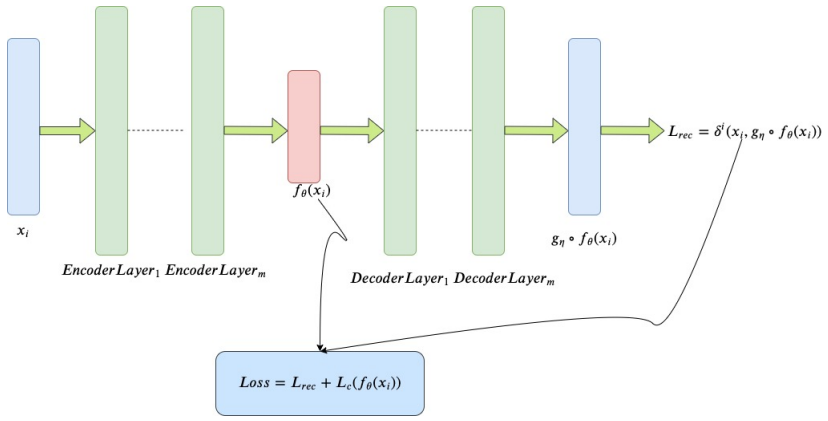


Figure 3.1: General DC architecture.

Different from approaches mentioned earlier, in [37] a new fast clustering approach based on the denoising Autoencoders is proposed. This method is inspired by the objective function of k-means and spectral clustering. In [38], deep stacked Autoencoders and graph clustering are combined such that by the help of stacked Autoencoders, the nonlinear embeddings of the graph is obtained first. Sequentially, this algorithm performs k-means algorithm on the learned embeddings to obtain final cluster representatives. It has been proved that the proposed method obtains state-of-the-art results compared to spectral clustering algorithm.

Clustering can be performed by utilizing pair-wise constraints [39] as well. The main idea of this paper is that similar pairs should have similar representations in the embedding space while dissimilar pairs should have different distant embeddings. The authors have used K-Nearest Neighbor (KNN) to obtain pairwise relations (for each data point the closes point is the similar point). This approach can be extended when partial labels are available (semi-supervised learning). In this case, both KNN and labels are used to build the pairwise relations. In [40] pairwise relations are being used in the pretraining phase. Indeed, this paper is focused more on pretraining phase of clustering since it has been assumed that by achieving better results in the pretraining phase, we can obtain overall better results in DC as well. The pretrained loss contains the weighted sum of all pairwise similarities. It has been illustrated that by utilizing this strategy, one can obtain state-of-the-art results for MNIST dataset. An ensemble learning framework is proposed in [41]. This paper focuses on non-linear dimensionality reduction by using an Autoencoder and attempts to fuse different layers of Autoencoder to capture salient information from high dimensional data. This framework can achieve robust results especially for imbalanced clusters. In [42], different Autoencoders have been combined to produce different representations for each cluster. No regularization techniques are needed for this algorithm to avoid singleton cluster. In [43] the embeddings obtained from an Autoencoder are fed to a DNN to assign embeddings to different clusters. It has been claimed that this approach can yield a k-means friendly space (similar to DCN). Indeed, by applying k-means on the learned embeddings (after training phase), more suitable results are obtained compared to similar methods. Maximizing Inter-cluster variance and minimizing Intra-cluster variance for image datasets has been studied in [44]. In this paper, for each image, one related image and one non-related image are selected. The goal is to make the representation of the input image close enough to the related image while it is well-separated from the non-related one.

In [45] a dual deep Autoencoder has been used which maps the input

and its noisy version of the image into a latent space then applies clustering. The results have shown that the method is robust to noisy data as expected. Dual-Adversarial Autoencoder is another method which uses dual adversarial Autoencoders. This method is able to simultaneously maximize the likelihood function and mutual information between observed examples and a subset of latent variables [46]. A dynamic Autoencoder is designed and used in an unsupervised framework such that the reconstruction error is gradually eliminated from the loss function [47]. Inspired by PCA-k-means [48], an orthogonal feature producing Autoencoder has been proposed in [49]. This Autoencoder has been employed in a joint clustering framework.

A new definition of denoising Autoencoder for text data has been proposed in [50]. In this paper, each document is not only influenced by its own information but by the neighboring documents as well. Neighboring documents are identified by cosine similarity. This algorithm has shown advantageous results specially on real-world datasets where a clustering algorithm is applied on the learned embeddings. The combination of the feature selection algorithms and Autoencoders has shown promising results [51]. Since not all hidden units are beneficial to obtain a well-separable embedding space, feature selection algorithms are used to extract only useful hidden units.

Variational Deep Embedding [52] is an unsupervised DC framework which employs Variational Autoencoders and GMM in its framework. In this algorithm, first GMM picks a cluster and then a latent embedding is generated from the chosen cluster. Finally DNNs decode the latent embedding into an output.

Deep Adversarial Clustering (DAC) [53] is another deep clustering approach which is based on Adversarial Autoencoders [54]. This algorithm utilizes adversarial training procedure to match the posterior of the latent representation with the prior distribution. DAC comprises of three losses: clustering loss, adversarial loss and reconstruction loss. Fig 3.2 illustrates the architecture of the Adversarial Autoencoder which is used by DAC.

Categorical Generative Adversarial Network (CATGAN) [55] is another variation of GANs which has been used for unsupervised learning representation. The proposed framework of CATGAN is slightly different from GAN. Indeed, in this approach, the discriminator tries to classify the data into a chosen number k categories instead of two categories (real or fake) like GAN. In this case, the generator generates samples belonging to one of the k clusters. Figure 3.3 illustrates the architecture of this approach.

ADEC [56] utilizes adversarial Autoencoders and k-means in a joint DC framework where cluster representatives and network parameters are updated

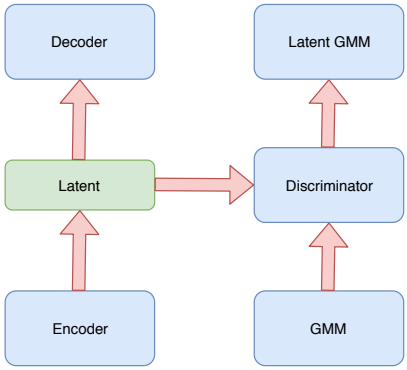


Figure 3.2: AAE architecture.

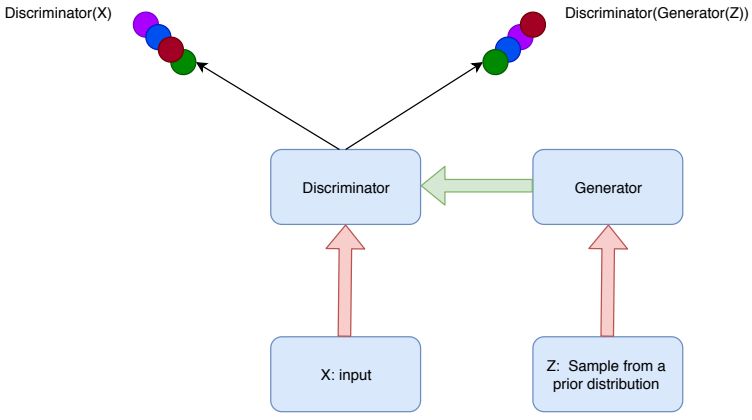


Figure 3.3: CATGAN architecture.

in a joint way.

Subspace clustering is another concept of deep learning which recently has been combined with DNNs. In subspace clustering the goal is to project each cluster into different subspaces. In [57] a novel architecture of deep subspace clustering has been proposed. In this paper, both local and global structure of the data have been preserved in the embedding space. Moreover, in this paper a prior sparsity information is considered in the representation learning phase to preserve the sparse reconstruction relations over the whole data. Another subspace clustering based on DNN is proposed in [58] where learning representations and clustering assignment has been done simultaneously. Using self-expressiveness property [59, 60] in deep subspace clustering was another novelty that is presented in [61]. In this paper deep clustering is framed as a subspace clustering problem such that mapping from the original data space to a low-dimensional subspace is learned by an Autoencoder. The key contribution of this paper is the self-expressive layer in junction with encoder and decoder which have been defined to encode self-expressiveness property. In [62] a deep generative CNN is used to preserve local information of data.

Deep Continuous Clustering (DCC) [63] is another deep clustering algorithm which is based on Robust Continuous Clustering (RCC) [64]. As the name implies, this method also benefits from jointly optimizing an Autoencoder and cluster representatives. In this method, the data is embedded into a lower dimension by an undercomplete Autoencoder and then it will be assigned to one of the clusters. Being independent of any prior knowledge of the number of ground-truth clusters is the highlight of the paper.

Deep Spectral Clustering (DSC) is another DC algorithm which has been studied recently. In [65] first a few points called landmarks are used to generate an adjacency matrix of the corresponding graph of the dataset. Then, a deep Autoencoder is used to perform eigen decomposition through defining the laplacian matrix of the graph. Spectral Clustering via Ensemble Deep Autoencoder Learning (SC-EDAE) [66] is another approach which has combined spectral clustering and DNNs. In this paper, first several Autoencoders with different hyperparameters are used to produce different representations. Then, these representations will be combined into a space which is shared among Autoencoders. Finally, the ensemble representations are fed to a subspace clustering algorithm. SpectralNet is a new approach in DSC [67] which is proposed to address major drawbacks of spectral clustering. In this algorithm, first an Autoencoder learns a function which maps the input data into the eigenspace of their associated laplacian matrix then

groups them into different clusters. Additionally, in this paper, a new training procedure based on stochastic optimization is proposed. The learned mapping function by Autoencoder allows SpectralNet to reach a better generalization while the proposed stochastic optimization improves its scalability.

3.2.2 Non-Autoencoder based Deep Clustering Approaches

In this category, instead of utilizing Autoencoders, any DNN algorithm can be used to map the data from the original space into k separate clusters directly.

Deep Embedded Regularized Clustering (DEPICT) [68] utilizes several layers of CNNs which are stacked by a multinomial logistic regression function. Also a KL divergence loss function has been used for clustering loss which is normalized by a prior reflecting the frequency of cluster assignments. Moreover, a regularization term has been defined which encourages balanced clustering. This algorithms shows promising results for clustering image data.

In the majority of the DC algorithms designed for image datasets, mostly the attention is towards introducing new architectures of CNNs. For instance, General Adversarial Neural Networks (GANs) [69] is one of the most popular DNNs algorithms that have been used for both supervised and unsupervised tasks. In this framework, a generative model and a discriminative model compete to defeat each other. The generative model tries to capture the distribution of the data while the discriminative model tries to estimate the probablity that a sample comes from training samples rather than generative model. GANs have been first studied for unsupervised learning in [70] where they have been used to learn representations in an unsupervised way. In this paper, a new CNN based GAN with a few constraints on the architecture has been proposed for learning representations of image datasets in an unsupervised manner. Although it is not guaranteed that this architecture can be used for other types of data rather than images, this approach obtains promising results for image datasets.

HashGan [71] is another approach which addresses the limitations of Deep Hash Clustering [72] approaches by introducing a new loss function and sharing discriminator and encoder parameters. This approach has shown improvements on the real image datasets compared to similar other approaches. A new framework of unsupervised learning which employs GAN is proposed in [73]. In this framework, clustering is used as a final step to solve the problem of unsupervised fine-grained object category discovery.

In [74] a novel algorithm is proposed which uses a joint optimization to learn cluster representatives and network parameters simultaneously. In this algorithm, first data is embedded using NNs then a cost function is defined such that learning cluster representatives and data representations are performed simultaneously. Finally, an additional layer is introduced to map the embedded data into logits which allows obtaining clustering memberships. In [75] multiple convolutional layers are stacked on top of each other and in the last layer, softmax is applied to obtain clustering distributions. In this paper, a discriminative loss is applied which enables the network to decrease intra-cluster variance and increase inter-cluster variance. This method achieves state-of-the-art results on image datasets.

Clustering Convolutional Neural Network (CCNN) [76] is another approach which as the name implies, is strongly dependent on CNNs. This approach first uses a pretrained model on ImageNet dataset to learn initial data representations and cluster representatives. In each iteration of the training phase, cluster representatives, cluster assignment and network parameters are updated (through back propagation). This algorithm reduces memory consumption compared to other CNN-based algorithms and proposes a novel way of updating cluster representatives. In [77] a recurrent process has been proposed. In this algorithm a CNN has been used to map the data from original space into several new spaces and finally hierarchical clustering is applied on the learned representations to obtain cluster representatives. In the forward pass, two clusters can be merged based on the predefined loss while in the backward pass, data representations obtained from CNN will be updated based on the merged clusters.

In [78] the clustering task has been transformed into a binary pairwise-classification framework to determine whether pairs of images belong to the same cluster or not. Due to the unsupervised nature of clustering, the CNN-based classifier in this approach is only trained on the noisily labeled examples obtained by selecting difficult-to-cluster samples in a curriculum learning fashion.

Information Maximizing Self-Augmented Training (IMSAT) [79] is another method which is a combination of NNs and regularized Information Maximization (RIM) [80]. In this paper, the goal is to design a function which maps the data into several discrete representations (clustering is a special task of this approach).

Deep Nonparametric Clustering (DNC) [81] is another clustering method which relies on Deep Belief Networks (DBN) [82]. In this algorithm first data representations are obtained by passing the input data through a DBN. Sequentially, a nonparametric maximum margin clustering [83] is applied

on the learned representations. Afterwards, a fine-tuning process will be performerd to update DBN parameters. Deep Density Clustering is another novel approach in the domain of DC [84] which has been used for image data. As the name implies, this method is based on density of the local neighborhood in the feature space.

Deep Embedding Clustering (DEC) [85] is one of the most remarkable methods in DC. In this algorithm, first an Autoencoder is pretrained by optimizing the reconstruction loss. Then, the decoder part of the network will be eliminated and the encoder part will be fine-tuned in the training process of clustering. In this approach, cluster representatives will be updated based on the KL divergence between soft labels and auxiliary target distribution. This algorithm has become one of the main baselines among DC algorithms. DEC with Data Augmentation (DEC-DA) [86] is an extension of DEC by using data augmentation. In this algorithm, first an Autoencoder is pretrained by using both original data and augmented data. Finally clustering is applied on the original non-augmented data (similar to DEC).

Discriminatedly Boosted Clustering (DBC) [87] is another clustering method which has exactly the same structure as DEC but instead of using FFNNs, it benefits from CNNs. Using CNNs vividly empowers DBC for clustering image datasets compared to DEC .

Completely different from other DC algorithms that utilize only encoder part of an Autoencoder, in [88] partial pairwise relationships between all samples of a given dataset are used to build a DC framework. In this paper, the relation between each pair is identified as must-link (ML) (of the same cluster) or cannot-link (CL) (of different clusters). The data samples and their relationships (ML or CL) are fed to a DNN and the goal is to: 1) minimize KL divergence loss between ML pairs and 2) maximize the KL divergence loss between CL pairs. In this algorithm, clustering assignments are obtained at the output layer where a softmax function is applied. To obtain reasonable results, this algorithm requires additional information about pairwise relation of the samples while in real scenarios where all labels are not available, this algorithm can not function well. Figure 3.4 illustrates the architecture of this algorithm.

Despite interesting recent advances in DC algorithms, most of these algorithms are designed for specific tasks or specific kinds of data (e.g., images, text and etc.). There are a few algorithms that have attempted to address the imitations exist in DC. Since our proposed DC framework is based on k-means algorithm, DCN and IDEC are the ones which share the most characteristics with our approach. The details of these algorithms are given below.

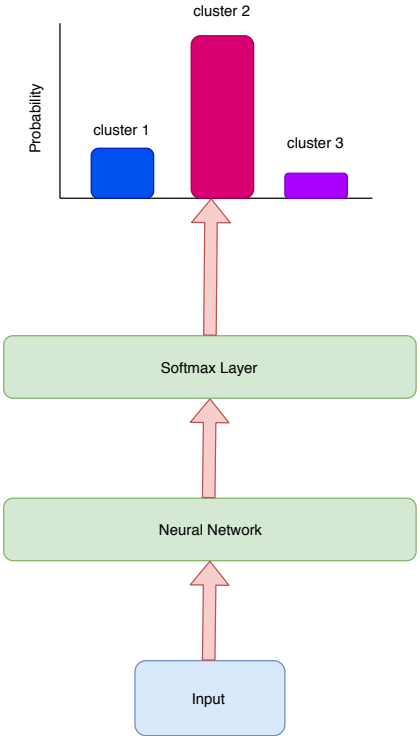


Figure 3.4: Pairwise Clustering Architecture.

IDEC As mentioned earlier, IDEC is an improved version of DEC where instead of discarding the decoder part of the Autoencoder, it benefits from the reconstruction loss obtained through the decoder during the training. The reconstruction loss is defined as:

$$\mathcal{L}_{\text{rec}}(\mathcal{X}, \eta, \theta) = \sum_{x \in \mathcal{X}} \delta^I(\mathbf{x}, g_\eta \circ f_\theta(\mathbf{x})) \tag{3.1}$$

Where $f_\theta(x)$ is the embedding of the data and $g_\eta \circ f_\theta(x)$ is the reconstructed version of the data obtained from the output layer. δ^I can be defined as a dissimilarity measure (e.g., Euclidean, cosine and etc.). After pretraining the Autoencoder over the whole data, the clustering loss shall be computed. In this algorithm, clustering is based on KL divergence between soft computed labels (Q) and the target distribution (P) obtained from Q. The soft labels can be computed as following:

$$q_{i,j} = \frac{(1 + ||f_\theta(x_i) - r_j||^2)^{-1}}{\sum_j (1 + ||f_\theta(x_i) - r_j||^2)^{-1}} \tag{3.2}$$

Where x_i is the i'th input and r_j is the j'th cluster representative. $q_{i,j}$ is defined as the similarity between the i'th embedded input and j'th cluster representative. As mentioned earlier, $p_{i,j}$ is computed based on $q_{i,j}$. Thus the target distribution is computed as follows:

$$p_{i,j} = \frac{q_{i,j}^2 / \sum_i q_{i,j}}{\sum_j (q_{i,j}^2 / \sum_i q_{i,j})} \tag{3.3}$$

Finally the clustering loss is defined as:

$$\mathcal{L}_{clus} = KL(P||Q) = \sum_i \sum_j p_{i,j} log \frac{p_{i,j}}{q_{i,j}} \tag{3.4}$$

Formula 3.5 represents the IDEC loss function. The IDEC loss (similar to major DC algorithms loss functions) is a combination of reconstruction loss and clustering loss. λ is a hyper-parameter which balances the emphasis between reconstruction loss and clustering loss.

$$\mathcal{L} = \mathcal{L}_{clus} + \mathcal{L}_{rec} \tag{3.5}$$

In IDEC, the labels can be obtained from the following formula:

$$s_i = \arg\max_j q_{i,j} \tag{3.6}$$

The whole IDEC algorithm is summarized in Algorithm 2.

DCN: This algorithm even shares more similarities with our proposed DC algorithm compared to IDEC. Generating a k-means friendly space is the highlight of the paper. Although in the paper it has been claimed that DCN is able to update Autoencoder weights and cluster representatives in a joint way, based on the proposed formulation, the learning process is performed by alternating between optimizing reconstruction loss and k-mean loss. The proposed objective function for DCN is similar to the one defined in Formula 3.1 (similar to IDEC). In this algorithm, the clustering loss is defined as:

$$\mathcal{L}_{clus} = \sum_{x \in \mathcal{X}} ||f_\theta(x) - Rs_{\mathbf{x}}||_2^2 \tag{3.7}$$

Algorithm 2: IDEC algorithm

Inputs: input data: $\mathcal{X}$; Number of clusters: K; Target distribution target interval: T; stopping threshold: δ; Maximum iterations: $MaxIter$
Outputs: Cluster representatives: R; Labels: S
for $iter \in 0, 1, ..., MaxIter$
  if $iter$ $\%T$==0
    Compute the embeddings for all points
    Compute P using Formula 3.3
    Save last label assignments: $s_{old} = s$
    Compute new label assignments based on Formula 3.6
    if $sum(s_{old} \neq s)/n < \delta$:
      Stop training
  Choose a minibatch
  Update Autoencoder parameters and cluster representatives

where R is the set of cluster representatives and $s_{\mathbf{x}}$ is assignment of data point x to one of the cluster representatives. Finally the loss can be defined as:

$$\mathcal{L} = \mathcal{L}_{clus} + \mathcal{L}_{rec} \tag{3.8}$$

In this algorithm first an Autoencoder is pretrained and sequentially the weights of the Autoencoder, cluster representatives $(r_k, k \in 1...K)$ and cluster assignments (S) are initialized. To optimize the loss function proposed in Formula 3.8, an alternative optimization technique is used. First the reconstruction loss (Formula 3.1) is optimized using back propagation. Then to optimize clustering loss (Formula 3.7), a new optimization algorithm is proposed. In this optimization algorithm, first cluster assignments are obtained as follows:

$$s_{x,j} = \begin{cases} 1, \text{if} j = \underset{k \in 1...K}{\arg\min} ||f_\theta(x) - r_k||^2 \\ 0, \text{otherwise} \end{cases} \tag{3.9}$$

To optimize cluster representatives (R), let us denote c_k^x as the number of times the algorithm assigned sample x to cluster k.

$$r_k = r_k - (\frac{1}{c_k^x})(r_k - f_\theta(x))s_{x,k} \tag{3.10}$$

Because of this alternative optimization scheme, DCN is not able to update cluster representatives and Autoencoder weights in a truly joint way. In the next chapter, we will discuss how our proposed DC framework is able to update cluster representatives and Autoencoder weights in a truly joint way which yields better performance and a space which is more k-means friendly compared to DCN.

Algorithm 3 summarizes DCN algorithm.

Algorithm 3: DCN algorithm

Inputs: input data: $\mathcal{X}$; Number of clusters: K; R: sets of cluster representatives, s: cluster assignments
Outputs: Cluster representatives: R; Labels: S
  for i in number of epochs:
    for j in number of batches:
      Withdraw a minibatch
      Compute reconstruction loss using Formula 3.1
      Update cluster assignments (S) using Formula 3.9
      Update cluster representatives (R) using Formula 3.10
      Compute clustering loss ($\mathcal{L}_{clus}$) using Formula 3.7
      Compute total loss ($\mathcal{L}$) using Formula 3.8

3.3 Constrained Clustering Approaches

Due to the subjectivity aspect of clustering, different users might impose different constraints on clustering. For instance, to cluster a set of news articles, one might be interested in having two clusters where one indicates liberal views and the other one democratic views, while another user might be interested in dividing documents into economy and environment topics. Constraints can take different forms such as *seed words* and *pairwise constraints*. Seed words correspond to sets of words which are able to characterize and define each cluster. For instance, 'player', 'coach', 'stadium' can be considered as descriptive seed words for a sports cluster while 'politician', 'congress', 'president' are able to describe politics.

Additionally, one might use pairwise constraints where a set of must-link and cannot-link documents are used as side information which can lead

towards better clustering results. As the names imply, must-link pairs contain documents coming from the same category while cannot-link pairs contain documents coming from different categories.

Following in this Chapter, we review some of the works in these domains. As this thesis is mainly focused on the document clustering, we concentrated on constraints in this domain while in other domains (e.g., image clustering) the nature of the constraints can be different [89].

3.3.1 Seed Words Based Approaches

The constrained clustering problem which includes seed words in fact bears strong similarity with the one of seed-guided dataless text classification, which consist in categorizing documents based on a small set of seed words describing the classes/clusters.

