Seed-Guided Deep Document Clustering

Mazar Moradi Fard[1](✉), Thibaut Thonet[2], and Eric Gaussier[1]

[1] Univ. Grenoble Alpes, CNRS - LIG, Grenoble, France
{maziar.moradi-fard,eric.gaussier}@univ-grenoble-alpes.fr
[2] NAVER LABS Europe, Meylan, France
thibaut.thonet@naverlabs.com

Abstract. 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. In this paper, we jointly learn deep representations and bias the clustering results through the seed words, leading to a Seed-guided Deep Document Clustering approach. Its effectiveness is demonstrated on five public datasets.

Keywords: Document clustering · Representation learning · Dataless text classification

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 [25]. For 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* [2]. Enabling users to provide clustering constraints in the context of an exploratory task can help obtain results better tailored to their needs. Typically, must-link and cannot-link constraints are considered (e.g., see [27,29]), 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,

J. M. Jose et al. (Eds.): ECIR 2020, LNCS 12035, pp. 3–16, 2020.
https://doi.org/10.1007/978-3-030-45439-5_1

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 [8,16,17,19], inspired by the Latent Dirichlet Allocation model [3]. Concurrently, important advances on clustering were recently enabled through its combination with deep representation learning (e.g., see [12,23,30,31]), which is now known as *deep clustering*. A common approach to deep clustering is to jointly train an autoencoder and perform clustering on the learned representations [23,30,31]. 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 study can be summarized as follows: (a) We introduce the **S**eed-guided **D**eep **D**ocument **C**lustering (SD2C) framework,[1] the first attempt, to the best of our knowledge, to constrain clustering with seed words based on a deep clustering approach; and (b) we validate this framework through experiments based on automatically selected seed words on five publicly available text datasets with various sizes and characteristics.

The remainder of the paper is organized as follows. In Sect. 2, we describe existing works on seed-guided constrained document clustering, also known as dataless text classification. Section 3 then introduces the seed-guided deep document clustering framework, which is then evaluated in Sect. 4. Section 5 concludes the paper and provides some perspectives on SD2C.

2 Related Work

One way to address the above problem is to try and identify multiple clustering views from the data in a purely unsupervised fashion [7,24,25]. While such an approach provides users with several possible clustering results to choose from, there is still no guarantee that the obtained clusters are those the users are interested in.

The constrained clustering problem we are addressing 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. For a more general survey on constrained clustering, we invite the reader to refer to [2]. The task of dataless text classification was introduced independently by Liu *et al.* [20] and Ko *et al.* [14]. In [20], the seed words are provided by a user and exploited to automatically label a part of the unlabeled documents. On the other hand, in [14], 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

[1] The code is available at https://github.com/MaziarMF/SD2C.

further investigated the exploitation of seed words for text classification [6,9,10]. Chang *et al.* [6] introduced both an ‘on-the-fly’ approach and a bootstrapping approach by projecting seed words and documents in the same space. 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 [10], 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 [9] 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 [8,16,17,19], extending the Latent Dirichlet Allocation model [3]. The topic model devised by Chen *et al.* [8] 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 [17], 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 [16] 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 by Li *et al.* [19] 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 [22] and GloVe [26]).

To the best of our knowledge, only one deep learning-based approach was proposed to address a problem similar to dataless text classification [18]. In this recent work, Li *et al.* devised a deep relevance model for zero-shot document filtering – which consists at test time in predicting the relevance of documents with respect to a category unseen in the training set, where each category is characterized by a set of seed words. This problem is nonetheless different from dataless text classification as it focuses on estimating documents' relevance (or lack thereof) instead of class membership.

3 Seed-Guided Deep Document Clustering

Deep clustering consists in jointly performing clustering and deep representation learning in an unsupervised fashion (e.g., with an auto-encoder). All deep clustering approaches aim at obtaining representations that are both faithful to the original documents and are more suited to document clustering purposes than the original document representation. To do so, they 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.