The task of dataless text classification was introduced independently in [90] and [91]. In [90], the seed words are provided by a user and exploited to automatically label a part of the unlabeled documents. On the other hand, in [91], seed words initially correspond to labels/titles for the classes of interest and are extended based on co-occurrence patterns. In both cases, a Naive Bayes classifier is applied to estimate the documents' class assignments. In the wake of these seminal works, several studies further investigated the exploitation of seed words for text classification [92–94]. Both an 'on-the-fly' approach and a bootstrapping approach by projecting seed words and documents in the same space were introduced in [92]. The former approach simply consists in assigning each document to the nearest class in the space, whereas the latter learns a bootstrapping Naive Bayes classifier with the class-informed seed words as initial training set. Another bootstrapping approach is studied in [94], where two different methods are considered to build the initial training set from the seed words: Latent Semantic Indexing and Gaussian Mixture Models. The maximum entropy classifier proposed in [93] instead directly uses seed words' class information by assuming that documents containing seed words from a class are more likely to belong to this class.

More recently, the dataless text classification problem was addressed through topic modeling approaches [95–98], extending the Latent Dirichlet Allocation model [99]. The topic model devised in [95] integrates the seed words as pseudo-documents, where each pseudo-document contains all the seed words given for a single class. The co-occurrence mechanism underlying topic models along with the known class membership of pseudo-documents help guide the actual documents to be classified towards their correct class.

In [96], the Seed-guided Topic Model (STM) distinguishes between two types of topics: category topics and general topics. The former describe the class information and are associated with an informed prior based on the seed words, whereas the latter correspond to the general topics underlying the whole collection. The category topics assigned to a document are then used to estimate its class assignment. STM was extended in [98] to simultaneously perform classification and document filtering – which consists in identifying the documents related to a given set of categories while discarding irrelevant documents – by further dividing category topics into relevant and non-relevant topics. Similarly to STM, the Laplacian Seed Word Topic Model (LapSWTM) introduced in [97] considers both category topics and general topics. It however differs from previous models in that it enforces a document manifold regularization to overcome the issue of documents containing no seed words. If these models outperform previously proposed models, they suffer from a lack of flexibility on the input representations they rely on. Indeed, topic models require documents to be organized as sets of discrete units – the word tokens. This prohibits the use of representation learning techniques such as word embeddings (e.g., word2vec [100] and GloVe [101]). Moreover, in these approaches, representation learning is not performed.

For a more general survey on constrained clustering, we invite the reader to refer to [102]

In this thesis, we discuss our proposed SD2C framework which is able to employ Autoencoders for representation learning. Indeed, SD2C is able to bias data representations based on the seed words provided by the users. Later in Chapter 5 we discuss our proposed framework in more details.

3.3.2 Must-link and Cannot-link Based Approaches

As mentioned earlier, additional information can help users to reflect their needs in clustering and pairwise constraints are one of them. So far, there have been a few studies in deep learning-based approached which address handling this type of constraints. For instance in [103] a deep clustering framework with a KL- divergence loss has been designed which is able to handle multiple types of constraints simultaenously. Also, there have been many works that utilized this information in non-deep clustering frameworks [104–109]. In [104], pairwise constraints have been used in a generative framework where Expected Maximization (EM) and Generalized EM procedures have been used to handle Must- link and Cannot-link constraints respectively. A boosting clustering algorithm named BoostCluster [106] is proposed to improve accuracy of any given clustering algorithms iteratively by using

pairwise constraints information. Maximum Margin Clustering (MMC) [110] which borrows from the theory of support vector machines [111] is coupled with pairwised constraints [107] to improve the accuracy of original MMC model. k-means objective function has been modified in [105] to utilize the pairwise constraints information. Although in all mentioned approaches an auxiliary term has been added to the clustering loss to improve its performance, in [108] and [109], using pairwise constraints is investigated in the domain of semi-supervised learning. Using pairwise constraints can be beneficial and lead toward better results, yet these approaches suffer from using the data in the original space and do not learn any new representations from the data.

Similar to seed words based approaches, in none of the above approaches, representation learning is not performed. Indeed, in Chapter 5 we introduce our PCD2C framework which is able to bias data representations based on the must-link and cannot-link constraints provided by users.

In the next Chapter we discuss our proposed DKM framework which is able to learn data representations and k-means cluster representatives in a truly joint way.

CHAPTER 4

Deep k-means Clustering

4.1 Introduction

In previous chapters we reviewed clustering and recent advances in DC. Although there are variety of DC algorithms, yet applicability of these algorithms for different types of data (e.g., image, text and etc.) is unknown. Indeed, most of the proposed approaches in DC are specifically designed for one type of data (mostly images). In this section we will propose our Deep k-means (DKM) approach. In fact, first, we study the problem of jointly clustering and learning representations. As several previous studies have shown, learning representations that are both faithful to the data to be clustered and adapted to the clustering algorithm can lead to better clustering performance, all the more so that the two tasks are performed jointly. We propose here such an approach for k-Means clustering based on a continuous reparametrization of the objective function that leads to a truly joint solution. Indeed, we have shown that one can solve k-means non-differentiability problem simply by using a softmax function instead of *argmin*. The behavior of our approach is illustrated on various datasets showing its efficacy in learning representations for objects while clustering them. The proposed DKM approach has been tested on variety types of data including image and text data. The obtained results show generalizability of our proposed approach. Later in this chapter, we provide more details on our proposed DKM framework.

4.2 Deep k-means

As mentioned earlier, due to non-differentiability of k-means objective function, jointly learning data representations and cluster representatives is not feasible. Indeed, in order to employ k-means in a DC framework, proposing a new k-means objective function which allows the gradients to be computed is necessary. In DCN [29], through an alternative updating of cluster representatives and data representations, it has been tried to address k-means non-differentiability problem. On the contrary, our proposed framework benefits from jointly updating cluster representatives and data representations. In this section we introduce our proposed DKM framework such that all network and clustering parameters are updated simultaneously.

In the remainder, x denotes an object from a set $\mathcal{X}$ of objects to be clustered. K will denote the number of clusters to be obtained, $\mathbf{r}_k \in \mathbb{R}^p$ the representative of cluster k, $1 \leq k \leq K$, and $\mathcal{R} = \{\mathbf{r}_1, \ldots, \mathbf{r}_K\}$ the set of representatives. We use the term *representative* rather than *centroid* here to emphasize the fact that in case of similarity functions, or dissimilarity functions different from the Euclidean distance (e.g., cosine distance), the representative of a cluster does not necessarily coincide with its centroid. δ^I and δ^E will denote similarity or dissimilarity functions in $\mathbb{R}^p$; for any vector $\mathbf{y} \in \mathbb{R}^p$, $c_{\delta^E}(\mathbf{y}; \mathcal{R})$ will denote the closest representative of $\mathbf{y}$ according to δ^E.

The deep k-Means problem takes the form:

$$\begin{aligned} &\min_{\mathcal{R},\theta,\eta} \sum_{x \in \mathcal{X}} \delta^I(g_\eta \circ f_\theta(x), x) + \lambda\, \delta^E(f_\theta(x), c_{\delta^E}(f_\theta(x); \mathcal{R})), \\ &\text{with: } c_{\delta^E}(f_\theta(x); \mathcal{R}) = \arg\min_{\mathbf{r} \in \mathcal{R}} \delta^E(f_\theta(x), \mathbf{r}) \end{aligned} \tag{4.1}$$

$\delta^I(g_\eta \circ f_\theta(x), x)$ is the reconstruction loss function that measures the error between an object x and its reconstruction by the Autoencoder $g_\eta \circ f_\theta(\mathcal{X})$. θ and η represent, as before, the set of the parameters of the Autoencoder. Figure 4.1 provides an overview of the proposed Deep k-Means approach. However, as most Autoencoders do not use regularization, we dispense with such a term here. $\delta^E(f_\theta(x), c_{\delta^E}(f_\theta(x); \mathcal{R}))$ is the clustering loss corresponding to the generalized k-Means objective function in the embedding space. Finally, λ in Formula 4.1 regulates the trade-off between seeking good representations for x – *i.e.*, representations that are faithful to the original examples – and representations that are useful for clustering purposes.

Standard k-Means is obtained by setting δ^E to the Euclidean distance, whereas generalized k-Means with the cosine similarity is obtained by setting δ^E to the cosine function. δ^I can also be set to Euclidean or cosine. In the remainder, we focus on functions δ^I that are differentiable wrt θ and η and functions δ^E that are differentiable wrt to θ, η and $\mathcal{R}$, where differentiability wrt $\mathcal{R}$ means differentiability wrt to all dimensions of $\mathbf{r}_k$, $1 \leq k \leq K$.

We now introduce a parametrized version of the above problem that constitutes a *continuous generalization*, whereby we mean here that all functions considered are continuous wrt the introduced parameter. To do so, we first note that the clustering objective function can be rewritten as:

$$\delta^E(\mathbf{f}_\theta(x), c_{\delta^E}(\mathbf{f}_\theta(x); \mathcal{R})) = \sum_{k=1}^{K} \delta_k^E(\mathbf{f}_\theta(x); \mathcal{R})$$

with:

$$\delta_k^E(\mathbf{f}_\theta(x); \mathcal{R}) = \begin{cases} f(\mathbf{f}_\theta(x), \mathbf{r}_k) \text{ if } \mathbf{r}_k = c_{\delta^E}(\mathbf{f}_\theta(x); \mathcal{R}) \\ 0 \text{ otherwise} \end{cases}$$

Let us now assume that we know some function $G_k(\mathbf{f}_\theta(x), \alpha; \mathcal{R})$ such that:

(i) G_k is differentiable wrt to $\theta, \mathcal{R}$ and continuous wrt α (differentiability wrt $\mathcal{R}$ has the meaning as before);

(ii) $\exists \alpha_0 \in \mathbb{R} \cup \{-\infty, +\infty\}$ such that:

$$\lim_{\alpha \to \alpha_0} G_k(\mathbf{f}_\theta(x), \alpha; \mathcal{R}) = \begin{cases} 1 \text{ if } \mathbf{r}_k = c_{\delta^E}(\mathbf{f}_\theta(x); \mathcal{R}) \\ 0 \text{ otherwise} \end{cases}$$

Then, one has:

Lemma 4.2.1. *$\forall x \in \mathcal{X}$:*

$$\lim_{\alpha \to \alpha_0} \delta^E(\mathbf{f}_\theta(x), \mathbf{r}_k)\, G_k(\mathbf{f}_\theta(x), \alpha; \mathcal{R}) = \delta_k^E(\mathbf{f}_\theta(x); \mathcal{R})$$

The proof directly derives from the definitions of δ_k^E and G_k.

From Lemma 4.2.1, one can see that:

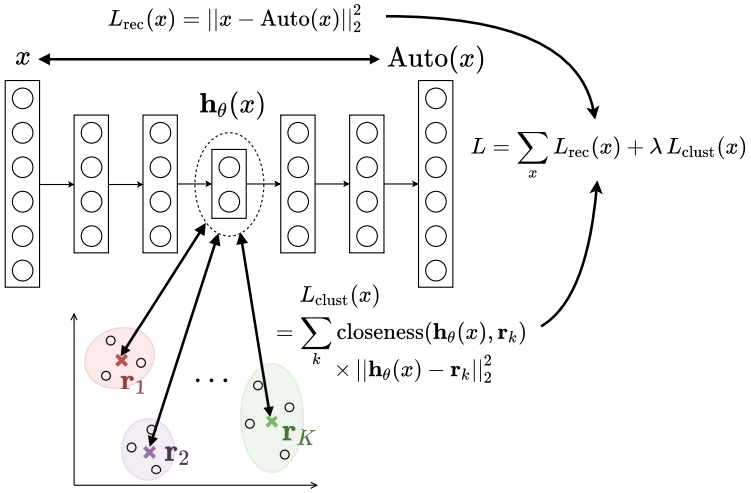


Figure 4.1: Overview of our Deep k-Means approach instantiated with losses based on the Euclidean distance.

Property 4.2.1. *The problem given in (4.1) is equivalent to:*

$$\min_{\mathcal{R},\theta} \lim_{\alpha \to \alpha_0} \mathcal{F}(\mathcal{X}, \alpha; \theta, \mathcal{R}),$$

$$\text{with: } \overbrace{\mathcal{F}(\mathcal{X}, \alpha; \theta, \mathcal{R}) = \sum_{x \in \mathcal{X}} \delta^I(g_\eta \circ f_\theta(x), x) + \lambda \sum_{k=1}^{K} \delta^E(\mathbf{f}_\theta(x), \mathbf{r}_k)\, G_k(\mathbf{f}_\theta(x), \alpha; \mathcal{R})}^{\mathcal{F}(\mathcal{X}, \alpha; \theta, \mathcal{R})} \tag{4.2}$$

All functions in the above formulation are fully differentiable wrt both θ and $\mathcal{R}$. One can thus estimate θ and $\mathcal{R}$ through a simple, joint optimization based on stochastic gradient descent (SGD) for a given α:

$$(\theta,\, \mathcal{R}) \leftarrow (\theta,\, \mathcal{R}) - \eta\, \frac{1}{|\tilde{\mathcal{X}}|}\, \nabla_{(\theta, \mathcal{R})} \mathcal{F}(\tilde{\mathcal{X}}, \alpha; \theta, \mathcal{R}) \tag{4.3}$$

with η the learning rate and $\tilde{\mathcal{X}}$ a random mini-batch of $\mathcal{X}$ and $\nabla_{(\theta, \mathcal{R})}$ the gradients with respect to θ and $\mathcal{R}$.

4.2.1 Choice of G_k

Several choices are obviously possible for G_k. A simple choice, used throughout this study, is based on a parametrized softmax function. The fact that the softmax function can be used as a differentiable approximation of arg max or arg min is well known and has been applied in different contexts, as in the

recently proposed Gumbel-softmax distribution employed to approximate categorical samples [112, 113]. We introduce a parametrized version here to better control the approximation and rely on a deterministic annealing scheme (see Section 4.2.2). The parametrized softmax function which we adopted takes the following form:

$$G_k(\mathbf{h}_\theta(x), \alpha; \mathcal{R}) = \frac{e^{-\alpha\delta^E(\mathbf{f}_\theta(x), \mathbf{r}_k)}}{\sum_{k'=1}^{K} e^{-\alpha\delta^E(\mathbf{f}_\theta(x), \mathbf{r}_{k'})}} \tag{4.4}$$

with $\alpha \in [0, +\infty)$. The function G_k defined by Eq. 4.4 is differentiable wrt $\theta, \mathcal{R}$ and α (condition *(i)*) as it is a composition of functions differentiable wrt these parameters. Furthermore, one has:

Property 4.2.2. *(condition (ii))*

Assuming that $c_{\delta^E}(\mathbf{h}_\theta(x); \mathcal{R})$ is unique for all $x \in \mathcal{X}$,

$$\lim_{\alpha \to +\infty} G_k(\mathbf{f}_\theta(x), \alpha; \mathcal{R}) = \begin{cases} 1 \text{ if } \mathbf{r}_k = c_{\delta^E}(\mathbf{f}_\theta(x); \mathcal{R}) \\ 0 \text{ otherwise} \end{cases}$$

Proof: Let $\mathbf{r}_j = c_{\delta^E}(\mathbf{h}_\theta(x); \mathcal{R})$ and let assume that δ^E is a dissimilarity function. One has:

$$\sum_{k'=1}^{K} e^{-\alpha\delta^E(\mathbf{f}_\theta(x), \mathbf{r}_{k'})} = e^{-\alpha\delta^E(\mathbf{f}_\theta(x), \mathbf{r}_j)} \times \left(1 + \sum_{k' \neq j} e^{-\alpha(\delta^E(\mathbf{f}_\theta(x), \mathbf{r}_{k'}) - \delta^E(\mathbf{f}_\theta(x), \mathbf{r}_j))}\right)$$

As $\delta^E(\mathbf{f}_\theta(x), \mathbf{r}_j) < \delta^E(\mathbf{f}_\theta(x), \mathbf{r}_{k'})$, $\forall k' \neq j$, one has:

$$\lim_{\alpha \to +\infty} e^{-\alpha(\delta^E(\mathbf{f}_\theta(x), \mathbf{r}_{k'}) - \delta^E(\mathbf{f}_\theta(x), \mathbf{r}_j))} = 0$$

Thus $\lim_{\alpha \to +\infty} G_k(\mathbf{f}_\theta(x), \alpha; \mathcal{R}) = 0$ if $k \neq j$ and 1 if $k = j$. The case where δ^E is a similarity function can be treated in the same way.

The assumption that $\delta^E(\mathbf{f}_\theta(x); \mathcal{R})$ is unique for all objects is necessary for G_k to take on binary values in the limit; it is not necessary to hold for small values of α. In fact, when α is set to 0, then the optimization problem given in (4.2) has a unique solution obtained by setting all representatives to the (single) vector that optimizes $\sum_{x \in \mathcal{X}} f(\mathbf{f}_\theta(x), \mathbf{r})$ (this vector corresponds to the centroid of all embedded representations when δ^E is the Euclidean distance). In the unlikely event that the above assumption does not hold for some x and large values of α, then one can slightly perturbate the representatives equidistant to x prior to updating them. We have never encountered this situation in practice.

Finally, under the uniqueness assumption above, Eq. 4.4 defines a valid (according to conditions (i) and (ii)) function G_k that can be used to solve the optimization problem given in (4.2). We adopt this function in the remainder of this study.

Prior to studying the effect of α in Eq. 4.4, we want to mention another possible choice for G_k. As $G_k(\mathbf{f}_\theta(x), \alpha; \mathcal{R})$ plays the role of a closeness function for object x wrt representative $\mathbf{r}_k$, membership functions used in fuzzy clustering are potential candidates for G_k. In particular, the membership function of the fuzzy C-Means algorithm [10] is a valid candidate according to conditions (i) and (ii). It takes the form, for dissimilarity functions:

$$G_k(\mathbf{f}_\theta(x), \alpha; \mathcal{R}) = \left(\sum_{k'=1}^{K} \left(\frac{\delta^E(\mathbf{f}_\theta(x), \mathbf{r}_k)}{\delta^E(\mathbf{f}_\theta(x), \mathbf{r}_{k'})} \right)^{\frac{2}{\alpha-1}} \right)^{-1}$$

with α defined on $[1; +\infty]$ and α_0 (condition (ii)) equal to 1. However, in addition to being slightly more complex than the parametrized softmax, this formulation presents the disadvantage that it may be undefined when a representative coincides with an object; another assumption (in addition to the uniqueness assumption) is required here to avoid such a case.

4.2.2 Choice of α

The parameter α can be defined in different ways. Indeed, α can play the role of an inverse temperature such that, when α is 0, each data point in the embedding space is equally close, through G_k, to all the representatives (corresponding to a completely soft assignment), whereas when α is $+\infty$, the assignment is hard. In the first case, for the deep k-Means optimization problem, all representatives are equal and set to the point $\mathbf{r} \in \mathbb{R}^p$ that minimizes $\sum_{x \in \mathcal{X}} f(\mathbf{f}_\theta(x), \mathbf{r})$. In the second case, the solution corresponds to exactly performing k-Means in the embedding space, the latter being learned jointly with the clustering process. Following a deterministic annealing approach [114], one can start with a low value of α (close to 0), and gradually increase it till a sufficiently large value is obtained. At first, representatives are randomly initialized. As the problem is smooth when α is close to 0, different initializations are likely to lead to the same local minimum in the first iteration; this local minimum is used for the new values of the representatives for the second iteration, and so on. The continuity of G_k wrt α implies that, provided the increment in α is not too important, one evolves smoothly from the initial local minimum to the last one.

In the above deterministic annealing scheme, α allows one to initialize

Algorithm 4: Deep k-Means algorithm

Input: data $\mathcal{X}$, number of clusters K, balancing parameter λ, scheme for α, number of epochs T, number of minibatches N, learning rate η

Output: autoencoder parameters θ (encoder parameter) and η (decoder parameter), cluster representatives $\mathcal{R}$

```
Initialize θ and r_k, 1 ≤ k ≤ K (randomly or through pretraining)
for α = m_α to M_α do                          ▷ inverse temperature levels
    for t = 1 to T do                          ▷ epochs per α
        for n = 1 to N do                      ▷ minibatches
            Draw a minibatch X̃ ⊂ X
            Update (θ, R) using SGD (Eq. 4.3)
        end
    end
end
```

cluster representatives. The initialization of cluster representatives can have an important impact on the results obtained and prior studies (*e.g.*, [29–31, 85]) have relied on pretraining for this matter. In such a case, one can choose a high value for α to directly obtain the behavior of the k-Means algorithm in the embedding space. We evaluate both approaches in our experiments (Section 4.3).

Algorithm 4 summarizes the deep k-Means algorithm for the deterministic annealing scheme, where m_α (respectively M_α) denote the minimum (respectively maximum) value of α, and T is the number of epochs per each value of α for the stochastic gradient updates. Even though M_α is finite, it can be set sufficiently large to obtain in practice a hard assignment to representatives. Lastly, when using pretraining, one sets $m_\alpha = M_\alpha$ (*i.e.*, a constant α is used).

4.3 Experiments

In order to evaluate our proposed DKM framework and other clustering and DC approaches, we conducted massive experiments on different datasets. Following this section we discuss the details of the experiments including the datasets that have been used for evaluation, experimental setups, etc.

4.3.1 Datasets

Different kinds of data including text and image have been used to evaluate different clustering and DC approaches. These datasets are detailed in the following.

Image datasets

The datasets that are used in the experiments are standard clustering benchmark collections. We considered both image and text datasets to demonstrate the general applicability of our proposed approach. Image datasets consist of **MNIST** (70,000 images, 28×28 pixels, 10 classes) and **USPS** (9,298 images, 16×16 pixels, 10 classes) which both contain hand-written digit images. We reshaped the images to one-dimensional vectors and normalized the pixel intensity levels (between 0 and 1 for MNIST, and between -1 and 1 for USPS). Below more detailed information can be found about these image data.

MNIST dataset

MNIST data is one of the most widely used datasets in the field of image clustering which contains handwritten digits. This dataset has been also used for training and testing multiple supervised learning algorithms as well. MNIST is a combination of another dataset which is called *NIST*. Train and test sets of NIST have been obtained by sampling from two different sources: 1) American Census Bureau and 2) American high school students. Although in unsupervised approaches the whole MNIST data (70,000 samples) are being used, originally the MNIST data includes 60,000 training and 10,000 test samples. Half of the training set and half of the test set were obtained from NIST's training samples while the other half is obtained from the NIST's test samples. Figure 4.2 illustrates an example of MNIST dataset.

Figure 4.2: Examples from the MNIST dataset.

USPS dataset

Similar to MNIST, this dataset contains handwritten digits (0-9). The samples of this dataset have been collected from U.S Post Office. The

envelopes were scanned and digits were presented as binary values of different sizes and orientations. AT & T Research Lab (in collaboration with Yann Le Cun) made this dataset available for research purposes. This dataset is imbalanced and the test set has been considered as difficult to discriminate (an error rate of 2.5% has been considered notoriously good). USPS includes 7291 training and 2007 test samples. Table 4.1 provides more detailed information about this dataset.

Table 4.1: Detailed information about each class of USPS dataset.

Set	0	1	2	3	4	5	6	7	8	9	Total
Training	1194	1005	731	658	652	556	664	645	542	644	7291
Test	359	264	198	166	200	160	170	147	166	177	2007
Training+Test	1553	1269	929	824	852	716	834	801	708	821	9298

Text datasets

The text collections that were considered for the experiments include: 20 Newsgroups dataset (hereafter, **20NEWS**), RCV1-v2 dataset (hereafter, **RCV1**), REUTERS, YAHOO, DBPEDIA and AGNEWS datasets. We used two different word representations methods in our experiments: 1) TF-IDF [115] and 2) Word2Vec [100].

In case of using TF-IDF, for 20NEWS, we used the whole dataset comprising of 18,846 documents labeled into 20 different classes. Similarly to [31,85], we sampled from the full RCV1-v2 collection a random subset of 10,000 documents, each of which pertains to only one of the four largest classes. Because of the text datasets' sparsity, and as proposed in [85], we selected the 2000 words with the highest tf-idf values to represent each document.

In case of using Word2Vec, REUTERS, 20NEWS, YAHOO, DBPEDIA and AGNEWS are selected for the experiments. We have trained a Word2Vec model for each collection (after applying stemming). Skip-gram version of Word2Vec with a window size of 50 for 20NEWS and 10 for the rest of the datasets has been used. The embedding size is 100 for all collections. Other word representation approaches such as pretrained Word2Vec, FastText [116] (both pretrained and non-pretrained) and Doc2Vec [117] have been used as well. Among all word representation techniques, training the Word2Vec seemed to obtain better results.

Please note that for all word representations techniques, the hyperparameters which are not mentioned here have been set to their default values provided by the Gensim[1] library in Python.

[1] https://radimrehurek.com/gensim/models/word2vec.html

20NEWS dataset

As the name implies, this dataset contains documents about news pertaining to different topics. 20NEWS consists of almost 20,000 documents categorized almost evenly into 20 different topics. For some topics it can be difficult to discriminate them from each other since the contents can be quite similar. For instance, *comp.sys.ibm.pc.hardware* and *comp.sys.mac.hardware* both cover computer hardware documents while other topics can be discriminated in an easier way. For instance, *misc.forsale* and *soc.religion.christian* represent completely different topics. Table 4.2 illustrates additional information about different topics in the 20NEWS dataset. Similar topics are grouped in the same cell.

Table 4.2: Different topics in 20NEWS data. Similar topics are grouped in a shared cell.

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

REUTERS dataset

REUTERS is one of the most important and widely used text datasets in the domain of text clustering. This dataset has been collected by Carnegie Group, Inc. and Reuters, Ltd. Similar to other approaches [118–121] we have used only the 10 largest (and highly imbalanced) categories. There are different versions of REUTERS but the version that we used (REUTERS-21578) is the most recent one. Table 4.3 includes detailed information about number of documents in each category. The information presented in Table 4.3 indicate a highly imbalanced dataset.

Table 4.3: Detailed information of REUTERS dataset .

	acq	coffee	crude	earn	gold	interest	money-fx	ship	sugar	trade	Total
#documents	2292	112	374	3923	90	272	309	144	122	326	7964
Percentage(%)	28.7	1.4	4.6	49.2	1.1	3.4	3.8	1.8	1.5	4.0	∼100

YAHOO dataset

YAHOO dataset was introduced in [122]. The authors obtained Yahoo! Comprehensive Questions and Answers (version 1.0) through the Yahoo! Webscope program. The corpus contains 4,483,032 questions and corresponding answers. The authors constructed a topic classification dataset from this corpus using 10 largest categories. We use only the test set and the whole 10 categories which results in 60,000 documents which are uniformly distributed over all 10 categories (6,000 documents for each category).

DBPEDIA dataset

DBPEDIA dataset is also introduced in [122]. DBPEDIA is a crowd-sourced community effort to extract information from Wikipedia. DBPEDIA dataset is formed by selecting 14 non-overlapping classes from DBPEDIA. In our experiments, we only use the test set made of 70,000 documents uniformly distributed in 14 classes (5,000 documents per category).

AGNEWS dataset

AGNEWS data is a collection of news article available on the web. We have used 4 largest classes of the training set of this dataset resulting in 120,000 documents evenly distributed in 4 categories (30,000 documents per category).

Table 4.4: Statistics of the datasets.

Dataset	#Samples	#Classes	Dimensions
MNIST	70,000	10	28×28
USPS	9,298	10	16×16
20NEWS(TF-IDF)	18,846	20	2,000
RCV1	10,000	4	2,000
REUTERS	10,000	10	100
20NEWS(Word2Vec)	18,846	20	100
YAHOO	60,000	10	100
DBPEDIA	70,000	10	100
AGNEWS	120,000	4	100

4.3.2 Baselines and Deep k-Means variants

Clustering models may use different strategies and different clustering losses, leading to different properties. As our goal in this work is to study the k-Means clustering algorithm in embedding spaces, we focus on the family of k-Means-related models and compare our approach against state-of-the-art models from this family, using both standard and deep clustering models. For the standard clustering methods, we used: the k-Means clustering approach [1] with initial cluster center selection [6], denoted **KM**; an approach denoted as **AE-KM** in which dimensionality reduction is first performed using an auto-encoder followed by k-Means applied to the learned representations. For the joint deep clustering models, the only previous, "true" deep clustering k-Means-related method is the Deep Clustering Network (DCN) approach described in [29]. In addition, we consider here the Improved Deep Embedded Clustering (IDEC) model [31] as an additional baseline. IDEC is, to the best of our knowledge, the state-of-the-art approach in the centroid-based – therefore related to k-Means – deep clustering family. As mentioned in Chapter 3, IDEC is an improved version of the DEC model [85]. For both of these approaches (DCN and IDEC), we studied two variants: with pretraining (**DCN$^{\mathbf{p}}$** and **IDEC$^{\mathbf{p}}$**) and without pretraining (**DCN$^{\mathbf{np}}$** and **IDEC$^{\mathbf{np}}$**). The pretraining we performed here simply consists in initializing the weights by training the auto-encoder on the data to minimize the reconstruction loss in an end-to-end fashion – we did not use layer-wise pretraining [123].

The proposed Deep k-Means (DKM) is, as DCN, a "true" k-Means approach in the embedding space; it jointly learns Autoencoder-based representations and relaxes the k-Means problem by introducing a parameterized softmax as a differentiable surrogate to k-Means argmin. In the experiments, we considered two variants of this approach. **DKM$^{\mathbf{a}}$** implements an annealing strategy for the inverse temperature α and does not rely on pretraining. The scheme we used for the evolution of the inverse temperature α in DKM$^{\mathrm{a}}$ is given by the following recursive sequence: $\alpha_{n+1} = 2^{1/\log(n)^2} \times \alpha_n$ with $m_\alpha = \alpha_1 = 0.1$. The 40 first terms of (α_n) are plotted in Figure 4.3. The rationale behind the choice of this scheme is that we want α to spend more iterations on smaller values and less on larger values while preserving a gentle slope. Alternatively, we studied the variant **DKM$^{\mathbf{p}}$** which is initialized by pretraining an auto-encoder and then follows Algorithm 4 with a constant α such that $m_\alpha = M_\alpha = 1000$. Such

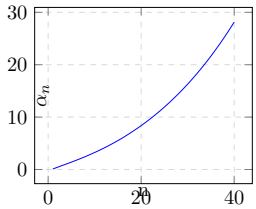


Figure 4.3: Annealing scheme for inverse temperature α, following the sequence $\alpha_{n+1} = 2^{1/\log(n)^2} \times \alpha_n$; $\alpha_1 = 0.1$.

a high α is equivalent to having hard cluster assignments while maintaining the differentiability of the optimization problem.

Implementation details
For IDEC, we used the Keras code shared by their authors.[2]. We used this version instead of https://github.com/XifengGuo/IDEC as only the former enables auto-encoder pretraining in a non-layer-wise fashion. Our own code for DKM is based on TensorFlow. To enable full control of the comparison between DCN and DKM – DCN being the closest competitor to DKM – we also re-implemented DCN in TensorFlow. The code for both DKM and DCN is available online.[3]

4.3.3 Experimental setup

AE description and training details
The auto-encoder we used in the experiments is the same across all datasets and is borrowed from previous deep clustering studies [31, 85]. Its encoder is a fully-connected multilayer perceptron with dimensions d-500-500-2000-d where d is the original data space dimension. The decoder is a mirrored version of the encoder. All layers except the one preceding the embedding layer and the one preceding the output layer are applied a ReLU activation function [124] before being fed to the next layer. For the sake of simplicity, we did not rely on any complementary training or regularization strategies such as batch normalization or dropout. The auto-encoder weights are initialized following the Xavier scheme [125]. For all deep clustering approaches, the training is based on the Adam optimizer [126] with standard learning rate $\eta = 0.001$ and momentum rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The minibatch size is set to 256 on all datasets following [31]. We emphasize that we chose exactly the same training configuration for all models to facilitate a fair comparison.

The number of pretraining epochs is set to 50 for all models relying on pretraining. The number of fine-tuning epochs for DCN$^{\text{p}}$ and IDEC$^{\text{p}}$ is fixed to 50 (or equivalently in terms of iterations: 50 times the number of minibatches). We set the number of training epochs for DCN$^{\text{np}}$ and IDEC$^{\text{np}}$ to 50 and 200, respectively – in the case of DCN$^{\text{np}}$, we observed little to no improvement in successive epochs and therefore limited its number of epochs. For DKM$^{\text{a}}$, we used the 40 terms of the sequence α given in Figure 4.3 as the annealing scheme and performed 5 epochs for each α term. DKM$^{\text{p}}$ is

[2] https://github.com/XifengGuo/IDEC-toy
[3] https://github.com/MaziarMF/deep-k-means

fine-tuned by performing 100 epochs with constant $\alpha = 1000$. The cluster representatives are initialized randomly from a uniform distribution $U(-1, 1)$ for models without pretraining. In case of pretraining, the cluster representatives are initialized by applying k-Means to the pretrained embedding space. Fig 4.4 illustrates the encoder part of auto-encoder. As mentioned before, the decoder is simply the mirror of the encoder.

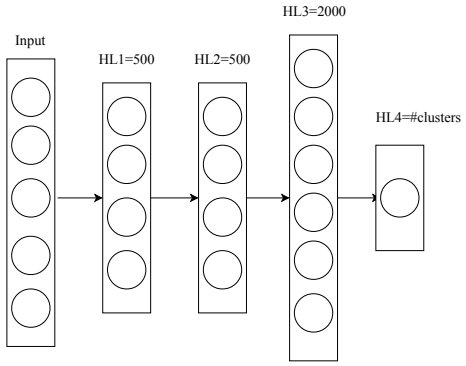


Figure 4.4: The architecture of the encoder part of the Autoencoder.

Hyperparameter selection

The hyperparameters λ for DCN and DKM and γ for IDEC, that define the trade-off between the reconstruction and the clustering error in the loss function, were determined by performing a line search on the set $\{10^i \mid i \in [-5, 3]\}$. We can select the hyper-parameters in two different ways: 1) tune the hyper-parameters for each collection and select the value corresponding to the best accuracy value (one can choose NMI or ARI to select the best hyper-parameter) or 2) use a general hyper-parameter for each approach (DKM, DCN and IDEC) which is the same for all datasets (not tuning). Below we have introduced these ways of selecting hyper-parameters in details.

Optimal λ and γ

To do so, we randomly split each dataset into a validation set (10% of the data) and a test set (90% of the dataset). Each model is trained on the whole data and only the validation set labels are leveraged in the line search to identify the optimal λ or γ (optimality is measured with respect to the clustering accuracy metric). The performance of different approaches by choosing optimal λ and γ are reported in Tables 4.6 and 4.7.

General λ and γ

One can also set a fixed value for λ and γ on all datasets. To do so, for each

approach, the result of hyperparameters are computed over all collections. Afterwards, the hyperparameter which obtains equally good results for all collection has been selected as the general hyperparameter. Indeed, for each hyperparameter, the average of the clustering accuracy of all datasets shall be computed. Afterwards, the hyperparameter which obtains the highest clustering accuracy (averaged on all datasets) will be selected as the general hyperparameter. The results corresponding general hyperparameters are reported in Table 4.8

Experimental protocol
We observed in pilot experiments that the clustering performance of the different models is subject to non-negligible variance from one run to another. This variance is due to the randomness in the initialization and in the minibatch sampling for the stochastic optimizer. When pretraining is used, the variance of the general pretraining phase and that of the model-specific fine-tuning phase add up, which makes it difficult to draw any confident conclusion about the clustering ability of a model. To alleviate this issue, we compared the different approaches using seeded runs whenever this was possible. This has the advantage of removing the variance of pretraining as seeds guarantee exactly the same results at the end of pretraining (since the same pretraining is performed for the different models). Additionally, it ensures that the same sequence of minibatches will be sampled. In practice, we used seeds for the models implemented in TensorFlow (KM, AE-KM, DCN and DKM). Because of implementation differences, seeds could not give the same pretraining states in the Keras-based IDEC. All in all, we randomly selected 10 seeds and for each model performed one run per seed. Additionally, to account for the remaining variance and to report statistical significance, we performed a Student's t-test from the 10 collected samples (i.e., runs).

Statistics and optimal hyperparameters We summarize in Table 4.4 the statistics of the different datasets used in the experiments, as well as the dataset-specific optimal values of the hyperparameter (λ for DKM-based and DCN-based methods and γ for IDEC-based ones) which trades off between the reconstruction loss and the clustering loss.

4.3.4 Clustering results

The results for the evaluation of the compared clustering methods on the different benchmark datasets are summarized in Tables 4.6, 4.7 and 4.8.

Table 4.5: dataset-specific optimal values of the hyperparameter (λ for DKM-based and DCN-based methods and γ for IDEC-based ones) which trades off between the reconstruction loss and the clustering loss

Dataset	$\lambda_{\text{DKM}^{\text{a}}}$	$\lambda_{\text{DKM}^{\text{p}}}$	$\lambda_{\text{DCN}^{\text{np}}}$	$\lambda_{\text{DCN}^{\text{p}}}$	$\gamma_{\text{IDEC}^{\text{np}}}$	$\gamma_{\text{IDEC}^{\text{p}}}$
MNIST	10^{-1}	1	10^{-2}	10	10^{-3}	10^{-2}
USPS	10^{-1}	1	10^{-1}	10^{-1}	10^{-1}	10^{3}
20NEWS (TF-IDF)	10^{-4}	10^{-1}	10^{-4}	10^{-1}	10^{-3}	10^{-1}
RCV1	10^{-4}	10^{-2}	10^{-3}	10^{-1}	10^{-4}	10^{-3}
REUTERS	N/A	1	N/A	1	N/A	10^{-1}
20NEWS(Word2Vec)	N/A	10^{-5}	N/A	10^{-3}	N/A	10^{0}
YAHOO	N/A	10^{-5}	N/A	10^{-4}	N/A	10^{-4}
DBPEDIA	N/A	10^{-1}	N/A	10^{-5}	N/A	10^{-3}
AGNEWS	N/A	10^{-5}	N/A	10^{-5}	N/A	10^{-3}

The clustering performance is evaluated with respect to three standard measures [127]: Normalized Mutual Information (NMI), Adjusted Rand Index (ARI) and the clustering accuracy (ACC). We report for each dataset/method pair the average and standard deviation of these metrics computed over 10 runs and conduct significance testing as described in Section 4.3.3. The bolded result in each column of all tables corresponds to the best result for the corresponding dataset/metric. Underlined results can be considered as equivalent to the best as their difference thereto is not statistically significant ($p > 0.05$).

Results on Image Data

The results of all approaches are illustrated in Table 4.6. Clearly, using auto-encoder helped K-means to achieve better results by obtaining new representations for data. This improvement is observable on all datasets and for all metrics. Thus, the basic intuition that using auto-encoders results in better data representations is confirmed.

DKM^a achieves far better results compared to non-pretrained versions of IDEC and DCN on all datasets and for all metrics. The considerable difference between results of DKM^a and non-pretrained versions of IDEC and DCN indicates that having an annealing scheme which starts from soft clustering assignment and turn into a hard clustering assignment is totally advantageous compared to simply start by initializing randomly weights and biases of the network and repeat the training.

IDEC^p achieves best results on MNIST and DKM^p obtains best results on USPS data. Also by taking a glance at the results we can notice that generally pretrained approaches are improving in terms of results compared to non-pretrained ones. This indicates that starting from a local optimal

point helps all algorithms to achieve better results compared to launching the algorithms with random initialization. Also the difference between results of DKM^p and DKM^a indicates that pretraining helps DKM to achieve better results compared to using annealing scheme. Another important aspect of the result is that DKM^p performed better than DCN^a on all image datasets.

Text data with TF-IDF representation
The results of different clustering approaches on 20NEWS and RCV1 are presented in Table 4.7. On 20NEWS, again we can notice that the results of AE-KM are higher than KM. DKM^a outperforms all non-pretrained approaches but its results are still lower than AE-KM. by comparing the pretrained versions vs the rest, we can notice:

- The results of $IDEC^p$ on 20NEWS are worse than AE-KM.
- The results of DCN^p are equal to AE-KM.
- The DKM^p outperforms other approaches.

For the RCV1 dataset, we can observe that again AE-KM achieves better results compared to KM. Among both non-pretrained and pretrained versions, IDEC performs better than the rest.

Text data with Word2Vec representation
As mentioned earlier, since the results of pretrained versions surpassed the results of non-pretrained versions, we only use pretrained versions for the rest of the thesis. In case of using general hyper-parameters (without tuning), we can note that both versions of DKM^p (EC) and DKM^p (CC) achieve better results compared to other approaches in terms of Macro-average results (average results of one approach over all datasets) of ACC and ARI while for NMI AE-KM and DKM^p achieve better results. On 20NEWS data, both variants of DKM^p obtain the best results for ACC, ARI and NMI. On REUTERS, DCN^p variants seem to obtain promising results on ACC and ARI while surprisingly KM obtains better NMI. On YAHOO, DCN^p (CC) results on ACC and NMI are superior to others while DKM^p (EC) is superior in terms of ARI. On DBPEDIA, DCN^p (EC) achieves state-of-the-art results in terms of all metrics. DCN^p (CC) shows promising results on AGNEWS.

The results of different approaches with tuned hyper-parameters are presented in Table 4.9. Similar to general hyper-parameters results, the Macro-average results indicate that for ACC and ARI, DKM^p(CC) achieves state-of-the-art results while for NMI, DKM^p(EC) and DCN^p(EC) share the

same best results. On 20NEWS, DKM^p(CC) obtains remarkable results for all metrics. On REUTERS, DKM^p(EC) results are superior in terms of ACC and ARI while similar to general hyper-parameters results, AE-KM obtains better results on NMI. On YAHOO, the best results for ACC, ARI and NMI belongs to DCN^p(CC), DKM^p(EC), DCN^p(CC). On DBPEDIA, DCN^p(EC) achieves state-of-the-art results for all metrics while DCN^p(CC) gets the best results on AGNEWS.

Table 4.6: Clustering results of the compared methods on image data. Performance is measured in terms of Accuracy, NMI and ARI (%); higher is better. Each cell contains the average and standard deviation computed over 10 runs. The best result for each dataset/metric is Bolded. Underlined values correspond to results with no significant difference ($p > 0.05$) to the best.

Model	MNIST			USPS			Macro-Average		
	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI
KM	53.5±0.3	36.6±0.1	49.8±0.5	67.3±0.1	53.5±0.1	61.4±0.1	60.4	45.0	56.9
AE-KM	80.8±1.8	69.4±1.8	75.2±1.1	72.9±0.8	63.2±1.5	71.7±1.2	76.8	66.3	73.4
DCN^{np} (EE)	16.8±0.5	1.4±0.1	2.0±0.2	16.3±0.9	1.0±0.2	1.7±0.3	16.5	1.2	1.8
IDEC^{np} (EE)	61.8±3.0	49.1±3.0	62.4±1.6	53.9±5.1	40.2±5.1	50.0±3.8	57.8	44.6	56.2
DKM^a (EE)	82.3±3.2	73.6±3.1	78.0±1.9	75.5±6.8	66.3±4.9	73.0±2.3	78.9	69.9	75.5
DCN^p (EE)	81.1±1.9	70.2±1.8	75.7±1.1	73.0±0.8	63.4±1.5	71.9±1.2	77.0	66.8	73.8
IDEC^p (EE)	**85.7±2.4**	**81.5±2.4**	**86.4±1.0**	75.2±0.5	68.1±0.5	74.9±0.6	80.4	74.8	80.6
DKM^p (EE)	84.0±2.2	75.0±1.8	79.6±0.9	**75.7±1.3**	**68.5±1.8**	**77.6±1.1**	79.8	70.4	78.6

Table 4.7: Clustering results of the compared methods on text data (TF-IDF). Performance is measured in terms of Accuracy, NMI and ARI (%); higher is better. Each cell contains the average and standard deviation computed over 10 runs. The best result for each dataset/metric is bolded. Underlined values correspond to results with no significant difference ($p > 0.05$) to the best.

Model	20NEWS			RCV1			Macro-Average		
	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI
KM	23.2±1.5	7.6±0.9	21.6±1.8	50.8±2.9	20.6±2.8	31.3±5.4	37.0	17.1	26.4
AE-KM	49.0±2.9	31.0±1.6	44.5±1.5	56.7±3.6	23.9±4.3	31.5±4.3	53.3	27.4	38.0
DCN^{np}(EE)	7.9±0.2	0.3±0	1.1±0.1	29.2±1.6	0.5±0.2	0.6±0.2	18.5	0.4	0.8
IDEC^{np}(EE)	22.3±1.5	9.8±1.5	22.3±1.5	56.7±5.3	28.5±5.3	31.4±2.8	39.5	19.1	26.8
DKM^a(EE)	44.8±2.4	26.7±1.5	42.8±1.1	53.8±5.5	20.7±4.4	28.0±5.8	49.3	23.7	35.4
DCN^p(EE)	49.2±2.9	31.3±1.6	44.7±1.5	56.7±3.6	24.0±4.3	31.6±4.3	52.9	27.6	38.1
IDEC^p(EE)	40.5±1.3	26.0±1.3	38.2±1.0	**59.5±5.7**	**32.9±5.7**	**34.7±5.0**	50.0	29.4	36.4
DKM^p(EE)	**51.2±2.8**	**33.9±1.5**	**46.7±1.2**	58.3±3.8	26.5±4.9	33.1±4.9	**54.7**	**30.2**	**39.9**

Table 4.8: The results of multiple clustering approaches obtained by general hyperparameters. Bold values represent the best results and underlined values have no statistically significant difference from the best results at $\alpha = 0.05$.

Model		KM	AE	DKM^p (EC)	DKM ^{p} (CC)	DCN ^{p} (EC)	DCN^p (CC)	IDEC ^{p}
20NEWS	ACC	68.3±1.8	75.2±1.4	**77.4±1.4**	77.0±1.7	76.0±1.7	67.8±2.4	73.3±3.0
	ARI	51.9±1.3	60.6±1.4	**64.6±1.1**	61.8±1.8	61.4±1.4	52.0±2.1	56.9±3.9
	NMI	68.8±0.5	0.8±0.7	1.9±0.7	**72.0±0.5**	70.9±0.7	66.2±1.3	69.7±1.6
REUTERS	ACC	39.8±1.0	39.7±1.2	42.2±1.1	41.9±4.5	40.5±0.9	**42.6±5.3**	42.4 ±1.1
	ARI	30.9±1.1	32.3±1.3	32.0±1.2	32.3±4.2	**32.4±1.0**	33.1±4.3	32.3±0.8
	NMI	54.3±1.4	**56.1±1.3**	54.7±1.1	54.3±1.8	55.8±1.3	55.2±1.8	5.4 ±0.7
YAHOO	ACC	52.5±2.1	53.8±2.6	56.3±2.7	54.4±3.1	54.3±2.7	**58.7±2.7**	53.2± 2.0
	ARI	27.7±0.8	29.9±1.4	**32.8±1.2**	31.4±1.6	30.6±1.6	32.7±1.4	27.3±3.4
	NMI	37.4±0.5	37.5±0.7	38.2±0.8	37.5±1.0	37.6±0.7	**39.9±0.9**	36.3±0.6
DBPEDIA	ACC	61.1±1.8	7.2±3.0	67.0±2.6	68.5±3.2	**69.0±2.2**	68.0±3.4	68.0±2.2
	ARI	51.9±2.1	57.6±4.0	58.0±4.0	57.8±3.4	**58.9±3.6**	54.5±3.4	56.6±3.2
	NMI	70.6±1.0	72.5±2.0	71.1±1.9	71.8±1.6	73.3±1.4	69.6±1.5	72.1±1.3
AGNEWS	ACC	84.1±0.0	84.4±0.7	84.9±0.3	83.1±1.7	84.2±0.5	**85.4±0.5**	84.0±1.0
	ARI	64.7±0.0	63.8±1.5	64.7±0.6	61.2±3.7	63.3±1.0	**65.7±1.1**	62.3±2.3
	NMI	64.0±0.0	59.6±1.1	60.3±0.5	57.7±3.4	59.1±0.8	**61.3±0.9**	58.7±1.6
Macro-average	ACC	61.1	64.0	**65.5**	65.0	64.8	64.5	64.1
	ARI	45.4	48.8	**50.4**	48.9	49.3	47.6	47.0
	NMI	59.0	**59.3**	59.2	58.6	**59.3**	58.4	58.4

Table 4.9: The results of multiple clustering approaches obtained by best hyperparameters. Bold values represent the best results and underlined values have no statistically significant difference from the best results at $\alpha = 0.05$.

Model		KM	AE	DKM ^{p} (EC)	DKM ^{p}(CC)	DCN^p (EC)	DCN^p (CC)	IDEC ^{p}
20NEWS	ACC	68.3±1.8	75.2±1.4	77.4±1.4	**79.5±0.6**	76.0±1.7	67.8±2.4	73.3±3.0
	ARI	51.9±1.3	60.6±1.4	64.6±1.1	**64.8±0.7**	61.4±1.4	52.0±2.1	56.9±3.9
	NMI	68.8±0.5	70.8±0.7	**71.9±0.7**	**71.9±0.5**	70.9±0.7	66.2±1.3	69.7±1.6
REUTERS	ACC	39.8±1.0	39.7±1.2	46.9±3.6	**64.3±6.0**	40.5±0.9	43.4±6.3	42.4 ±1.1
	ARI	30.9±1.1	32.3±1.3	37.0±3.4	**50.3±10.4**	32.4±1.0	33.5±5.2	32.3±0.8
	NMI	54.3±1.4	**56.1±1.3**	55.1±1.9	54.4±6.8	55.8±1.3	55.2±1.8	55.4 ±0.7
YAHOO	ACC	52.5±2.1	53.8±2.6	56.3±2.7	55.1±2.8	54.7±3.0	**58.7±2.7**	53.4± 2.4
	ARI	27.7±0.8	29.9±1.4	**32.8±1.2**	31.9±1.6	30.6±1.6	32.7±1.4	27.4±1.3
	NMI	37.4±0.5	37.5±0.7	38.2±0.8	37.7±1.0	37.8±0.7	**39.9±0.9**	36.4±0.6
DBPEDIA	ACC	61.1±1.8	67.2±3.0	67.0±2.6	68.5±3.2	**69.0±2.2**	68.0±3.4	68.0±2.2
	ARI	51.9±2.1	57.6±4.0	58.0±4.0	57.8±3.4	**58.9±3.6**	54.5±3.4	56.6±3.2
	NMI	70.6±1.0	72.5±2.0	71.1±1.9	71.8±1.6	**73.3±1.4**	69.6±1.5	72.1±1.3
AGNEWS	ACC	84.1±0.0	84.4±0.7	84.9±0.3	83.1±1.7	84.6±0.4	**85.4±0.5**	84.0±1.0
	ARI	64.7±0.0	63.8±1.5	64.7±0.6	61.2±3.7	64.0±0.8	**65.7±1.1**	62.3±2.3
	NMI	64.0±0.0	59.6±1.1	60.3±0.5	57.7±3.4	59.1±0.6	**61.3±0.9**	58.7±1.6
Macro-average	ACC	61.1	64.0	66.5	**70.1**	64.9	64.6	64.1
	ARI	45.4	48.8	51.4	**53.2**	49.4	47.6	47.0
	NMI	59.0	59.3	59.3	58.7	**59.3**	58.4	58.4

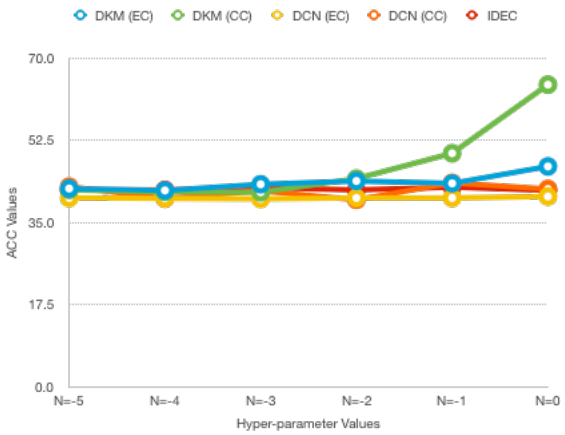


Figure 4.5: Accuracy evolution of different deep clustering methods with respect to hyper-parameter values (10^{-N}) on Reuters.

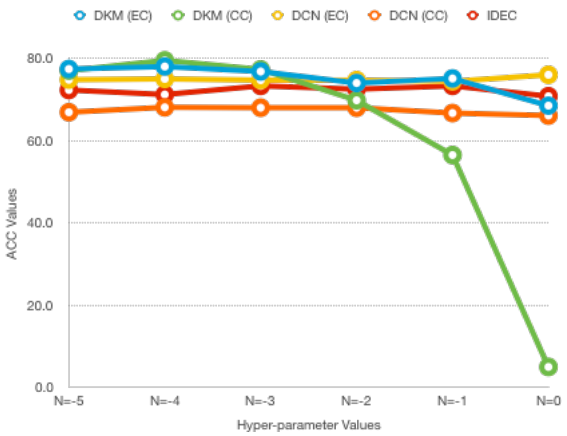


Figure 4.6: Accuracy evolution of different deep clustering methods with respect to hyper-parameter values (10^{-N}) on 20news.

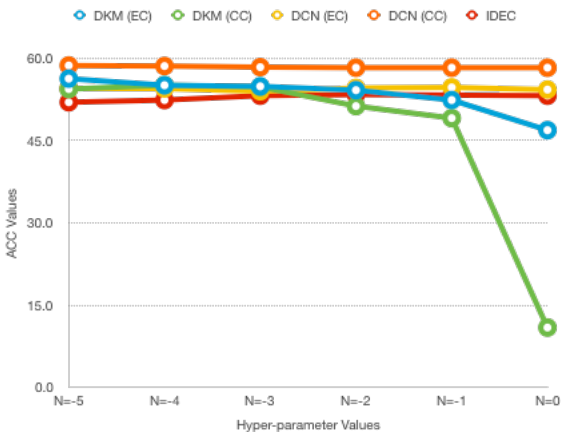


Figure 4.7: Accuracy evolution of different deep clustering methods with respect to hyper-parameter values (10^{-N}) on Yahoo.

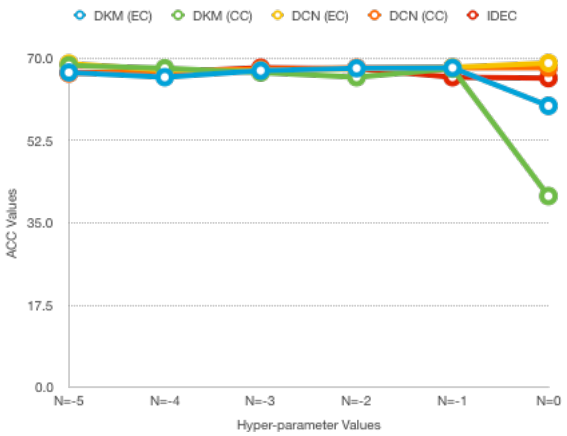


Figure 4.8: Accuracy evolution of different deep clustering methods with respect to hyper-parameter values (10^{-N}) on DBPedia.

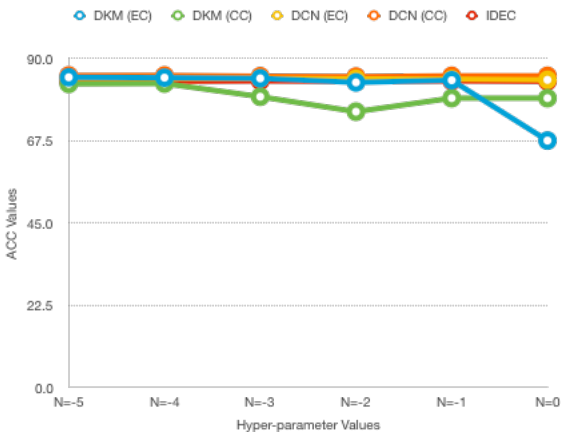
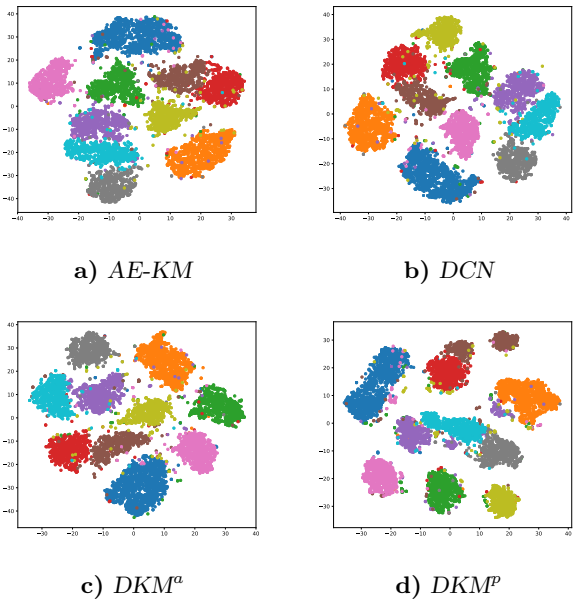


Figure 4.9: Accuracy evolution of different deep clustering methods with respect to hyper-parameter values (10^{-N}) on AGnews.

Table 4.10: Clustering results for k-Means applied to different learned embedding spaces to measure the k-Means-friendliness of each method. Performance is measured in terms of NMI and clustering accuracy (%); higher is better. Each cell contains the average and standard deviation computed over 10 runs. The best result for each dataset/metric is bolded. Underlined values correspond to results with no significant difference ($p > 0.05$) to the best.

Model	MNIST		USPS		20NEWS		RCV1	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
AE-KM	80.8±1.8	75.2±1.1	72.9±0.8	71.7±1.2	49.0±2.9	44.5±1.5	56.7±3.6	31.5±4.3
DCN^{p} + KM	84.9±3.1	79.4±1.5	73.9±0.7	74.1±1.1	50.5±3.1	46.5±1.6	57.3±3.6	32.3±4.4
DKM^{a} + KM	84.8±1.3	78.7±0.8	**76.9±4.9**	74.3±1.5	49.0±2.5	44.0±1.0	**53.4±5.9**	27.4±5.3
DKM^{p} + KM	**85.1±3.0**	**79.9±1.5**	75.7±1.3	**77.6±1.1**	**52.1±2.7**	**47.1±1.3**	**58.3±3.8**	**33.0±4.9**

4.3.5 k-Means-friendliness of the learned representations



a) *AE-KM* **b)** *DCN*

c) DKM^{a} **d)** DKM^{p}

Figure 4.10: t-SNE visualization of the embedding spaces learned on USPS.

In the previous experiment, we investigated the clustering ability of the different models. While the quality of the clustering results and that of the representations learned by the models are likely to be correlated, it is relevant to study to what extent learned representations are distorted to facilitate clustering – in other words, how they are biased towards "clustering-friendliness". More specifically, we focus on the representations learned by the deep clustering approaches related to k-Means: DCN^{p}, DKM^{a}, and DKM^{p} (DCN^{np} was discarded due to its poor clustering performance). We analyze how effective applying k-Means to these representations is in comparison to

applying k-Means to the AE-learned representations (*i.e.*, AE-KM).

We can observe that on most datasets the representations learned by k-Means-related deep clustering approaches lead to significant improvement with respect to AE-learned representations. This confirms that all these deep clustering methods truly bias their representations. Interestingly, we note that applying k-Means to the representations learned by DCN$^{\text{p}}$ yields substantial improvements in comparison to the results. Overall, although the difference is not statistically significant on all datasets/metrics, the representations learned by DKM$^{\text{p}}$ are shown to be the most appropriate to k-Means. This goes in line with the insight gathered from Section 4.3.4.

To further support this latter finding and to bring a more interpretable view of the learned representations, we illustrate the embedded samples provided by AE, DCN$^{\text{p}}$, DKM$^{\text{a}}$, DKM$^{\text{p}}$ on USPS in Figure 4.10 (best viewed in color). We used for that matter the t-SNE visualization method [128] which projected embeddings into a 2D space. We observe that the representations for points from different clusters are clearly better separated and disentangled in DKM$^{\text{p}}$ than in other models. This once again supports the superior ability of DKM$^{\text{p}}$ to learn k-Means-friendly representations.

4.4 Conclusion

We have presented in this chapter a new approach for jointly clustering with k-Means and learning representations by considering the k-Means clustering loss as the limit of a differentiable function. To the best of our knowledge, this is the first approach that truly jointly optimizes, through simple stochastic gradient descent updates, representation and k-Means clustering losses. In addition to pretraining, that can be used in all methods, this approach can also rely on a deterministic annealing scheme for parameter initialization. Based on the results, deterministic annealing scheme can be useful in some cases and even working better than DCN^{p} in some cases. We further conducted careful comparisons with previous approaches by ensuring that the same architecture, initialization and sequence minibatches are used. The experiments conducted on several datasets (image and text datasets) confirm the good behavior of the proposed approach. At the first glance at the results presented earlier, we can conclude pretraining is an essential step in DC which shall not be neglected. On the other hand, the proposed deterministic annealing scheme can be improved or one can introduce a new scheme to outperform current results. On image datasets, both DKM and IDEC can be chosen as the best approaches. On text datasets using TF-IDF, except for 20NEWS where all the DC methods perform equally (including

AE-KM), on RCV1 our proposed DKM method can outperform IDEC and DCN.

The results obtained by applying general hyper-parameters indicates that DKM variations obtain the best Macro-average results of ACC and ARI results while other approaches can achieve better NMI results. In the detailed results we can notice that on each collection different methods can achieve the best results but as mentioned earlier on average DKM variants perform the best on ACC and ARI.

The results obtained by the tuned hyper-parameters indicate that DKM^p (CC) outperforms other approaches on Macro-average results of ACC and ARI. DKM^p (EC) and DCN^p (EC) share the same best Macro-average result for NMI.

In the next chapter we will investigate different deep clustering approaches while enforcing constraints.

CHAPTER 5

Constrained Deep Document Clustering

Different users may be interested in different clustering views underlying a given collection (e.g., topic and writing style in documents). Enabling them to provide constraints reflecting their needs can then help obtain tailored clustering results. For document clustering, constraints can be provided in the form of seed words, each cluster being characterized by a small set of words. This seed-guided constrained document clustering problem was recently addressed through topic modeling approaches. Another form of constraints is the Must-links (MLs) and Cannot-links (CL) where given a pair of document, one can annotate the documents based on their relatedness. Meaning that if a pair of documents are related then they will be considered as ML otherwise CL. In this paper, we jointly learn deep representations and bias the clustering results through the seed words or MLs and CLs, respectively leading to Seed-guided Deep Document Clustering [129] and Pairwise-Constrained Deep Document Clustering [130] approaches.

5.1 Introduction

Clustering traditionally consists in partitioning data into subsets of similar instances with no prior knowledge on the clusters to be obtained. However, clustering is an ill-defined problem in the sense that the data partitions output by clustering algorithms have no guarantee to satisfy end users' needs. Indeed, different users may be interested in different views underlying

the data [131]. Also due to the subjectivity aspect of clustering, including constraints in the clustering seems essential. Indeed, by including constraints in a deep clustering framework, more suitable and tailored partitions can be obtained. In deep clustering, including constraints can be done through biasing the representations.

Figure 5.1: Different vehicle types and colors.

As can be seen in Fig 5.1, one can separate the vehicles based on their colors (blue or red) or based on vehicle types (car or bicycle). That is why expert external knowledge can be important to be integrated in the clustering. As another example, considering either the topics or the writing style in a collection of documents leads to different clustering results. In this study, we consider a setting where clustering is guided through user-defined constraints, which is known as *constrained clustering* [102]. Enabling users to provide clustering constraints in the context of an exploratory task can help obtain results better tailored to their needs. Typically, ML and CL constraints are considered (e.g., see [104, 132]), which state whether two data instances should be (respectively, should not be) in the same cluster. However, important manual annotation efforts may still be required to provide such constraints in sufficient number. In the specific case of document clustering, constraints can otherwise be provided in the form of *seed words*: each cluster that the user wishes to obtain is described by a small set of words (e.g., 3 words) which characterize the cluster. For example, a user who wants to explore a collection of news articles might provide the set of seed words {'sport', 'competition', 'champion'}, {'finance', 'market', 'stock'}, {'technology', 'innovation', 'science'} to guide the discovery of three clusters on sport, finance, and technology, respectively. Recent studies which include seed word constraints for document clustering are mostly focused on topic modeling approaches [95–98], inspired by the Latent Dirichlet Allocation

model [99].

In the previous chapter we introduced our DKM framework and thoroughly analyzed its results with respect to IDEC and DCN. One advantage of deep clustering approaches lies in their ability to leverage semantic representations based on word embeddings, enabling related documents to be close in the embedding space even when they use different (but related) words.

The main contributions of this chapter can be summarized as follows: (a) We introduce the **S**eed-guided **D**eep **D**ocument **C**lustering (SD2C) framework, the first attempt, to the best of our knowledge, to constrain clustering with seed words based on a deep clustering approach. (b) We introduce the **P**airwise-Constrained **D**eep **D**ocument **C**lustering (PCD2C) framework. (c) We validate these frameworks through experiments based on automatically selected seed words on five publicly available text datasets with various sizes and characteristics; and (d) We conducted human experiments that involved users to participate in the selection of seed words and ML/CL documents leading to a close to realistic experiments where we can analyze SD2C and PCD2C approaches in details and measure their performance, time consumption, etc. in a real case scenario.

5.2 Seed-guided Deep Document Clustering

Deep clustering consists in jointly performing clustering and deep representation learning in an unsupervised fashion (e.g., with an Autoencoder). All deep clustering approaches aim at obtaining representations that are both faithful to the original data and are more suited to clustering purposes than the original representation. To do so, Autoencoder-based deep clustering approaches trade off between a reconstruction loss, denoted $\mathcal{L}_{\text{rec}}$, and a clustering loss, denoted $\mathcal{L}_{\text{clust}}$, through a joint optimization problem of the form: $\mathcal{L}_{\text{rec}} + \lambda_0 \mathcal{L}_{\text{clust}}$, where λ_0 is an hyperparameter balancing the contribution of the reconstruction and clustering losses.

In the remainder, $\mathcal{X}$ will denote the set of documents to cluster. Each document $x \in \mathcal{X}$ is associated with a representation $\mathbf{x}$ in $\mathbb{R}^d$ – thereafter, the *input space* – defined as the average of the (precomputed) embeddings of the words in x, where d is the dimension of the word embedding space. Each word w is thus represented as a d-dimensional vector $\mathbf{w}$ corresponding to its embedding. Let $f_\theta : \mathbb{R}^d \to \mathbb{R}^p$ and $g_\eta : \mathbb{R}^p \to \mathbb{R}^d$ be an encoder and a decoder with parameters θ and η, respectively; $g_\eta \circ f_\theta$ then defines an Autoencoder. $\mathbb{R}^p$ denotes the space in which we wish to embed the learned document representations – thereafter, the *embedding space*. Lastly, we denote by $\mathcal{R}$ the parameters of the clustering algorithm. With a slight abuse of notations

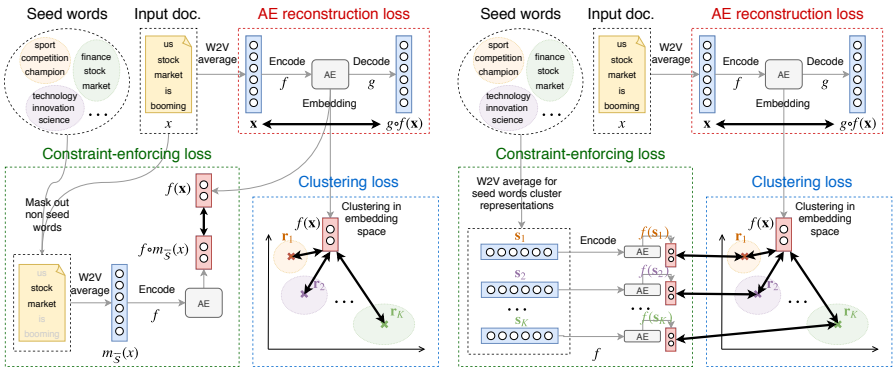


Figure 5.2: Illustration of SD2C-Doc (left) and SD2C-Rep (right). Thick double arrows indicate the computation of a distance between two vectors.

in which $f_\theta(\mathcal{X})$ corresponds to the application of the function f_θ to each element of the set $\mathcal{X}$, the overall deep clustering (DC) optimization problem takes the form:

$$\underset{\theta,\eta,\mathcal{R}}{\arg\min} \underbrace{\mathcal{L}_{\text{rec}}(\mathcal{X}, g_\eta \circ f_\theta(\mathcal{X})) + \lambda_0 \mathcal{L}_{\text{clust}}(f_\theta(\mathcal{X}), \mathcal{R})}_{\mathcal{L}_{\text{dc}}(\mathcal{X},\theta,\eta,\mathcal{R})}. \tag{5.1}$$

We propose to integrate constraints on seed words in this framework by biasing the embedding representations, which guarantees that the information pertaining to seed words will be used in the clustering process. This can be done by enforcing that seed words have more influence either on the learned document embeddings, a solution we refer to as **SD2C-Doc**, or on the cluster representatives, a solution we refer to as **SD2C-Rep**. Note that the second solution can only be used when the clustering process is based on cluster representatives (i.e., $\mathcal{R} = \{r_k\}_{k=1}^K$ with K the number of clusters), which is indeed the case for most current deep clustering methods [133].

In addition to the notations introduced previously, we will denote by s_k the subset of seed words corresponding to cluster k, and by $\mathcal{S} = \{s_k\}_{k=1}^K$ the complete set of seed words defining the prior knowledge on the K clusters to recover. We further define $\overline{\mathcal{S}} = \bigcup_{k=1}^K s_k$, the set of seed words from all clusters.

5.2.1 SD2C-Doc

To bias the document representations according to the seed words, we first define, for each document, a masked version of it that is based on seed words. This can be done *aggressively*, by retaining, in the masked version, only the words that correspond to seed words and by computing an average of

their word embeddings, or *smoothly* by reweighing all words in the original document according to their proximity with seed words. A weighted average of their embeddings then defines the smooth, masked version of the documents. The equation below formalizes these two approaches:

$$m_{\overline{\mathcal{S}}}(x) = \begin{cases} \frac{1}{\sum_{w \in \overline{\mathcal{S}}} \mathrm{tf}_x(w)} \sum_{w \in \overline{\mathcal{S}}} \mathrm{tf}_x(w) \cdot \mathbf{w} \\ \frac{1}{|\overline{\mathcal{S}}| \cdot |x|} \sum_{w' \in x} \sum_{w \in \overline{\mathcal{S}}} \frac{1 + \cos(\mathbf{w}, \mathbf{w}')}{2} \cdot \mathbf{w}' \end{cases} \tag{5.2}$$

where cos denotes the cosine similarity. If a document x does not contain any seed word, $m_{\overline{\mathcal{S}}}(x)$ is ill-defined when using the first version of Eq. 5.2 as $\sum_{w \in \overline{\mathcal{S}}}$ is null in that case. To address this issue, one can simply discard the documents without seed words. In practice, the two masked versions of Eq. 5.2 yielded the same results in our experiments. Because of its simplicity, we rely on the first one in the remainder of the paper. One can then force the embedding representation of documents to be close to the embedding of their masked version by minimizing the dissimilarity in the embedding space, denoted by δ^E, between $f_\theta(\mathbf{x})$ and $f_\theta \circ m_{\overline{\mathcal{S}}}(x)$, leading to:

$$\arg\min_{\theta, \eta, \mathcal{R}} \mathcal{L}_{\mathrm{dc}}(\mathcal{X}, \theta, \eta, \mathcal{R}) + \lambda_1 \sum_{x \in \mathcal{X}} \delta^E\left(f_\theta(\mathbf{x}), f_\theta \circ m_{\overline{\mathcal{S}}}(x)\right), \tag{5.3}$$

where λ_1 is an hyperparameter controlling the importance of the deep clustering loss $\mathcal{L}_{dc}$ and the loss associated to seed words. Fig. 5.2 (left) illustrates the global architecture corresponding to this problem.

5.2.2 SD2C-Rep

The other bias one can consider in the embedding space is the one related to cluster representatives. Here, one can naturally push cluster representatives towards the representation of seed words, in order to ensure that the discovered clusters will account for the prior knowledge provided by them. For that purpose, we first build a representation for each subset of seed words by averaging the word embeddings of the seed words it contains:

$$\mathbf{s}_k = \frac{1}{|s_k|} \sum_{w \in s_k} \mathbf{w}.$$

$\mathbf{s}_k$ thus corresponds to the seed word-based representation of cluster k in $\mathbb{R}^d$. The optimization problem solved by SD2C-Rep, depicted in Fig. 5.2 (right),

then takes the form:

$$\arg\min_{\theta,\eta,\mathcal{R}} \mathcal{L}_{\text{dc}}(\mathcal{X},\theta,\eta,\mathcal{R}) + \lambda_1 \sum_{k=1}^{K} \delta^E(\mathbf{r}_k, f_\theta(\mathbf{s}_k)) \tag{5.4}$$

As before, δ^E denotes a dissimilarity in the embedding space. The last term in Eq. 5.4 forces cluster representatives to be close to subsets of seed words, the alignment between the two being defined by the initialization of the cluster representatives performed after pretraining.

5.2.3 SD2C-Att

The last approach directly operates on the input representations of the documents that are fed to the model to account for the seed words. Indeed, allocating different weights to the words in a document according to their proximity to seed words might help provide input representations which are better tailored to the constrained clustering problem we wish to solve. We here propose to learn those weights based on an attention mechanism [134].[1]

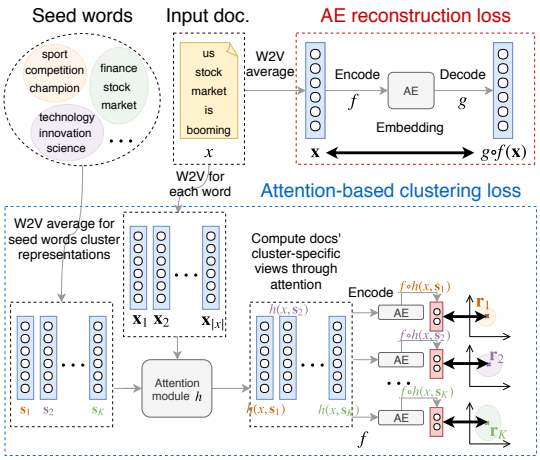


Figure 5.3: Illustration of SDC2-Att. The function h denotes the seed words-based attention module applied to the documents' input representations. Thick double arrows indicate the computation of a distance between two vectors.

In addition to the biasing of document representatives (as in SD2C-Doc) or the biasing of cluster representatives (as in SD2C-Rep), this approach would consist in operating on the input representations directly.

[1] An alternative parameter-free attention mechanism simply consists in fixing those weights based on the similarity between words in a document and the seed words. This is however only a special case of the learned attention mechanism we study here.

The seed words from different clusters are used to compute different 'views' of each document (one view per cluster). These different views are then integrated in the k-Means clustering loss. Note that, in this version, no term is added to the formulation in Problem 5.1 – only the input representation fed to the clustering module is changed. The resulting model, which we denote as SDC2-I, is illustrated in Fig. 5.3. Its optimization problem corresponds to:

$$\underset{\theta,\eta,\zeta,\mathcal{R}}{\arg\min} \sum_{x\in\mathcal{X}} \delta^I(\mathbf{x}, g_\eta \circ f_\theta(\mathbf{x})) + \lambda \sum_{x\in\mathcal{X}} \min_{1\leq k\leq K} \delta^E(f_\theta \circ h_\zeta(x, \mathbf{s}_k), \mathbf{r}_k) \tag{5.5}$$

where $h_\zeta(x, \mathbf{s}_k)$ learns a new input representation for document x biased by the k-th seed word cluster representation $\mathbf{s}_k$ through an attention mechanism. We define $h_\zeta(x, \mathbf{s}_k)$ as follows, introducing the attention score e^x_{jk} and attention weight α^x_{jk} for $j = 1, \ldots, |x|$, $k = 1, \ldots, K$:

$$h_\zeta(x, \mathbf{s}_k) = \sum_{j=1}^{|x|} \alpha^x_{jk} \mathbf{w}^x_j, \quad \alpha^x_{jk} = \frac{\exp\left(e^x_{jk}\right)}{\sum_{j'=1}^{|x|} \exp\left(e^x_{j'k}\right)}, \quad e^x_{jk} = a_\zeta(\mathbf{w}^x_j, \mathbf{s}_k) \tag{5.6}$$

where $\mathbf{w}^x_j$ is the word embedding for the j-th word of document x, $|x|$ is the length of document x, and a_ζ is a map parameterized by ζ. Here, we use an additive attention mechanism and define a_ζ as a neural network with one hidden layer as follows:

$$a_\zeta(\mathbf{w}^x_j, \mathbf{s}_k) = \mathbf{u}^\top \tanh\left(W[\mathbf{w}^x_j; \mathbf{s}_k]\right) \tag{5.7}$$

where $\zeta = \{\mathbf{u}, W\}$ are the parameters to learn ($\mathbf{u} \in \mathbb{R}^h$, $W \in \mathbb{R}^{h\times 2d}$ with h the dimension of the hidden space) and $[\mathbf{w}^x_j; \mathbf{s}_k] \in \mathbb{R}^{2d}$ denotes the vertical concatenation of $\mathbf{w}^x_j$ and $\mathbf{s}_k$. Note that because of the attention mechanism, which has to compute weights α^x_{jk} for every word j in x, every cluster k, and every document x, SDC2-Att is computationally more expensive than the two previous variants. Further, as the results for SDC2-Att were less promising than those of SDC2-Doc and SDC2-Rep in our preliminary study, we essentially focus on the latter approaches in the experiments.

5.3 Pairwise-Constrained Deep Document Clustering

In case of document clustering, the relevance in topic can be used for measuring the similarity between documents while it is expected that similar documents fall into the same category. Due to the subjectivity aspect of clustering, different users might impose different constraints on clustering. For instance, to cluster a set of news articles, one might be interested in having two clusters where one indicates liberal views and the other one democratic views, while another user might be interested in dividing documents into economy and environment topics. Constraints can take different forms such as *seed words* (discussed earlier) and *pairwise constraints*. In this section, we will study pairwise constraints where a set of must-link and cannot-link documents are used as side information which can lead towards better clustering results. As the names imply, must-link pairs contain documents coming from the same category while cannot-link documents contain documents coming from different categories. This additional information is provided by the user before applying clustering and will be taken into account while clustering the underlying data.

So far, there have been several studies about using pairwise constraints in clustering [105, 106, 135] which investigated the efficiency of side information (pairwise constraints) but none of them have explored the impact of pairwise constraints in an end-to-end Autoencoder-based deep clustering framework. To answer this problem in an end-to-end deep clustering framework, we propose the **P**airwise-**C**onstrained **D**eep **D**ocument **C**lustering (PCD2C) model [130] which benefits from a fully differentiable objective function.

PCD2C Formulation

Based on prior deep clustering and SD2C definitions, we can consider pairwise constraints as additional information for deep clustering framework as well which yield in proposing a new objective function. In order to integrate the pairwise constraints in a deep clustering framework, we can simply integrate them in 5.1. This can be done by minimizing the distance between must-link documents and maximizing the distance between cannot-link documents (both in the learned document embedding space). In this case, pairwise information can be used to bias data representations which yields improved clustering results. CL can be considered as the list containing all cannot-link document pairs and ML can be considered as the list containing all must-link document pairs. To involve pairwise constraints, the following formulation is

proposed:

$$\arg\min_{\theta,\eta,\mathcal{R}} \mathcal{L}_{\text{dc}}(\mathcal{X},\theta,\eta,\mathcal{R}) + \lambda_1 \Bigg(\frac{1}{|ML(minibatch)|} \sum_{(i,j)\in ML(minibatch)} \delta^E(f_\theta(\mathbf{x}_i), f_\theta(\mathbf{x}_j)) + \frac{1}{|CL(minibatch)|} \sum_{(i,j)\in CL(minibatch)} max(0, \mu - \delta^E(f_\theta(\mathbf{x}_i), f_\theta(\mathbf{x}_j)) \Bigg) \tag{5.8}$$

Thus, one can minimize the dissimilarity – denoted by δ^E– between must-link documents and maximize the dissimilarity between cannot-link documents to utilize pairwise constraints information. λ_1 is an hyperparameter which controls the importance of clustering loss and pairwise constraint loss. In this formulation, if a pair of documents linked by an ML or CL constraint occurs in the current minibatch during training, then the constraint loss will be activated otherwise it will not affect the total loss (set to zero).

5.3.1 Choice of Deep Clustering Framework

In practice, we use fully differentiable formulations of Problems 5.3, 5.4 and 5.8. In the context of the k-Means algorithm, a popular clustering method, such differentiable formulations can be directly developed on top of the algorithms provided in [29] (called DCN) and [136] (called DKM), the latter proposing a truly joint formulation of the deep clustering problem. Other state-of-the-art deep clustering approaches, as IDEC [31], also based on cluster representatives, could naturally be adopted as well. The comparison between these approaches performed in [136] (shown in the previous chapter as well) nevertheless suggests that DKM outperforms the other approaches. This difference was confirmed on the text collections retained in this study. We thus focus here on the DKM algorithm introduced in [136] with:

$$\mathcal{L}_{\text{rec}}(\mathcal{X}, g_\eta \circ f_\theta(\mathcal{X})) = \sum_{x\in\mathcal{X}} \delta^I(\mathbf{x}, g_\eta \circ f_\theta(\mathbf{x}))\,, \tag{5.9}$$

where δ^I denotes a dissimilarity in the input space, and:

$$\mathcal{L}_{\text{clust}}(f_\theta(\mathcal{X}), \mathcal{R}) = \sum_{x\in\mathcal{X}} \sum_{k=1}^{K} \delta^E(f_\theta(\mathbf{x}), \mathbf{r}_k)\, G_k(f_\theta(\mathbf{x}), \alpha; \mathcal{R}) \tag{5.10}$$

where α is an inverse temperature parameter and $G_k(f_\theta(\mathbf{x}), \alpha; \mathcal{R})$ is a softmax function parameterized by α defined as follows:

$$G_k(f_\theta(\mathbf{x}), \alpha; \mathcal{R}) = \frac{\exp\Big(-\alpha \cdot \delta^E(f_\theta(\mathbf{x}), \mathbf{r}_k)\Big)}{\sum_{k'=1}^{K} \exp\Big(-\alpha \cdot \delta^E(f_\theta(\mathbf{x}), \mathbf{r}_{k'})\Big)}. \tag{5.11}$$

The k-means solution is recovered when α tends to $+\infty$.

5.4 Experiments

In this section, we discuss thoroughly the way we conducted the experiments and provide all of the required information.

5.4.1 Datasets

The experiments we performed to evaluate the proposed SD2C and PCD2C frameworks are based on five publicly available datasets with various sizes and characteristics that have been extensively used in the context of text classification and clustering: The 20 Newsgroups[2] dataset, referred to as *20NEWS*; the Reuters-21578[3] dataset, referred to as *REUTERS*, from which, similarly to [95–98], we use only the 10 largest (and highly imbalanced) categories; the Yahoo! Answers dataset introduced in [122], referred to as *YAHOO*, from which we use only the test set comprising 60,000 documents evenly split into 10 classes; the DBPedia dataset curated in [122], referred to as *DBPEDIA*, from which we also only use the test set made of 70,000 documents uniformly distributed in 14 classes; and the AG News dataset, introduced as well in [122] and referred to as *AGNEWS*, from which we use the training set, composed of 120,000 documents evenly split into 4 classes.

5.4.2 SD2C Baselines and Variants

For both SD2C-Doc and SD2C-Rep, different dissimilarities can be adopted for δ^I and δ^E. As the cosine distance performed consistently better for δ^E than the Euclidean distance in our preliminary experiments, it is adopted here. We nevertheless did not observe such a clear trend for δ^I, and we indicate here the results obtained both for the cosine distance and Euclidean

[2] http://qwone.com/~jason/20Newsgroups/
[3] http://www.daviddlewis.com/resources/testcollections/reuters21578/

distance. This yields two versions for each method, which we denote as SD2C-Doc-e/SD2C-Rep-e and SD2C-Doc-c/SD2C-Rep-c, depending on whether the Euclidean or the cosine distance is used for δ^I, respectively.

To compare against SD2C, we considered the following baseline methods:

- *KM*, *AE-KM* and different deep clustering frameworks: KM corresponds to k-Means [1] applied on the same input for documents as the one used for SD2C (average of documents' word embeddings); AE-KM first trains an Autoencoder on the collection and then applies k-Means to the document embeddings learned by the Autoencoder; DKM is the deep k-Means algorithm[4] presented in [136] which we also study under the two variants DKM-e and DKM-c. Also, other deep clustering frameworks such as DCN [29] and IDEC [31] can be used in this framework and their results are compared versus DKM algorithm. [5]
- *NN*: This method is similar to the 'on-the-fly' nearest neighbor-like classification described in [92]. Each document, represented by its word embeddings average, is assigned to the nearest class, in terms of the cosine distance, which outperformed the Euclidean distance, represented by the class' average seed word embeddings (denoted as $\{\mathbf{s}_k\}_{k=1}^K$ in Sec 5.2).
- *STM*: In our experiments, we ran the Java implementation of the Seed-guided Topic Model [96] provided by the authors[6] and used the standard hyperparameters indicated in the paper. Given that this approach was not scalable when the whole vocabulary is used, we only kept the 2000 most frequent words (after preprocessing) for each dataset[7].

5.4.3 PCD2C Baselines and Variants

Similar to SD2C, in this case, *KM*, *AE-KM* and different Deep Clustering frameworks such as DKM, DCN and IDEC are being used for evaluating the

[4] https://github.com/MaziarMF/deep-k-means

[5] seed words are not utilized in these approaches.

[6] https://github.com/ly233/Seed-Guided-Topic-Model

[7] Very recently, another topic modeling approach, the Laplacian seed word Topic Model (LapSWTM), was proposed in [97]. However, firstly, LapSWTM counts 8 hyperparameters that were empirically optimized in the original paper, and it is not straightforward how these hyperparameters should be tuned on the additional datasets used here. Secondly, LapSWTM shares a lot with the STM model in its construction and performance. Thirdly, the code for LapSWTM is, as far as we are aware, not publicly available. For these different reasons, we simply chose STM to represent the state of the art in topic modeling-based dataless text classification.

effectiveness of the pairwise constraints.[8]

5.4.4 Constraints Selection

To evaluate the proposed SD2C and PCD2C algorithms, we followed two approaches to select constraints: 1) Automatic Approach and 2) Manual Approach which are detailed below.

5.4.4.1 SD2C: automatic seed words selection

Recent works on dataless text classification [95–98] only considered the 20NEWS and REUTERS datasets in their experiments, relying respectively on the seed words induced by the class labels and on the manually curated seed words from [95]. To perform an evaluation on all the collections retained here, we devised a simple heuristics based on tf-idf to propose seed words. For a given collection and for each class k of the collection, all words w in the vocabulary are scored according to:

$$\text{score}(w,k) = \left(\text{tf}_k(w) - \frac{1}{K-1} \sum_{\substack{k'=1 \\ k' \neq k}}^{K} \text{tf}_{k'}(w) \right) \times \text{idf}(w), \tag{5.12}$$

where $\text{idf}(w)$ is the inverse document frequency computed on the documents of the whole collection and $\text{tf}_k(w)$ is the term frequency for class k, which we define as the sum of $\text{tf}_x(w)$ for all documents x in class k. The rationale for this score is that one wishes to select words that are frequent in class k and unfrequent in other classes, hence the penalization term inside the brackets. Based on this score, one can then select the top words for each class as seed words. We emphasize that such heuristics is only adopted for the purpose of simulating seed words during the evaluation: it is not destined to be used to identify seed words in a real-world application, where ground truth is unknown. In our preliminary studies, we have generated five seed words per cluster while in most cases we have used only three seed words per cluster. The rest of seed words have been used for more evaluations and analysis towards our proposed framework. Table 5.1 shows SD2C: Manual seed words selection the obtained seed words for each dataset.

[8] Pairwise constraints are not utilized in these approaches.

Table 5.1: Automatically selected seed words for each dataset. Each row corresponds to each class of the dataset while seed words pertaining to one class are sorted from the most relevant ones to the less relevant ones.

Dataset	seed words
20NEWS	atheist god moral islam livesey graphic imag file format program window file driver microsoft ms drive card scsi ide monitor mac appl monitor drive quadra window motif server widget xterm sale offer ship sell condit car engin ford auto oil bike dod ride motorcycl bmw game basebal pitch player team game team hockey play player key encrypt clipper chip secur circuit batteri electron wire power pitt doctor medic gordon food space nasa orbit henri moon god christian church jesu sin gun fbi stratu atf weapon israel armenian arab jew muslim cramer optilink gay homosexu clayton christian sandvik god jesu moral
REUTERS	share compani dlr acquir pct coffe quota export ico bag oil crude barrel price opec net shr loss dlr rev gold ounc ton feet coin rate pct bank prime cut bank stg currenc market dollar ship port strike vessel gulf sugar tonn white trader intervent trade japan billion export japanes
YAHOO!	god peopl christian jesu believ water earth number answer time doctor weight eat day pain school word colleg mean answer comput yahoo file download window team win game play cup job compani money work busi song movi music love favorit love guy girl friend feel peopl countri bush presid state
DBPEDIA	compani base oper servic product school high colleg univers locat born singer american known music footbal play born leagu profession politician repres member born serv navi class ship built aircraft histor church build hous locat river lake mountain tributari romania villag district popul counti provinc famili speci moth genu snail speci plant famili genu flower album releas record band studio film direct star drama comedi publish novel book journal seri
AGNEWS	kill iraq presid minist afp game win team season night oil stock price compani profit new microsoft softwar internet compani

5.4.4.2 SD2C: manual seed words selection

Evaluating algorithms in real case scenarios can be useful to analyze them better. Selecting seed words manually and comparing its results versus other

algorithms provides us enough insights towards SD2C. Manually selecting seed words from thousands of documents can be difficult and frustrating. Also, since selecting seed words from different datasets can become a tedious task, we decided to focus only on DBPEDIA dataset and design two different user experiments tasks to withdraw seed words with the help of different users. In order to perform the task, we selected only five classes of this dataset (containing 25,000 documents in total) to make the user experiments more feasible. We have designed two different experiments which are detailed below.

Using K-means for subsampling the documents. Presenting 25,000 already-selected documents for users to extract seed words from seems impossible. To deal with this problem, we have used the k-means algorithm to select the most discriminative and informative documents. Indeed, without using any class information, K-means has been applied on the whole collection (containing 25,000 documents) where K (number of clusters) has been set to 20 (the value of K is an hyperparameter and one can choose a different value). Then, the n closest documents to each cluster center have been selected. Finally, by setting $n = 10$, we ended up with 200 documents. In our preliminary studies we noted that this approach yields representative documents where all classes have been covered. Moreover, this approach can lead towards an imbalance in choosing different classes. In fact, in the beginning of experiments, the input contained 25,000 documents equally distributed in five different classes while utilizing this approach yields $K \times n$ (in this case 200) documents which are not equally distributed among all classes. Moreover, since documents can be quite long, we kept only the first sentence of each document to present them to the users. Finally, these documents have been presented to different users for selecting seed words. In this experiment, five users participated in this case study and were asked to first identify different classes and second provide three seed words per class. More detailed information about the seed words that have been selected from each user and the time consumption of the experiment for different users are given in Table 5.2. Note that to avoid biasing the users by the order of the presented documents obtained from K-means, we shuffled the documents for each user.

Using Latent Dirichlet Allocation (LDA) for extracting seed words. Instead of presenting the first sentence of each document and present them to the users, one might use LDA on the original collection (containing 25,000 documents) and use LDA to extract relevant seed words for each detected

Table 5.2: The obtained seed words by presenting the documents obtained by K-means for each user alongside the dedicated time (in seconds) to finish the experiment. Please note that the provided seed words for each users are shown such that each line of seed words pertains to one class.

Users	seed words	Time (in Seconds)
#1	book newspaper,journal company,business technology film comedy history school,university college aircraft navy war	594
#2	car motor automotive Film cinema drama navy ship war education school college newspaper novel,book	600
#3	film movie direct school university college company,industry job ship aircraft tank write publish book	623
#4	newspaper book journal company,business,industry flim hollywood bollywood university college school navy ship aircraft	698
#5	school college university book newspaper magazin fiction comedy movie technology business industry airplane submarine car	1061
Average Time Consumption	N/A	715.2

topic. LDA is one of the most well-known and widely used algorithm in topic modeling [137] which is mostly used in unsupervised settings. In our experiments, we fed the whole collection to LDA to extract the relevant seed words. In this case, we can simply get rid of documents and present seed words to the users instead. The number of topics has been set to 20 and the number of seed words for each topic is set to 10. Finally, LDA will produce **200** seed words in total to be presented to participants. Similarly to the K-means solution mentioned above, we asked the users to first get through all seed words and identify five different classes and then provide three seed words pertaining to each class. Again, similar to K-means approach mentioned above, five users participated in this case study as well. Table 5.3 shows more detailed information about selected seed words and required time for each user to finish the experiment. Note that to avoid biasing the users by the order of the presented seed words obtained from LDA, we shuffled the seed words for each user.

Table 5.3: The obtained seed words by providing seed words from LDA for each user alongside the dedicated time (in seconds) to finish the experiment. Note that the provided seed words for each users are shown such that each line of seed words pertains to one class.

Participants	Seed words	Time (in seconds)
#1	film drama comedy publish journal novel class school college navy ship uss company technology produce	1502
#2	film play series school university faculty book novel journal cars train ship war aircraft navy	554
#3	film comedy series boston virginia vegas car volvo airplane college elementary university company found public	302
#4	company samsung volvo school college university film action comedy japanese american english dockyard navy ship	581
#5	united american state american german bangladeshi book stories publish aircraft airline international college university class	384
Average Time Consumption	N/A	664.6

5.4.4.3 PCD2C: Automatic Pairwise Constraints Selection

To provide pairwise constraints, we have assumed that there are few labeled data that can be used in the experiments such that pairs with the same labels are considered to be must-link pairs and pairs with different labels are considered to be cannot-link pairs. In this case, we have used different numbers of pairs to measure the effectiveness of increasing this number in our experiments. In our experiments we have used 50, 100, 200, 500 and 1000 pairs. We started by randomly selecting 50 pairs of documents and assigning them to must-link or cannot-link (by using the true cluster labels). Then, to obtain 100 pairs, 50 already generated pairs are used as a base and 50 more pairs are randomly selected and added to the current 50 pairs. The same process is repeated iteratively to obtain 200, 500 and 1000 pairs.

5.4.4.4 PCD2C: Manual Pairwise Constraints Selection

In order to validate our proposed framework in a real case scenario, we have designed a user-based experiment including pairwise constraints. Similar to the automatic pairwise constraints selection described above, we started the process of randomly selecting the pairs of documents with 50 pairs. Then the same iterative process has been used to generate 100 and 300 pairs of documents. These pairs are shown to users for further annotation. Again, five users participated in this experiments and they were asked to assign must-link or cannot-link constraints to each pair. The pairs of documents have been shuffled for each user to avoid biasing the users based on the order of pairs. More details can be found in Table 5.4.

Table 5.4: Detailed information about number of must-link and cannot-link pairs for each user. The dedicated time to finish the experiment for each number of pairs is shown in seconds.

Users	#Pairs	#Must-links	#Cannot-links	Time (in Seconds)
#1	50	4	46	600
	100	11	89	1140
	300	28	272	2940
#2	50	4	46	584
	100	6	94	1129
	300	21	279	3012
#3	50	2	48	739
	100	7	93	1433
	300	16	284	3697
#4	50	5	45	671
	100	12	88	1295
	300	21	279	3442
#5	50	7	43	660
	100	17	83	1380
	300	37	273	3540
Using True Labels (No User Experiments)	50	10	40	N/A
	100	19	81	N/A
	300	53	247	N/A
Average time consumption	50	N/A	N/A	650.8
	100	N/A	N/A	1275.4
	300	N/A	N/A	3326.2

5.4.5 Experimental Setup

Following prior deep clustering works [29, 31, 85, 136], we initialize the Autoencoder parameters through pretraining by first only optimizing the reconstruction loss of the Autoencoder. In the pretraining of SD2C-Doc, we also include the contraint-enforcing term (second term in Problem 5.3) so that learned representations are impacted by seed words early in the training. At the end of pretraining, the cluster centers are initialized by the seed

words cluster embeddings $\{\mathbf{s}_k\}_{k=1}^{K}$.[9] Then, in the fine-tuning phase, the whole loss – including the clustering loss and the constraint-enforcing loss (for SD2C-Doc and SD2C-Rep) – is optimized. The same pretraining has been done for PCD2C except that the cluster representatives are initialized by performing k-means on the learned representations obtained from the Autoencoder (similar to [29, 31, 85, 136]).

Architecture and hyperparameters. The Autoencoder used in our experiments on all datasets is similar to the ones adopted in prior deep clustering works [29, 31, 85, 136]. The encoder and decoder are mirrored fully-connected neural networks with dimensions d-500-500-2000-50 and 50-2000-500-500-d, respectively – d is the input space dimension and 50 corresponds to the dimension p of the Autoencoder embedding space. Neural networks' weights are initialized based on the Xavier scheme [125]. The SD2C, DKM, and AE-KM models are trained with the Adam optimizer [126] with standard hyperparameters ($\eta = 0.001$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$) and minibatches of 256 documents. The number of epochs for the Autoencoder pretraining and model finetuning are fixed to 50 and 200, respectively, as in [136]. We also use the inverse temperature $\alpha = 1000$ from [136] for the parameterized softmax-based differentiable reformulations of SD2C models. The balancing hyperparameters λ_0 and λ_1 of SD2C-Doc and SD2C-Rep (e.g., DKM-Doc/Rep, IDEC-Doc/Rep, DCN-Doc/Rep) were both set to 10^{-5}–noted as general hyperparameters.

As mentioned in the previous chapter, we trained a Word2Vec model on each collection individually.

5.4.6 Clustering Results

5.4.6.1 SD2C with Automatic Seed Word Selection

We measure the clustering performance in terms of clustering accuracy (ACC), normalized mutual information (NMI) and adjusted rand index (ARI), which are standard clustering metrics [127]. Table 5.5 first provides the macro-average (over the 5 datasets) of these measures for all methods, using the top 3 automatically selected seed words per cluster. As one can note, the use of seed words is beneficial to the clustering. Indeed, the approaches which use seed words (NN, STM, SD2C (DKM-based)) have markedly higher

[9] This is especially important for SD2C-Rep which is based on the assumption that the clusters defined by the seed words and those defined by the cluster representatives are aligned.

ACC, NMI and ARI than those which do not (KM, AE-KM, DKM, DCN (e/c), IDEC (e/c)). Among these latter methods, DKM is the best one (as a comparison, DCN and IDEC, mentioned in Section 5.2, respectively obtain 64.8 and 64.1 for ACC, and 49.3 and 47 for ARI). Among the methods exploiting seed words, SD2C methods are the best ones, outperforming the baseline NN and the STM method by up to 2.6 points for ACC and 3.5 points for ARI. DKM-based SD2C approaches obtain quite higher results compared to DCN-based and IDEC-based SD2C approaches. Indeed, adding constraints and using general hyperparameters for these approaches seem ineffective.

We further provide in Table 5.6 a detailed account of the performance of the methods based on seed words. The results have been averaged over 10 runs and are reported with their standard deviation. We furthermore performed an unpaired Student t-test with a significance level of 0.01 to study whether differences are significant or not (all results in bold are not statistically different from the best result). As one can note, the proposed DKM-based SD2C models compare favorably against STM, the strongest baseline. Indeed, all DKM-based SD2C approaches significantly outperform STM on 20NEWS, and DKM-Doc-e/c as well as DKM-Rep-e also significantly outperform STM on YAHOO and AGNEWS. On the other hand, STM obtained significantly better results in terms of both ACC and ARI on REUTERS and DBPEDIA, the difference on these collections (and especially on DBPEDIA) being nevertheless small. Among the DKM methods, DKM-Doc-c yields the best performance overall (as shown in Table 5.5). Again, we can notice that DKM-based SD2C approach obtains significantly better results compared to DCN-based and IDEC-based approaches.

Impact of the number of seed words. In our general setting used to report the previous results, the number of seed words per class was arbitrarily set to 3. For comprehensiveness, we study the clustering results of the DKM-based SD2C models when the number of (automatically selected) seed words per cluster is varied from 1 to 5. The evolution of the performance for the SD2C models in terms of accuracy is illustrated in Figure 5.4. We observe that using more seed words leads to notable improvements in most cases. This trend is particularly apparent when the number of seed words is increased from 1 to 2. Although slight performance gain is observed between 2 and 5 seed words, the results exhibit greater stability. This suggests that providing as few as 2 seed words per cluster – which constitutes a modest annotation effort for humans – can prove highly beneficial for the clustering results obtained by our DKM-based SD2C approaches.

Table 5.5: Macro-average results in terms of accuracy (ACC), normalized mutual information (NMI) and adjusted rand index (ARI). The double vertical line separates approaches which leverage seed words (right) from approaches which do not (left). Bold values correspond to the best results.

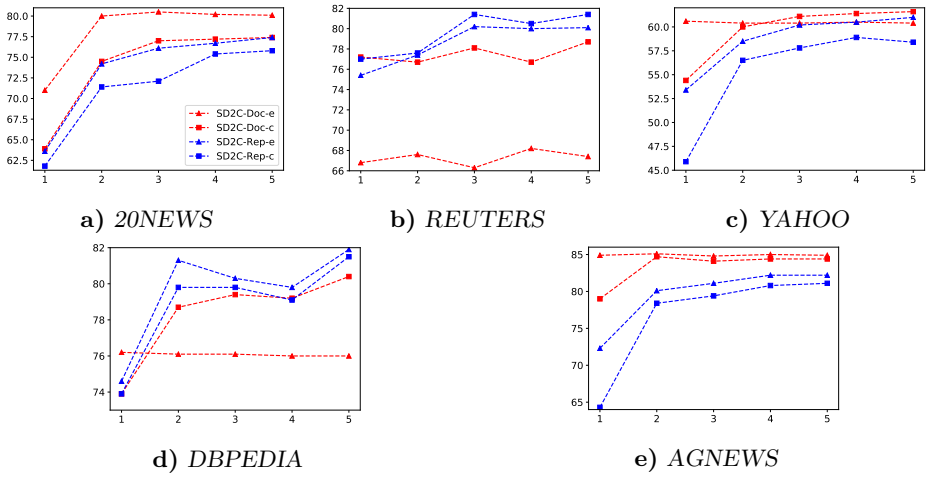
Model	ACC	ARI	NMI
KM	61.4	45.4	59.0
AE-KM	64.0	48.8	59.3
DKM-e	65.5	50.4	59.2
DKM-c	65.0	48.9	58.6
DCN-e	64.8	49.3	59.3
DCN-c	64.5	47.6	59.0
IDEC	64.1	47.0	58.4
NN	72.6	50.9	58.1
STM	73.3	53.6	57.6
DKM-Doc-e	73.6	56.2	61.5
DKM-Doc-c	**75.9**	**57.1**	61.0
DKM-Rep-e	75.6	55.7	59.1
DKM-Rep-c	74.1	53.5	58.5
DCN-Doc-e	56.1	37.4	51.7
DCN-Doc-c	61.4	43.8	54.2
DCN-Rep-e	52.7	33.0	44.5
DCN-Rep-c	63.8	46.9	58.8
IDEC-Doc-e	45.1	19.6	34.7
IDEC-Doc-c	45.8	18.8	34.8
IDEC-Rep-e	46.1	18.4	35.0
IDEC-Rep-c	37.5	17.2	32.3

Table 5.6: Seed-guided constrained clustering results on DBPEDIA and AGNEWS with 3 seed words per cluster. Bold results denote the best, as well as not significantly different from the best, results. Italicized SD2C results indicate a significant improvement over STM.

Model	DBPEDIA			AGNEWS		
	ACC	ARI	NMI	ACC	ARI	NMI
NN	79.9±0.0	64.7±0.0	75.7±0.0	77.3±0.0	51.9±0.0	51.1±0.0
STM	**80.9±0.4**	**72.7±0.4**	**79.1±0.2**	79.7±0.2	55.8±0.3	52.5±0.2
DKM-Doc-e	76.1±0.2	63.0±0.2	72.8±0.3	*84.8±0.2*	*64.4±0.5*	60.1±0.4
DKM-Doc-c	79.4±1.9	66.3±2.1	76.0±0.8	*84.1±1.1*	63.3±2.3	*59.9±1.5*
DKM-Rep-e	*80.3±0.5*	67.0±0.6	75.0±0.4	81.1±0.3	58.1±0.5	54.7±0.4
DKM-Rep-c	*79.8±1.4*	66.1±1.8	74.9±0.9	79.4±2.0	55.0±3.6	53.4±2.0
DCN-Doc-e	52.7±4.5	34.4±3.8	55.6±2.4	80.1±1.3	58.8±3.2	58.4±2.3
DCN-Doc-c	66.2±3.5	54.1±3.2	69.4±1.2	*84.2±1.1*	63.1±2.4	*59.1±1.9*
DCN-Rep-e	53.1±3.4	38.6±3.0	56.7±2.3	71.7±4.0	41.3±6.9	39.4±6.0
DCN-Rep-c	68.0±3.2	57.1±4.1	73.2±1.5	**85.1±0.8**	**65.0±1.8**	**61.1±1.5**
IDEC-Doc-e	39.5±7.0	19.5±5.8	40.5±4.5	39.8±6.5	6.2±5.5	16.5±5.4
IDEC-Doc-c	41.1±5.7	20.6±5.7	42.1±3.5	46.7±7.6	12.6±10.2	23.0±6.8
IDEC-Rep-e	41.8±6.1	21.2±4.7	42.9±3.4	48.5±6.3	13.1±7.0	24.1±5.9
IDEC-Rep-c	27.4±5.9	14.3±6.4	29.3±4.1	44.2±8.3	10.5±9.2	19.7±7.1

Table 5.7: Seed-guided constrained clustering results on 20NEWS, REUTERS and YAHOO with 3 seed words per cluster. Bold results denote the best, as well as not significantly different from the best, results. Italicized SD2C results indicate a significant improvement over STM.

Model	20NEWS			REUTERS			YAHOO		
	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI
NN	72.3±0.0	53.2±0.0	67.2±2.0	79.0±0.0	58.3±0.0	61.3±0.0	54.7±0.0	26.5±0.0	35.0±0.0
STM	65.7±0.9	47.5±1.0	58.3±0.9	**83.0±0.7**	**66.3±1.2**	**64.4±0.6**	57.1±0.1	29.3±0.2	33.8±0.1
DKM-Doc-e	**80.5±0.6**	**66.4±0.7**	**73.0±0.5**	66.3±3.7	53.0±3.2	62.1±1.2	*60.4±0.3*	*34.3±0.3*	39.5±0.2
DKM-Doc-c	77.0±1.5	61.7±2.2	70.0±1.3	78.1±1.8	60.2±1.1	58.7±0.9	**61.1±0.8**	**34.4±1.3**	**40.4±0.3**
DKM-Rep-e	76.1±0.3	60.1±0.5	68.3±0.2	80.2±0.8	59.9±0.8	59.8±1.1	*60.2±0.3*	*33.5±0.4*	37.8±0.3
DKM-Rep-c	72.1±1.5	55.7±1.7	65.8±1.0	81.4±0.7	61.1±0.9	62.0±1.0	57.8±1.7	29.7±2.7	36.8±0.7
DCN-Doc-e	53.8±5.1	34.1±3.9	52.8±2.7	41.1±3.3	30.4±2.6	54.1±1.9	53.1±2.8	29.4±1.7	37.6±0.8
DCN-Doc-c	60.4 ±4.3	42.2±3.4	57.9±2.2	40.9±4.3	29.5±4.0	47.2±2.6	55.6±3.1	30.2±1.3	37.8±0.9
DCN-Rep-e	55.9±2.1	36.6±1.9	53.1±1.1	38.5±2.0	28.1±2.4	46.1±2.1	44.6±2.6	20.7±2.0	27.4±1.5
DCN-Rep-c	66.4±2.7	48.5±2.4	65.4±1.2	41.9±4.7	32.3±4.7	54.6±2.7	57.7±3.8	31.6±1.9	39.8±1.1
IDEC-Doc-e	44.1±.9	26.9±2.8	47.1±1.8	69.6±2.9	36.1±5.3	49.3±2.4	32.9±3.8	9.5±3.1	20.1±3.2
IDEC-Doc-c	36.9±7.1	13.7±5.5	39.9±4.4	69.2±3.0	35.3±4.7	47.2±3.0	35.3±4.7	11.8±3.8	21.8±3.3
IDEC-Rep-e	36.3±7.0	13.4±5.4	39.3±4.5	68.4±2.8	33.5±6.5	46.9±3.0	35.6±3.6	11.0±4.1	21.8±3.1
IDEC-Rep-c	32.5±5.9	11.2±6.1	38.7±5.5	38.8±3.1	29.2±4.8	46.3±3.7	44.9±3.4	21.2±2.6	27.5±1.1



a) *20NEWS* **b)** *REUTERS* **c)** *YAHOO*

d) *DBPEDIA* **e)** *AGNEWS*

Figure 5.4: Clustering results in terms of ACC for SD2C-Doc-e, SD2C-Doc-c, SD2C-Rep-e and SD2C-Rep-c with 1 to 5 seed words.

5.4.6.2 SD2C with Human Selected Seed Words

The results of different SD2C approaches on the subset of DBPEDIA (including 25,000 documents equally distributed into 5 different classes) are shown in Table 5.9 and Table 5.11 (on this subset, the results of multiple clustering and Deep Clustering algorithms are shown in Table 5.8). The results indicate that using DKM as the choice of clustering method leads towards more beneficial results compared to DCN and IDEC. Moreover, the DKM-Doc-e approach is able to obtain better results compared to the case where seed words are not being used in the framework (e.g., DKM). The results of using seed words obtained by presenting LDA-seed words to different users also indicate that DKM-Doc-e outperforms other baselines. They however suggest

Table 5.8: Results of different clustering approaches on selected documents from DBPEDIA dataset–without using constraints. Bold results represent the best results.

Metric	KM	AE-KM	DKM-e	DKM-c	DCN-e	DCN-c	IDEC
ACC	83.0±0.0	**84.3±0.3**	83.6±0.5	84.1±1.0	84.0±0.4	84.0±0.6	83.2±0.3
ARI	65.9±0.0	**68.5±0.6**	66.4±0.8	67.6±1.5	67.8±0.6	67.8±1.1	67.1±0.5
NMI	70.7±0.0	**71.6±0.5**	68.3±0.7	69.9±1.1	70.3±0.6	70.3±0.8	69.9± 0.7

that the improvement gained by using these seed words compared to the case where no seed words are used (e.g., DKM) is quite negligible. In fact, the results obtained from presenting selected documents (through K-means) to the users outperform the results obtained from presenting LDA-seed words to the users. Based on the time consumption required to fulfill the experiments for these two approaches which are presented in Table 5.2 and Table 5.3, there are no significant difference in terms of required time to extract seed words from these approaches but presenting the documents (through K-means) obtains significantly better results compared to presenting LDA-seed words.

Table 5.9: Results of different deep clustering with the constraints obtained by presenting the first sentence of each document (obtained by applying k-means). Bolded results represent the best results and italicized results are statistically equivalent to the best results (p=0.01).

Model	kw1			kw2			kw3		
	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI
DKM-Rep-e	*84.9±1.2*	*67.4±2.3*	66.1±2.0	76.3±2.6	55.9±3.7	58.6±3.4	80.4±1.4	61.7±1.8	62.0±1.8
DKM-Rep-c	*84.5±2.2*	*66.7±3.8*	66.3±2.9	74.1±2.9	52.7±3.6	57.0±3.0	81.4±2.2	61.9±3.8	62.5±2.9
DKM-Doc-e	*85.2±1.6*	*69.2±2.6*	*69.3±2.0*	**85.1±1.7**	**69.0±2.5**	**69.1±1.7**	*85.0±1.6*	*68.8±2.4*	*69.0±2.1*
DKM-Doc-c	*85.8±2.3*	*69.3±4.1*	*69.2±2.6*	74.7±4.7	54.8±5.6	60.1±3.7	**86.3±1.6**	**70.6±2.4**	**70.2±1.7**
DCN-Rep-e	40.3±0.4	6.7±0.3	6.9±0.3	36.9±0.6	5.3±0.3	5.6±0.3	39.0±0.5	6.3±0.2	6.6±0.2
DCN-Rep-c	40.0±0.4	6.5±0.2	6.8±0.2	36.9±0.8	5.2±0.4	5.6±0.4	39.2±0.8	6.4±0.3	6.7±0.3
DCN-Doc-e	38.3±1.6	6.1±0.6	6.3±0.3	35.9±2.3	5.1±0.8	5.4±0.8	37.0±1.7	6.2±0.6	6.3±0.6
DCN-Doc-c	**86.0±1.9**	**69.7±3.3**	**69.5±2.3**	73.6±5.3	53.9±6.3	59.5±4.4	*86.1±1.6*	*70.2±2.1*	*69.7±1.7*
IDEC-Rep-e	39.1±8.1	6.9±6.7	22.6±8.1	44.6±5.4	14.9±6.7	30.3±5.6	35.2±5.3	4.9±4.3	20.9±4.3
IDEC-Rep-c	37.1±9.0	8.1±8.3	20.4±7.6	40.0±4.1	9.8±5.0	24.8±4.2	28.1±2.6	1.6±1.2	15.3±3.5
IDEC-Doc-e	34.6±5.6	4.1±3.2	19.5±4.7	47.8±7.3	20.3±9.6	33.7±6.4	39.5±7.1	9.8±7.5	26.8±6.3
IDEC-Doc-c	43.0±8.1	12.1±8.4	25.0±6.8	41.1±7.5	11.6±8.7	25.9±7.4	27.5±2.2	1.3±0.8	15.0±3.1

Table 5.10: Results of different deep clustering with the constraints obtained by presenting the first sentence of each document (obtained by applying K-means). Bolded results represent the best results and italicized results are statistically equivalent to the best results (p=0.01).

Model	kw4			kw5			Macro-AV		
	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI
DKM-Rep-e	83.0±1.7	63.9±3.3	62.9±2.7	79.5±2.2	59.7±3.2	61.7±2.2	80.8	61.7	62.2
DKM-Rep-c	81.5±2.4	60.9±3.8	61.2±2.7	77.6±2.3	57.4±2.6	60.4±1.5	79.8	59.9	61.4
DKM-Doc-e	*84.4±1.6*	*68.2±2.5*	*68.9±1.9*	**84.6±1.1**	**68.4±1.5**	69.0±1.2	**85.0**	**68.7**	**69.0**
DKM-Doc-c	**85.4±1.1**	**68.8±2.0**	**69.2±1.2**	*83.1±2.7*	65.2±4.5	66.5±3.4	83.0	65.7	67.0
DCN-Rep-e	39.5±0.4	6.2±0.5	6.5±0.2	38.1±0.7	5.9±0.3	6.2±0.3	38.7	6.0	6.3
DCN-Rep-c	38.8±0.6	5.9±0.3	6.1±0.3	37.7±0.4	5.8±0.1	6.0±0.1	38.5	5.9	6.2
DCN-Doc-e	37.9±2.3	6.1±0.8	6.3±0.8	36.1±1.1	5.8±0.8	6.0±0.6	37.0	5.8	6.0
DCN-Doc-c	*84.9±1.7*	*67.9±2.8*	*68.6±1.6*	*83.5±3.2*	*66.2±5.0*	*67.3±3.6*	82.8	65.5	66.9
IDEC-Rep-e	29.4±3.3	1.3±1.0	14.3±3.7	49.7±0.7	22.8±8.2	32.8±5.8	39.6	10.1	24.1
IDEC-Rep-c	33.4±4.9	4.8±5.1	15.4±4.3	41.1±9.5	16.0±12.4	25.8±11.3	35.9	8.0	20.3
IDEC-Doc-e	26.7±1.9	0.8±0.5	12.6±2.7	48.9±5.5	23.0±4.9	33.7±3.8	39.5	11.6	25.2
IDEC-Doc-c	36.5±3.9	6.8±4.4	18.7±3.7	44.5±6.5	20.0±9.8	29.1±9.3	38.5	10.3	22.7

Table 5.11: Results of different deep clustering approaches with the constraints obtained by presenting the seed words obtained from LDA. Bolded results represent the best results and italicized results are statistically equivalent to the best results (p=0.01).

Model	kw1			kw2			kw3		
	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI
DKM-Rep-e	*83.3±1.1*	65.0±2.1	64.2±2.3	76.1±2.9	55.7±2.9	58.6±2.3	60.5±1.4	40.0±1.0	44.4±1.3
DKM-Rep-c	*83.5±1.6*	65.2±2.6	65.5±1.9	76.4±3.1	56.3±3.4	59.8±2.7	59.3±2.3	41.7±2.1	47.4±1.8
DKM-Doc-e	**84.6±1.3**	**68.3±2.1**	**68.9±1.7**	**84.3±1.4**	**67.9±2.3**	**68.7±2.0**	**82.7±5.4**	**66.2±7.0**	**67.7±5.0**
DKM-Doc-c	*83.8±1.9*	*66.8±3.2*	*68.0±2.5*	**84.3±2.1**	*67.3±3.3*	*68.2±2.0*	63.0±7.3	48.3±6.7	56.1±4.4
DCN-Rep-e	39.9±0.5	6.4±0.3	6.7±0.3	36.6±0.6	5.3±0.2	5.6±0.2	32.7±0.5	3.9±0.2	4.3±0.2
DCN-Rep-c	40.1±0.7	6.5±0.4	6.8±0.3	38.6±0.6	5.8±0.6	6.1±0.2	32.3±0.7	4.1±0.3	4.5±0.3
DCN-Doc-e	38.4±1.4	5.9±0.6	6.2±0.6	36.4±2.1	5.8±0.9	6.0±0.8	34.0±1.2	4.9±0.5	5.1±0.5
DCN-Doc-c	*83.5±1.0*	*66.6±1.7*	*68.2±1.4*	*83.8±3.8*	*65.8±5.0*	*66.9±3.3*	63.9±6.2	49.2±5.0	56.7±3.4
IDEC-Rep-e	59.0±7.8	26.8±9.4	38.0±7.1	55.8±6.8	31.3±9.0	41.1±6.7	46.1±5.1	20.2±5.9	27.0±4.3
IDEC-Rep-c	44.6±5.2	13.1±7.4	24.6±5.0	39.5±7.4	11.8±8.7	25.5±7.8	31.8±4,2	3.9±2.5	10.7±2.9
IDEC-Doc-e	59.3±9.8	29.9±11.3	39.3±9.0	54.0±7.8	29.1±8.4	38.9±6.0	46.4±5.3	20.3±6.6	28.4±4.9
IDEC-Doc-c	40.1±9.5	9.5±8.9	23.1±8.4	43.0±9.8	16.8±10.6	28.2±10.1	32.3±2.8	3.8±1.8	12.5±2.7

Table 5.12: Results of different deep clustering approaches with the constraints obtained by presenting the seed words obtained from LDA. Bolded results represent the best results and italicized results are statistically equivalent to the best results (p=0.01).

Model	kw4			kw5			Macro-AV		
	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI
DKM-Rep-e	78.0±2.3	56.7±3.6	58.4±3.7	59.2±6.1	33.6±6.4	37.0±6.0	71.4	50.2	52.5
DKM-Rep-c	74.3±4.6	52.3±4.9	55.8±3.7	59.2±3.7	37.5±3.5	42.9±2.6	70.5	50.6	54.2
DKM-Doc-e	**85.0±1.2**	**68.9±1.7**	**69.4±1.3**	**81.8±6.9**	**65.2±8.9**	**67.1±5.7**	**83.6**	**67.3**	**68.3**
DKM-Doc-c	*83.7±1.3*	*66.8±2.1*	*68.1±1.9*	74.8±7.6	56.7±8.2	59.6±6.4	77.9	61.1	64.0
DCN-Rep-e	37.9±0.7	5.6±0.3	5.9±0.3	30.3±0.9	2.6±0.3	2.9±0.4	35.4	4.7	5.0
DCN-Rep-c	37.4±0.8	5.4±0.4	5.7±0.4	31.3±1.2	3.1±0.6	3.5±0.5	35.9	4.9	5.3
DCN-Doc-e	36.4±2.0	5.7±0.7	5.9±0.7	32.6±2.4	4.5±0.8	4.8±0.8	35.5	5.8	5.6
DCN-Doc-c	*84.1±1.8*	*67.0±3.0*	*68.0±2.2*	78.4±5.2	59.8±6.7	61.9±5.2	79.6	61.6	64.3
IDEC-Rep-e	49.8±5.2	20.6±5.2	30.4±5.3	36.9±6.5	8.7±5.2	17.9±5.7	49.5	21.5	30.8
IDEC-Rep-c	38.1±4.7	8.7±4.2	20.8±4.1	34.0±6.8	6.6±5.8	14.6±5.5	37.6	8.8	19.2
IDEC-Doc-e	49.9±5.5	18.4±0.5	30.3±3.9	31.8±5.2	4.6±4.0	13.8±3.9	48.2	20.4	30.1
IDEC-Doc-c	36.9±4.0	8.1±4.6	20.7±3.9	32.8±5.2	5.9±4.3	13.0±4.9	37.0	8.8	19.5

5.4.6.3 PCD2C Using Groundtruth-based ML/CL Constraints

Table 5.13: Results of PCD2C framework (using DKM) on 20NEWS, REUTERS and YAHOO with general hyperparameters.

Model	20NEWS			REUTERS			YAHOO		
	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI
DKM-e(50)	**75.0 ±2.0**	**62.5±1.4**	**71.6 ± 0.6**	*42.7 ±1.3*	*33.6±0.8*	*55.8 ± 1.3*	*54.7 ±2.5*	*31.6±1.1*	*37.8 ±0.7*
DKM-e(100)	*73.5 ±1.8*	*61.7±1.4*	*71.1 ± 0.9*	*43.3 ±1.7*	*33.9±1.0*	**56.3 ± 1.1**	**56.0 ±3.1**	*32.2±1.5*	*38.1 ±0.9*
DKM-e (200)	*73.7 ±3.1*	*61.8±1.6*	*71.0 ± 1.1*	**43.8 ±3.6**	**34.2±3.0**	*56.1 ± 1.3*	*56.4 ±2.7*	**32.5±1.3**	**38.4 ± 0.8**
DKM-e(500)	*73.8 ±4.2*	*61.7±1.1*	*71.2 ± 1.2*	*42.5 ±0.9*	*33.6±0.9*	*55.9 ± 1.0*	*55.7 ±2.8*	*32.1±1.1*	*38.1±0.7*
DKM-e(1000)	*74.2 ±2.8*	*62.0±1.3*	*71.2 ± 0.9*	*42.6 ±2.0*	*33.0±1.5*	*55.5 ± 0.9*	*55.5±2.9*	*31.9±1.4*	*38.0±0.8*

Table 5.14: Results of PCD2C framework (using DKM) on DBPEDIA and AGNEWS with general hyperparameters.

Model	DBPEDIA			AGNEWS			Macro-average		
	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI
DKM-e(50)	*67.9±2.6*	58.7±3.2	*72.3±1.5*	*85.2±0.4*	*65.4±0.7*	*61.0±0.6*	65.1	50.3	59.7
DKM-e(100)	**68.2±3.4**	**59.5±3.7**	**72.9±1.5**	**85.5±0.4**	**65.9±0.7**	**61.4±0.6**	65.3	**50.6**	**59.9**
DKM-e (200)	*67.8±2.2*	*58.9±3.5*	*72.6±1.7*	*85.3±0.5*	*65.6±1.0*	*61.3±0.8*	**65.4**	**50.6**	59.8
DKM-e(500)	*66.1±2.5*	*56.9±3.3*	*71.7±1.6*	*85.3±0.5*	*65.6±1.0}*	*61.2±0.9*	64.6	49.9	59.6
DKM-e(1000)	*67.5±1.7*	*58.0±3.0*	*72.3±1.3*	*85.4±0.5*	**65.9±1.0**	*61.5±0.8*	65.0	50.1	59.7

Similar to the SD2C approach, one may select different deep clustering algorithms in PCD2C framework. The proposed PCD2C framework has been

Table 5.15: Results of PCD2C framework (using DKM) on 20NEWS, REUTERS and YAHOO by tuning λ_1 hyperparameter (λ_0 has been set to 10^{-5}).

Model	20NEWS			REUTERS			YAHOO		
	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI
DKM-e(50)	**75.6 ±3.1**	**62.5±1.4**	*71.7 ± 0.9*	*80.8 ±4.6*	83.1±3.0	69.1 ± 2.5	54.7 ±2.5	*31.6±1.1*	37.8 ±0.7
DKM-e(100)	*75.3 ±3.5*	**62.5±2.9**	*71.8 ± 1.0*	*82.7 ±2.7*	84.2±2.4	69.8 ± 3.2	56.0 ±3.1	*32.2±1.5*	38.1 ±0.9
DKM-e(200)	*74.3 ±2.3*	*62.1±1.7*	*71.9 ± 0.7*	*84.4 ±2.0*	86.4±2.4	70.8 ± 2.7	**56.4 ±2.7**	**32.5±1.3**	**38.4 ± 0.8**
DKM-e(500)	*73.9 ±2.1*	*61.9±2.1*	*71.6 ± 0.7*	*83.7 ±3.6*	*86.5±4.6*	72.1 ± 3.4	55.7 ±2.8	*32.1±1.1*	38.1±0.7
DKM-e(1000)	*74.8 ±3.6*	*62.2±1.8*	**72.0 ± 1.1**	**85.2 ±1.8**	**89.9±1.4**	**76.3 ± 2.1**	55.5±2.9	*31.9±1.4*	38.0±0.8

Table 5.16: Results of PD2C framework (using DKM) on DBPEDIA and AGNEWS datasets by tuning λ_1 hyperparameter (λ_0 has been set to 10^{-5}).

Model	DBPEDIA			AGNEWS			Macro-average		
	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI
DKM-e(50)	*67.9±2.6*	*58.7±3.2*	*72.3±1.5*	*85.2±0.4*	*65.4±0.7*	*61.0±0.6*	72.8	60.1	62.3
DKM-e(100)	**68.2±3.4**	**59.5±3.7**	**72.9±1.5**	**85.5±0.4**	**65.9±0.7**	*61.4±0.6*	73.5	60.8	62.8
DKM-e(200)	*67.8±2.2*	*58.9±3.5*	*72.6±1.7*	*85.3±0.5*	*65.6±1.0*	*61.3±0.8*	**73.6**	61.1	63.0
DKM-e(500)	*66.1±2.5*	*56.9±3.3*	*71.7±.16*	*85.3±0.5*	*65.6±1.0*	*61.2±0.9*	72.9	60.6	62.9
DKM-e(1000)	*67.5±1.7*	*58.0±3.0*	*72.3±1.3*	*85.4±0.5*	**65.9±1.0**	**61.5±0.8**	**73.6**	**61.5**	**64.0**

evaluated by using different deep clustering algorithms. In this thesis, we have used DKM [136], DCN [29] and IDEC [31] and performed 10 individual seeded runs to measure the effectiveness of the model. Table 5.13 includes the results of different variants of PCD2C framework on 5 publicly available datasets using general hyperparameters while Table 5.15 includes the results of different PCD2C variations by tuning λ_1 hyperparameter (λ_0 has been set to 10^{-5}). As the results in Table 5.13 indicate, using pairwise constraints with general hyperparameters does not result in improvements over deep clustering approaches (DKM, DCN and IDEC) when no constraints is integrated into the framework. Moreover, as opposed to our expectations, adding more constraints (e.g., from 50 to 100) does not necessarily yield in better results. On the contrary, the improvements in the results by tuning λ_1 hyperparameter is more clear. Indeed, in terms of Macro-average results, the proposed framework outperforms DKM, DCN and IDEC where no pairwise constraints are used. Moreover, similar to the case where general hyperparameters are used, except on REUTERS, there are no clear trends in terms of adding more constraints to the proposed framework. Thus, we can conclude that if a few labeled data points are available, by tuning hyperparameter/s we can obtain state-of-the-art results. Figure 5.5 and Figure 5.6 illustrate the change in the clustering accuracy (ACC) by changing the λ_1 hyperparameter for 20NEWS and REUTERS datasets.

5.4.6.4 PCD2C User Constraints

We discussed the details of the user experiment that have been conducted and in this section we provide more details about the performance of the proposed

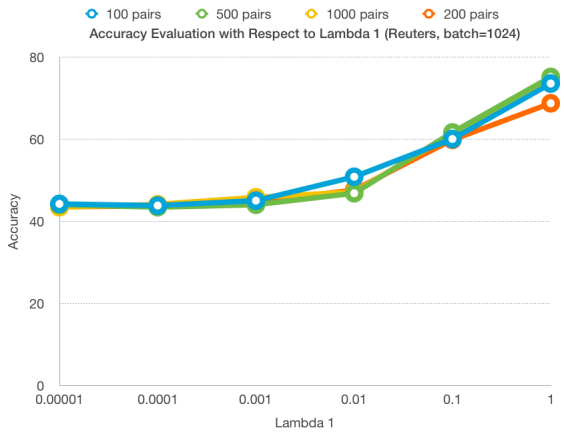


Figure 5.5: Evolution of PCD2C in the case of clustering accuracy (ACC) for REUTERS with respect to different values for λ_1 for different number of pairs–when pairwise constraints are used.

Figure 5.6: Evolution of PCD2C in the case of clustering accuracy (ACC) for 20NEWS with respect to different values for λ_1 for different number of pairs–when pairwise constraints are used.

PCD2C algorithm. Similar to using true labels for aligning must-link and cannot-link documents, we have used DKM [136], DCN [29] and IDEC [31] as our proposed candidates for evaluating the performance of the proposed PCD2C framework. In order to simulate a real case scenario when (possibly) there are no already-labeled documents, no hyperparameter tuning has been perform (similar to user experiments with SD2C approach) and both λ_0 and λ_1 has been set to 10^{-5} for all variants of PCD2C framework (DKM, DCN and IDEC). The results are presented in Table 5.17 and indicate that using DKM as the choice of deep clustering for the proposed PCD2C framework

leads towards outperforming DCN and IDEC in PCD2C framework. Indeed, only selecting DKM as the deep clustering algorithm in PCD2C leads towards better results compared to when no pairwise constraints are being used (original DKM). Here, similar to the case where true labels are used for obtaining pairwise constraints, no clear trends are observed in case of increasing the number of constraints.

Table 5.17: Results of different versions of PD2C algorithm (DKM, DCN and IDEC) on subset of DBPEDIA (including 25,000 documents) dataset with human selected seed words. λ_1 and λ_0 have been set to 10^{-5} (general hyperparameters).

Model	User 1			User 2			User 3		
	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI
DKM-e(50)	**86.0±1.1**	**71.4±1.5**	**72.7±0.7**	**86.3±1.0**	**71.8±1.3**	**72.9±0.9**	*84.3±6.0*	*71.6±3.6*	*72.8 ±0.7*
DKM-e(100)	**86.0±1.3**	**71.4±1.8**	**72.7±1.1**	*85.8±0.7*	*70.9±0.8*	*72.3±0.8*	*85.8±1.0*	*71.0±1.4*	*72.4±0.9*
DKM-e(300)	*85.9±1.2*	*71.1±1.7*	*72.4±1.2*	*85.9±1.2*	*71.2±1.6*	*72.5±1.1*	**86.2±1.1**	**71.7±1.6**	**72.9±0.9**
DKM-c(50)	80.3±9.0	*64.8±9.9*	*68.6±5.5*	*81.8±8.8*	*67.0±8.6*	*69.7±4.6*	*80.3±8.9*	64.7±10.2	*68.6±5.7*
DKM-c(100)	*82.6±8.4*	*67.8±8.9*	*70.4±4.9*	*83.6±6.5*	*68.3±6.6*	*70.5±3.8*	*83.4±6.8*	68.2±7.3	*70.4±4.3*
DKM-c(300)	*84.1±6.1*	*68.9±6.8*	*70.9±4.0*	*83.8±7.2*	*68.8±7.5*	*70.8±4.4*	*82.2±8.1*	*67.0±8.8*	*69.9±5.0*
DCN-e(50)	83.7±1.3	67.4±1.5	*71.0±1.3*	83.6±0.8	67.4±1.2	*70.8±0.9*	83.4±1.3	67.4±1.7	70.8±1.3
DCN-e(100)	83.7±1.0	67.6±1.7	*71.0±1.4*	83.5±0.8	67.4±1.3	*70.7±1.0*	83.5±1.0	67.4±1.6	70.8±1.3
DCN-e(300)	83.9±1.5	67.9±2.2	*71.0±1.4*	84.0±1.3	68.1±1.9	*71.2±1.2*	83.7±1.4	67.5±2.0	70.7±1.2
DCN-c(50)	83.7±1.1	67.6±1.8	*71.0±1.4*	83.4±1.3	67.2±2.1	*70.7±1.5*	83.9±1.3	67.9±1.8	71.1±1.0
DCN-c(100)	84.0±1.2	67.9±1.7	*71.0±1.2*	83.8±1.5	67.6±2.2	*70.8±1.3*	83.8±1.4	67.6±2.1	70.7±1.4
DCN-c(300)	83.4±1.0	67.2±1.7	*70.7±1.2*	83.6±0.9	67.5±1.4	*71.0±1.2*	83.7±1.0	67.7±1.7	71.1±1.3
IDEC-e(50)	83.7±1.5	67.7±2.3	*71.2±1.5*	83.7±1.5	67.6±2.4	*71.2±1.6*	83.7±1.5	67.6±2.4	71.2±1.5
IDEC-e(100)	83.7±1.5	67.6±2.3	*71.1±1.5*	83.7±1.5	67.6±2.4	*71.2±1.6*	83.7±1.5	67.6±2.4	71.2±1.6
IDEC-e(300)	83.7±1.5	67.7±2.4	*71.2±1.5*	83.7±1.5	67.6±2.4	*71.2±1.6*	83.7±1.5	67.6±2.4	71.2±1.6
IDEC-c(50)	75.8±8.3	56.8±9.1	64.0±5.6	74.2±10.3	55.2±10.5	63.5±6.7	74.9±10.0	55.9±11.2	63.8±6.9
IDEC-c(100)	74.5±9.9	55.7±10.3	63.6±6.3	74.4±9.9	55.6±11.0	63.5±7.1	78.0±8.4	60.3±9.0	66.1±5.6
IDEC-c(300)	77.9±8.1	59.8±8.1	65.8±5.2	75.3±10.2	57.2±10.2	64.4±6.3	74.2±10.1	55.9±10.6	63.9±6.5

Table 5.18: Results of different versions of PD2C algorithm (DKM, DCN and IDEC) on subset of DBPEDIA (including 25,000 documents) dataset with human selected seed words. λ_1 and λ_0 have been set to 10^{-5} (general hyperparameters).

Model	User 4			User 5			Macro-average		
	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI
DKM-e(50)	*86.3±1.3*	**71.7±1.8**	**72.8±1.0**	*86.0±1.0*	*71.2±1.4*	*72.5±0.9*	85.7	**71.5**	72.7
DKM-e(100)	**86.3±0.9**	*71.6±1.2*	*72.7±0.8*	**86.1±0.9**	**71.3±1.2**	*72.5±1.0*	**86.0**	71.2	72.5
DKM-e(300)	*85.8±0.9*	*70.9±1.2*	*72.2±0.9*	*85.9±1.0*	*71.2±1.3*	**72.6±0.9**	85.9	71.2	**72.8**
DKM-c(50)	*82.2±7.1*	*67.0±8.6*	*69.8±4.9*	*81.9±8.9*	*67.0±8.9*	*69.7±5.1*	81.3	66.1	69.2
DKM-c(100)	*82.4±8.3*	*67.3±9.0*	*70.1±5.0*	*81.5±8.7*	*66.8±8.2*	*69.8±4.6*	82.7	67.6	70.2
DKM-c(300)	*82.0±8.8*	*67.3±8.7*	*70.0±4.8*	*84.2±6.2*	*69.2±7.5*	71.0±4.4	83.2	68.2	70.5
DCN-e(50)	83.5±1.1	67.8±1.8	*70.8±1.6*	83.6±0.9	67.2±1.8	70.7±1.0	83.3	67.4	70.8
DCN-e(100)	83.8±1.4	67.8±2.1	*70.9±1.3*	83.7±1.5	67.6±2.1	70.8±1.2	83.6	67.5	70.8
DCN-e(300)	83.5±0.9	67.3±1.5	70.7±1.2	83.7±1.6	67.6±2.3	70.8±1.4	83.7	67.6	70.8
DCN-c(50)	83.6±0.8	67.5±1.3	70.8±1.1	84.0±1.5	68.1±2.0	*71.1±1.3*	83.7	67.6	70.9
DCN-c(100)	84.0±1.5	68.0±2.2	*70.9±1.5*	83.6±1.6	67.5±2.3	70.7±1.4	83.8	67.7	70.8
DCN-c(300)	83.3±1.1	67.0±1.8	70.6±1.3	83.8±1.6	67.7±2.3	70.7±1.4	83.5	67.4	70.8
IDEC-e(50)	83.7±1.5	67.6±2.3	*71.2±1.6*	83.7±1.5	67.6±2.3	*71.2±1.6*	83.7	67.6	71.2
IDEC-e(100)	83.7±1.5	67.6±2.4	*71.2±1.6*	83.7±1.5	67.6±2.3	*71.2±1.6*	83.7	67.6	71.2
IDEC-e(300)	83.7±1.5	67.6±2.4	*71.2±1.6*	83.7±1.5	67.6±2.3	*71.2±1.6*	83.7	67.6	71.2
IDEC-c(50)	76.1±9.3	57.0±10.1	64.4±6.2	73.8±10.7	56.0±11.4	63.8±6.5	74.9	56.0	64.1
IDEC-c(100)	77.4±7.5	59.3±7.3	65.6±6.5	73.8±10.6	55.1±11.7	63.0±7.5	75.6	57.1	64.3
IDEC-c(300)	76.3±9.3	58.2±9.9	65.1±6.1	73.4±9.9	54.3±10.9	62.8±6.6	75.4	57.8	64.4

5.5 Conclusion

In this chapter we proposed two frameworks named SD2C and PCD2C which are able to include constraints in end-to-end deep clustering methods. SD2C-Rep, SD2C-Doc and SD2C-Att are able to include seed words in deep

clustering methods while PCD2C is able to include pairwise constraints in deep clustering methods.

Moreover, we discussed that one can use any deep clustering methods such as DKM, DCN and IDEC in SD2C and PCD2C frameworks. To evaluate our proposed frameworks, we designed two different experiments: 1) automated constraints selection and 2) human constraints selection.

The results of both automated and human constraints selection indicate that using constraints in an end-to-end deep clustering framework helps to obtain better results compared to when constraints are not used. Indeed, SD2C and PCD2C are able to bias the data representations (learned by an Autoencoder) through utilizing constraints in their frameworks.

Moreover, the results also indicate that in most cases, using DKM as the deep clustering algorithm in SD2C and PCD2C can help to obtain better results compared to when DCN and IDEC are used.

CHAPTER 6

Conclusion

6.1 Summary

In this thesis we proposed the DKM algorithm where learning data representations and cluster representatives are performed in a truly joint way. The results indicate that our proposed DKM approach is able to outperform DCN and IDEC in the most cases. In fact, the results showed that optimizing Autoencoder weights and cluster representatives in a truly joint way yields better performance. In fact, DKM objective function allows the gradients to be computed smoothly which yields better optimization. On the contrary, similar methods such as DCN are not able to compute the gradients smoothly. To validate our approach, we tested our DKM framework, DCN and IDEC on variety types of data including text and images. The obtained results confirmed our intuition that optimizing Autoencoder weights and cluster representatives in a joint way leads toward better results.

Moreover, we tried to address the subjectivity aspect of clustering (more precisely, in deep clustering) by proposing two different approaches named as SD2C and PCD2C. In fact, we provided two deep clustering frameworks which can reflect user needs. Users can provide seed words (SD2C case) or pairwise constraints (PCD2C case) as additional information. All these provided additional information can be handled by these two proposed frameworks.

SD2C is able to include seed words constraints into any deep document clustering approaches. In fact, through biasing data representations (learned by an Autoencoder) towards the provided seed words, we can obtain suitable results which can satisfy user needs. The results of this framework indicate

that including such constraints are useful for obtaining more tailored results. In fact, if the provided seed words per cluster are able to define the clusters well (e.g., football for sport cluster) then the representations will be biased and the algorithm is able to obtain better results. Otherwise, if the quality of seed words is not good, we may not obtain better results.

Moreover, we devised two different ways of obtaining seed words which are names as automatically selected seed words and manually selected ones. For the case of manually selecting seed words, we performed several experiments with multiple participants who were asked to withdraw seed words through reading variety of sentences. In these experiments, users did not receive any external help from computers and tools to select the seed words. The results of both manually and automatically selected seed words approved our unique way of extracting automatic seed words. Moreover, the results of manually selected seed words indicate that our proposed framework is able to handle real world case scenarios.

We also discussed that another type of constraints are pairwise constraints where if a pair of documents are similar (belong to the same cluster) then they are labeled as ML and otherwise, if they belong to the different clusters, they are labeled as CL. PCD2C approach has been designed to be able to include pairwise constraints provided by the users into any deep document clustering approaches. In this case, by including ML and CL constraints into the PDC2C framework, we tried to bias the data representations learned by an Autoencoder (similar to SD2C).

Similar to the SD2C, we also performed several experiments where different participants where asked to assign ML and CL labels to multiple pairs of documents. Then, the manually selected pairwise constraints were fed into the PCD2C framework to measure the performance of the algorithm.

The results of the both manually selected pairwise constraints and automatically selected ones showed that PCD2C is able to integrate pairwise constraints in an useful way where we can obtain better tailored results. Compared to the SD2C, PCD2C is less affected by the wrong choice of the user. In another words, the variance of the results of PCD2C are less compared to SD2C.

Another strong aspect of all the proposed frameworks including DKM, SD2C and PCD2C is that for the Autoencoder part, no hyperparameters tuning has been performed. Certainly, through hyperparameters tuning, we can obtain better results.

6.2 Future work

In this section, we discuss about the future works that might be considered for this thesis. First, we discuss future works about DKM and in general deep clustering approaches without the seed words and then discuss the possible future works of deep clustering approaches which try to integrate the seed words.

6.2.1 Deep Clustering Without Seed Words

In the case of deep clustering approaches (without constraints), other clustering algorithms rather than $k-$means can be used and tested. Indeed, In this thesis, we focused on designing an alternative objective function for $k-$means which allows gradients to be computed. There are tremendous amounts of other clustering algorithms which can be re-designed to be used in an end-to-end deep clustering framework.

Moreover, the capability of memory-based neural network algorithms can be investigated. In this case, different representations per cluster can be defined and these representations can place in a virtual memory (matrix). Similar to the [138] reading from the memory and writing to the memory can be defined.

Also, different types of Autoenocoders can be explored as well. Due to the infrastructure limitations, we were not able to explore Seq2Seq Autoencoders (similar to Machine Translation) as these networks tend to process the input sequence one-by-one (in case of using RNN based neural networks in the Autoencoder architecture). Since such Autoencoders seem to be more beneficial when the input data is text, trying and testing such models are recommended.

Moreover, we used Word2Vec method for word representations. As mentioned in the previous chapters, we used fastText as well but the results of Word2Vec were quite better copmpared to the fastText. In the future, more modern techniques such as Bert [139] can be used. Since in this method, the representation of the document can be obtained (along with the representation of each word in the document), it is highly possible that utilizing this method yields better results.

6.2.2 Deep Clustering Including Seed Words

In the case of using seed words in the deep clustering architecture, we can try the same possible future works mentioned for the case of deep clustering without seed words.

Moreover, in this case, a combination of SD2C-Doc and SD2C-Rep will be investigated. Their losses can be summed or a new formulation based on their losses can be defined. Moreover, using attention mechanism in such frameworks can be explored. In this thesis, we tried a single version of attention based model but this model suffers from complexity and not producing good results. Reformulating the current model and solving complexity is a direction that we can explore as well.

Bibliography

[1] J. Macqueen. Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967. (*Cited on pages 2, 6, 42, and 69.*)

[2] Lior Rokach and Oded Maimon. The data mining and knowledge discovery handbook: A complete guide for researchers and practitioners, 2005. (*Cited on page 5.*)

[3] Anil K. Jain. Data clustering: 50 years beyond k-means, 2008. (*Cited on page 5.*)

[4] Amandeep Kaur Mann & Navneet Kaur. Paper on clustering techniques by amandeep kaur mann & navneet. 2013. (*Cited on page 6.*)

[5] K. A Nazeer and M Sebastian. Improving the accuracy and efficiency of the k-means clustering algorithm. *Lecture Notes in Engineering and Computer Science*, 2176, 07 2009. (*Cited on page 6.*)

[6] David Arthur and Sergei Vassilvitskii. K-means++: the advantages of careful seeding. In *In Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007. (*Cited on pages 6 and 42.*)

[7] Rui Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, May 2005. (*Cited on page 6.*)

[8] Shi Li and Xiangyu Guo. Distributed k-clustering for data with heavy noise. In *Advances in Neural Information Processing Systems*, pages 7838–7846, 2018. (*Cited on page 6.*)

[9] S. Na, L. Xumin, and G. Yong. Research on k-means clustering algorithm: An improved k-means clustering algorithm. In *2010 Third*

International Symposium on Intelligent Information Technology and Security Informatics, pages 63–67, April 2010. (*Cited on page 7.*)

[10] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984. (*Cited on pages 7 and 36.*)

[11] R. R. Yager and D. P. Filev. Approximate clustering via the mountain method. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(8):1279–1284, Aug 1994. (*Cited on page 7.*)

[12] I. Gath and A. B. Gev. Unsupervised optimal fuzzy clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(7):773–780, July 1989. (*Cited on page 7.*)

[13] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1180–1189. JMLR.org, 2015. (*Cited on page 8.*)

[14] Peicheng Zhou, Junwei Han, Gong Cheng, and Baochang Zhang. Learning compact and discriminative stacked autoencoder for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 2019. (*Cited on page 8.*)

[15] S Mostafa Mousavi, Weiqiang Zhu, William Ellsworth, and Gregory Beroza. Unsupervised clustering of seismic signals using deep convolutional autoencoders. *IEEE Geoscience and Remote Sensing Letters*, 2019. (*Cited on page 8.*)

[16] FJ Martinez-Murcia, A Ortiz, JM Gorriz, J Ramirez, and D Castillo-Barnes. Studying the manifold structure of alzheimer's disease: A deep learning approach using convolutional autoencoders. *IEEE journal of biomedical and health informatics*, 2019. (*Cited on page 8.*)

[17] Lei Cai, Hongyang Gao, and Shuiwang Ji. Multi-stage variational auto-encoders for coarse-to-fine image generation. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 630–638. SIAM, 2019. (*Cited on page 8.*)

[18] Shigeki Karita, Shinji Watanabe, Tomoharu Iwata, Marc Delcroix, Atsunori Ogawa, and Tomohiro Nakatani. Semi-supervised end-to-end speech recognition using text-to-speech and autoencoders. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and*

Signal Processing (ICASSP), pages 6166–6170. IEEE, 2019. (*Cited on page 8.*)

[19] Yu-An Chung, Wei-Hung Weng, Schrasing Tong, and James Glass. Towards unsupervised speech-to-text translation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7170–7174. IEEE, 2019. (*Cited on page 8.*)

[20] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. (*Cited on page 9.*)

[21] Guillaume Alain and Yoshua Bengio. What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593, 2014. (*Cited on page 9.*)

[22] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pages 899–907, 2013. (*Cited on page 9.*)

[23] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990. (*Cited on page 9.*)

[24] Deng Cai, Xiaofei He, and Jiawei Han. Locally consistent concept factorization for document clustering. *IEEE Transactions on Knowledge and Data Engineering*, 23(6):902–913, 2010. (*Cited on page 10.*)

[25] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. (*Cited on page 10.*)

[26] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854, 2010. (*Cited on page 11.*)

[27] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018. (*Cited on page 13.*)

[28] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010. (*Cited on page 14.*)

[29] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3861–3870. JMLR. org, 2017. (*Cited on pages 14, 32, 37, 42, 67, 69, 75, 76, 82, and 83.*)

[30] Peihao Huang, Yan Huang, Wei Wang, and Liang Wang. Deep embedding network for clustering. In *2014 22nd International Conference on Pattern Recognition*, pages 1532–1537. IEEE, 2014. (*Cited on pages 14 and 37.*)

[31] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *IJCAI*, pages 1753–1759, 2017. (*Cited on pages 14, 37, 39, 42, 43, 67, 69, 75, 76, 82, and 83.*)

[32] Dongdong Chen, Jiancheng Lv, and Yi Zhang. Unsupervised multi-manifold clustering by learning deep representation. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017. (*Cited on page 15.*)

[33] Kai Tian, Shuigeng Zhou, and Jihong Guan. Deepcluster: A general clustering framework based on deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 809–825. Springer, 2017. (*Cited on page 15.*)

[34] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006. (*Cited on page 15.*)

[35] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011. (*Cited on page 15.*)

[36] Jinghua Wang and Jianmin Jiang. An unsupervised deep learning framework via integrated optimization of representation learning and gmm-based modeling. In *Asian Conference on Computer Vision*, pages 249–265. Springer, 2018. (*Cited on page 15.*)

[37] Ming Shao, Sheng Li, Zhengming Ding, and Yun Fu. Deep linear coding for fast graph clustering. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015. (*Cited on page 16.*)

[38] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning deep representations for graph clustering. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014. (*Cited on page 16.*)

[39] Sharon Fogel, Hadar Averbuch-Elor, Daniel Cohen-Or, and Jacob Goldberger. Clustering-driven deep embedding with pairwise constraints. *IEEE computer graphics and applications*, 39(4):16–27, 2019. (*Cited on page 16.*)

[40] Elad Tzoreff, Olga Kogan, and Yoni Choukroun. Deep discriminative latent space for clustering. *arXiv preprint arXiv:1805.10795*, 2018. (*Cited on page 16.*)

[41] Mengqi Wu, Guannan Liu, Junjie Wu, Peng Li, and Xin Wan. Ensemble clustering via learning representations from auto-encoder. In *2018 15th International Conference on Service Systems and Service Management (ICSSSM)*, pages 1–6. IEEE, 2018. (*Cited on page 16.*)

[42] Shlomo E Chazan, Sharon Gannot, and Jacob Goldberger. Deep clustering based on a mixture of autoencoders. *arXiv preprint arXiv:1812.06535*, 2018. (*Cited on page 16.*)

[43] P. Dahal. Learning embedding space for clustering from deep representations. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3747–3755, Dec 2018. (*Cited on page 16.*)

[44] D. Das, R. Ghosh, and B. Bhowmick. Deep representation learning characterized by inter-class separation for image clustering. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 628–637, Jan 2019. (*Cited on page 16.*)

[45] Xu Yang, Cheng Deng, Feng Zheng, Junchi Yan, and Wei Liu. Deep spectral clustering using dual autoencoder network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4066–4075, 2019. (*Cited on page 16.*)

[46] Pengfei Ge, Chuan-Xian Ren, Dao-Qing Dai, Jiashi Feng, and Shuicheng Yan. Dual adversarial autoencoders for clustering. *IEEE transactions on neural networks and learning systems*, 2019. (*Cited on page 17.*)

[47] Nairouz Mrabah, Naimul Khan, and Riadh Ksantini. Deep clustering with a dynamic autoencoder, 01 2019. (*Cited on page 17.*)

[48] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. In *Proceedings of the twenty-first international conference on Machine learning*, page 29, 2004. (*Cited on page 17.*)

[49] W. Wang, D. Yang, F. Chen, Y. Pang, S. Huang, and Y. Ge. Clustering with orthogonal autoencoder. *IEEE Access*, 7:62421–62432, 2019. (*Cited on page 17.*)

[50] Milad Leyli Abadi, Lazhar Labiod, and Mohamed Nadif. Denoising autoencoder as an effective dimensionality reduction and clustering of text data. pages 801–813, 04 2017. (*Cited on page 17.*)

[51] Shuyang Wang, Zhengming Ding, and Yun Fu. Feature selection guided auto-encoder. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 2725–2731. AAAI Press, 2017. (*Cited on page 17.*)

[52] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*, 2016. (*Cited on page 17.*)

[53] Warith Harchaoui, Pierre-Alexandre Mattei, and Charles Bouveyron. Deep adversarial gaussian mixture auto-encoder for clustering. 2017. (*Cited on page 17.*)

[54] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015. (*Cited on page 17.*)

[55] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015. (*Cited on page 17.*)

[56] W. Zhou and Q. Zhou. Deep embedded clustering with adversarial distribution adaptation. *IEEE Access*, 7:113801–113809, 2019. (*Cited on page 17.*)

[57] Xi Peng, Shijie Xiao, Jiashi Feng, Wei-Yun Yau, and Zhang Yi. Deep subspace clustering with sparsity prior. In *IJCAI*, pages 1925–1931, 2016. (*Cited on page 19.*)

[58] Xi Peng, Jiashi Feng, Jiwen Lu, Wei-Yun Yau, and Zhang Yi. Cascade subspace clustering. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. (*Cited on page 19.*)

[59] Ehsan Elhamifar and Rene Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–2781, 2013. (*Cited on page 19.*)

[60] Shankar R. Rao, Roberto Tron, René Vidal, and Yi Ma. Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(10):1832–1845, 2010. (*Cited on page 19.*)

[61] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian Reid. Deep subspace clustering networks. In *Advances in Neural Information Processing Systems*, pages 24–33, 2017. (*Cited on page 19.*)

[62] Changlu Chen, Chaoxi Niu, Xia Zhan, and Kun Zhan. A generative approach to unsupervised deep local learning. *arXiv preprint arXiv:1906.07947*, 2019. (*Cited on page 19.*)

[63] Sohil Atul Shah and Vladlen Koltun. Deep continuous clustering. *arXiv preprint arXiv:1803.01449*, 2018. (*Cited on page 19.*)

[64] Sohil Atul Shah and Vladlen Koltun. Robust continuous clustering. *Proceedings of the National Academy of Sciences*, 114(37):9814–9819, 2017. (*Cited on page 19.*)

[65] Ershad Banijamali and Ali Ghodsi. Fast spectral clustering using autoencoders and landmarks. In *International Conference Image Analysis and Recognition*, pages 380–388. Springer, 2017. (*Cited on page 19.*)

[66] Séverine Affeldt, Lazhar Labiod, and Mohamed Nadif. Spectral clustering via ensemble deep autoencoder learning (sc-edae). *arXiv preprint arXiv:1901.02291*, 2019. (*Cited on page 19.*)

[67] Uri Shaham, Kelly Stanton, Henry Li, Boaz Nadler, Ronen Basri, and Yuval Kluger. Spectralnet: Spectral clustering using deep neural networks. *arXiv preprint arXiv:1801.01587*, 2018. (*Cited on page 19.*)

[68] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5736–5745, 2017. (*Cited on page 20.*)

[69] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. (*Cited on page 20.*)

[70] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. (*Cited on page 20.*)

[71] Kamran Ghasedi Dizaji, Feng Zheng, Najmeh Sadoughi, Yanhua Yang, Cheng Deng, and Heng Huang. Unsupervised deep generative adversarial hashing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3664–3673, 2018. (*Cited on page 20.*)

[72] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3270–3278, 2015. (*Cited on page 20.*)

[73] Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. *arXiv preprint arXiv:1811.11155*, 2018. (*Cited on page 20.*)

[74] Thomas Brox, Andrés Bruhn, and Mario Fritz, editors. *Pattern Recognition - 40th German Conference, GCPR 2018, Stuttgart, Germany, October 9-12, 2018, Proceedings*, volume 11269 of *Lecture Notes in Computer Science*. Springer, 2019. (*Cited on page 21.*)

[75] X. Shaol, K. Ge, H. Su, L. Luo, B. Peng, and D. Li. Deep discriminative clustering network. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, July 2018. (*Cited on page 21.*)

[76] Chih-Chung Hsu and Chia-Wen Lin. Cnn-based joint clustering and representation learning with feature drift compensation for large-scale image data. *IEEE Transactions on Multimedia*, 20(2):421–429, 2017. (*Cited on page 21.*)

[77] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016. (*Cited on page 21.*)

[78] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5879–5887, 2017. (*Cited on page 21.*)

[79] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1558–1567. JMLR. org, 2017. (*Cited on page 21.*)

[80] Andreas Krause, Pietro Perona, and Ryan G Gomes. Discriminative clustering by regularized information maximization. In *Advances in neural information processing systems*, pages 775–783, 2010. (*Cited on page 21.*)

[81] Gang Chen. Deep learning with nonparametric clustering. *arXiv preprint arXiv:1501.03084*, 2015. (*Cited on page 21.*)

[82] Geoffrey E. Hinton. Deep belief networks. *Scholarpedia*, 4:5947, 01 2009. (*Cited on page 21.*)

[83] Hamed Valizadegan and Rong Jin. Generalized maximum margin clustering and unsupervised kernel learning. In *Advances in neural information processing systems*, pages 1417–1424, 2007. (*Cited on page 21.*)

[84] Wei-An Lin, Jun-Cheng Chen, Carlos D Castillo, and Rama Chellappa. Deep density clustering of unconstrained faces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8128–8137, 2018. (*Cited on page 22.*)

[85] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016. (*Cited on pages 22, 37, 39, 42, 43, 75, and 76.*)

[86] Xifeng Guo, En Zhu, Xinwang Liu, and Jianping Yin. Deep embedded clustering with data augmentation. In *Asian Conference on Machine Learning*, pages 550–565, 2018. (*Cited on page 22.*)

[87] Fengfu Li, Hong Qiao, and Bo Zhang. Discriminatively boosted image clustering with fully convolutional auto-encoders. *Pattern Recognition*, 83:161–173, 2018. (*Cited on page 22.*)

[88] Yen-Chang Hsu and Zsolt Kira. Neural network-based clustering using pairwise constraints. *arXiv preprint arXiv:1511.06321*, 2015. (*Cited on page 22.*)

[89] Thi-Bich-Hanh Dao, Chia-Tung Kuo, SS Ravi, Christel Vrain, and Ian Davidson. Descriptive clustering: Ilp and cp formulations with applications. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 1263–1269, 2018. (*Cited on page 27.*)

[90] Bing Liu, Xiaoli Li, Wee Sun Lee, and Philip S. Yu. Text Classification by Labeling Words. In *Proceedings of AAAI/IAAI*, pages 425–430, 2004. (*Cited on page 27.*)

[91] Youngjoong Ko and Jungyun Seo. Learning with Unlabeled Data for Text Categorization Using Bootstrapping and Feature Projection Techniques. In *Proceedings of ACL*, pages 255–262, 2004. (*Cited on page 27.*)

[92] Ming-Wei Chang, Lev Ratinov, Dan Roth, and Vivek Srikumar. Importance of Semantic Representation: Dataless Classification. In *Proceedings of AAAI*, pages 830–835, 2008. (*Cited on pages 27 and 69.*)

[93] Gregory Druck, Gideon Mann, and Andrew Mccallum. Learning from Labeled Features using Generalized Expectation Criteria. In *Proceedings of SIGIR*, pages 595–602, 2008. (*Cited on page 27.*)

[94] Alfio Gliozzo, Carlo Strapparava, and Ido Dagan. Improving Text Categorization Bootstrapping via Unsupervised Learning. *ACM Transactions on Speech and Language Processing*, 6(1):1–24, 2009. (*Cited on page 27.*)

[95] Xingyuan Chen, Yunqing Xia, Peng Jin, and John Carroll. Dataless Text Classification with Descriptive LDA. In *Proceedings of AAAI*, pages 2224–2231, 2015. (*Cited on pages 27, 60, 68, and 70.*)

[96] Chenliang Li, Jian Xing, Aixin Sun, and Zongyang Ma. Effective Document Labeling with Very Few Seed Words: A Topic Model Approach. In *Proceedings of CIKM*, pages 85–94, 2016. (*Cited on pages 27, 28, 60, 68, 69, and 70.*)

[97] Ximing Li, Changchun Li, Jinjin Chi, Jihong Ouyang, and Chenliang Li. Dataless Text Classification: A Topic Modeling Approach with Document Manifold. In *Proceedings of CIKM*, pages 973–982, 2018. (*Cited on pages 27, 28, 60, 68, 69, and 70.*)

[98] Chenliang Li, Shiqian Chen, Jian Xing, Aixin Sun, and Zongyang Ma. Seed-Guided Topic Model for Document Filtering and Classification. *ACM Transactions on Information Systems*, 37(1), 2018. (*Cited on pages 27, 28, 60, 68, and 70.*)

[99] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003. (*Cited on pages 27 and 61.*)

[100] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. (*Cited on pages 28 and 39.*)

[101] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of EMNLP*, pages 1532–1543, 2014. (*Cited on page 28.*)

[102] Sugato Basu, Ian Davidson, and Kiri Wagstaff. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman & Hall/CRC, first edition, 2008. (*Cited on pages 28 and 60.*)

[103] H. Zhang, S. Basu, and I. Davidson. A framework for deep constrained clustering - algorithms and advances. In *ECML/PKDD*, 2019. (*Cited on page 28.*)

[104] Noam Shental, Aharon Bar-Hillel, Tomer Hertz, and Daphna Weinshall. Computing Gaussian Mixture Models with EM using Equivalence Constraints. In *Proceedings of NeurIPS*, pages 465–472, 2003. (*Cited on pages 28 and 60.*)

[105] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584, 2001. (*Cited on pages 28, 29, and 66.*)

[106] Yi Liu, Rong Jin, and Anil Jain. Boostcluster: Boosting clustering by pairwise constraints. pages 450–459, 01 2007. (*Cited on pages 28 and 66.*)

[107] Yang Hu, Jingdong Wang, Nenghai Yu, and Xian-Sheng Hua. Maximum margin clustering with pairwise constraints. In *2008 Eighth IEEE International Conference on Data Mining*, pages 253–262. IEEE, 2008. (*Cited on pages 28 and 29.*)

[108] Hong Zeng and Yiu-ming Cheung. Semi-supervised maximum margin clustering with pairwise constraints. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):926–939, 2011. (*Cited on pages 28 and 29.*)

[109] Sugato Basu, Arindam Banerjee, and Raymond J Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 2004 SIAM international conference on data mining*, pages 333–344. SIAM, 2004. (*Cited on pages 28 and 29.*)

[110] Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In *Advances in neural information processing systems*, pages 1537–1544, 2005. (*Cited on page 29.*)

[111] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, July 1998. (*Cited on page 29.*)

[112] Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017. (*Cited on page 35.*)

[113] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017. (*Cited on page 35.*)

[114] Kenneth Rose, Eitan Gurewitz, and Geoffrey Fox. A Deterministic Annealing Approach to Clustering. *Pattern Recognition Letters*, 11(9):589–594, 1990. (*Cited on page 36.*)

[115] Juan Ramos. Using tf-idf to determine word relevance in document queries. (*Cited on page 39.*)

[116] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification, 2016. (*Cited on page 39.*)

[117] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014. (*Cited on page 39.*)

[118] Xingyuan Chen, Yunqing Xia, Peng Jin, and John Carroll. Dataless text classification with descriptive lda. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. (*Cited on page 40.*)

[119] Chenliang Li, Shiqian Chen, Jian Xing, Aixin Sun, and Zongyang Ma. Seed-guided topic model for document filtering and classification. *ACM Trans. Inf. Syst.*, 37(1):9:1–9:37, December 2018. (*Cited on page 40.*)

[120] Chenliang Li, Jian Xing, Aixin Sun, and Zongyang ma. Effective document labeling with very few seed words: A topic model approach. pages 85–94, 10 2016. (*Cited on page 40.*)

[121] Ximing Li, Changchun Li, Jinjin Chi, Jihong Ouyang, and Chenliang Li. Dataless text classification: A topic modeling approach with document manifold. pages 973–982, 10 2018. (*Cited on page 40.*)

[122] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015. (*Cited on pages 41 and 68.*)

[123] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy Layer-Wise Training of Deep Networks. In *Proceedings of the 20th Annual Conference on Neural Information Processing Systems*, NIPS '06, pages 153–160, 2006. (*Cited on page 42.*)

[124] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. (*Cited on page 43.*)

[125] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010. (*Cited on pages 43 and 76.*)

[126] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. (*Cited on pages 43 and 76.*)

[127] Deng Cai, Xiaofei He, and Jiawei Han. Locally Consistent Concept Factorization for Document Clustering. *IEEE Transactions on Knowledge and Data Engineering*, 23(6):902–913, 2011. (*Cited on pages 46 and 76.*)

[128] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. (*Cited on page 56.*)

[129] MM Fard, T Thonet, and E Gaussier. Seed-guided deep document clustering. *Advances in Information Retrieval*, 12035:3–16, 2020. (*Cited on page 59.*)

[130] Maziar Moradi Fard, Thibaut Thonet, and Eric Gaussier. Pairwise-constrained deep document clustering. In Igor Kabashkin, Irina Yatskiv, and Olegas Prentkovskis, editors, *Reliability and Statistics in Transportation and Communication*, pages 12–21, Cham, 2020. Springer International Publishing. (*Cited on pages 59 and 66.*)

[131] Donglin Niu, Jennifer G Dy, and Michael I Jordan. Multiple Non-Redundant Spectral Clustering Views. In *Proceedings of ICML*, pages 831–838, 2010. (*Cited on page 60.*)

[132] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained K-means Clustering with Background Knowledge. In *Proceedings of ICML*, pages 577–584, 2001. (*Cited on page 60.*)

[133] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, Maximilian Strobel, and Daniel Cremers. Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648*, 2018. (*Cited on page 62.*)

[134] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. (*Cited on page 64.*)

[135] Noam Shental, Aharon Bar-Hillel, Tomer Hertz, and Daphna Weinshall. Computing gaussian mixture models with em using equivalence constraints. In *Advances in neural information processing systems*, pages 465–472, 2004. (*Cited on page 66.*)

[136] Maziar Moradi Fard, Thibaut Thonet, and Eric Gaussier. Deep k-Means: Jointly Clustering with k-Means and Learning Representations. *arXiv:1806.10069*, 2018. (*Cited on pages 67, 69, 75, 76, 82, and 83.*)

[137] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003. (*Cited on page 73.*)

[138] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016. (*Cited on page 89.*)

[139] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. (*Cited on page 89.*)