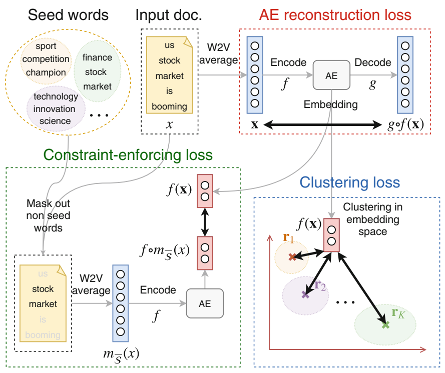
In the remainder, $\mathcal{X}$ will denote the set of documents to cluster. Each document $x \in \mathcal{X}$ is associated with a representation $\mathbf{x}$ in $\mathbb{R}^d$ – thereafter, the *input space* – defined as the average of the (precomputed) embeddings of the words in x, where d is the dimension of the word embedding space. Each word w is thus represented as a d-dimensional vector $\mathbf{w}$ corresponding to its embedding (Sect. 4 further discusses the different word embeddings considered). Let $f_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^p$ and $g_\eta : \mathbb{R}^p \rightarrow \mathbb{R}^d$ be an encoder and a decoder with parameters θ and η, respectively; $g_\eta \circ f_\theta$ then defines an auto-encoder (AE). $\mathbb{R}^p$ 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 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:

$$\underset{\theta,\eta,\mathcal{R}}{\operatorname{argmin}} \underbrace{\mathcal{L}_{\text{rec}}(\mathcal{X}, g_\eta \circ f_\theta(\mathcal{X})) + \lambda_0 \mathcal{L}_{\text{clust}}(f_\theta(\mathcal{X}), \mathcal{R})}_{\mathcal{L}_{\text{dc}}(\mathcal{X},\theta,\eta,\mathcal{R})}. \tag{1}$$

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 [1].

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.

SD2C-Doc. One way to bias the document representations according to the seed words is to reduce the gap in the embedding space between the representation of the documents and the representation of the seed words occurring in these documents. For that purpose, 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


Seed words
Input doc.
AE reconstruction loss
sport competition champion
finance stock market
technology innovation science
us stock market is booming
W2V average
Encode
Decode
AE
Embedding
Constraint-enforcing loss
Clustering loss
Mask out non seed words
Clustering in embedding space

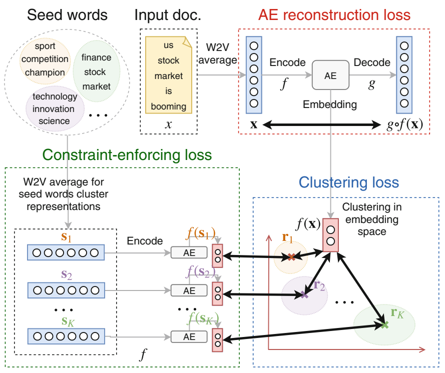

Seed words
Input doc.
AE reconstruction loss
W2V average
Encode
Decode
AE
Embedding
Constraint-enforcing loss
W2V average for seed words cluster representations
Clustering loss
Clustering in embedding space

Fig. 1. Illustration of SD2C-Doc (left) and SD2C-Rep (right). Thick double arrows indicate the computation of a distance between two vectors.

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:

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

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. 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. 2 yielded the same results in our experiments. Because of its simplicity, we rely on the first one in the remainder of the paper, which is illustrated in Fig. 1 (left). 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:

$$\operatorname*{argmin}_{\theta,\eta,\mathcal{R}} \mathcal{L}_{\mathrm{dc}}(\mathcal{X},\theta,\eta,\mathcal{R}) + \lambda_1 \sum_{x\in\mathcal{X}} \delta^E(f_\theta(\mathbf{x}), f_\theta\circ m_{\overline{\mathcal{S}}}(x)), \tag{3}$$

where λ_1 is an hyperparameter controlling the importance of the deep clustering loss $\mathcal{L}_{dc}$ and the loss associated to seed words.

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. 1 (right), then takes the form:

$$\underset{\theta,\eta,\mathcal{R}}{\operatorname{argmin}} \mathcal{L}_{\text{dc}}(\mathcal{X}, \theta, \eta, \mathcal{R}) + \lambda_1 \sum_{k=1}^{K} \delta^E(\mathbf{r}_k, f_\theta(\mathbf{s}_k)), \tag{4}$$

As before, δ^E denotes a dissimilarity in the embedding space. The last term in Eq. 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 (see Sect. 3.1 below).

3.1 Training

In practice, we use fully differentiable formulations of Problems 3 and 4. 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 [31] (called DCN) and [23] (called DKM), the latter proposing a truly joint formulation of the deep clustering problem. Other state-of-the-art deep clustering approaches, as IDEC [12], also based on cluster representatives, could naturally be adopted as well. The comparison between these approaches performed in [23] 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 [23] with:

$$\mathcal{L}_{\text{rec}}(\mathcal{X}, g_\eta \circ f_\theta(\mathcal{X})) = \sum_{x \in \mathcal{X}} \delta^I(\mathbf{x}, g_\eta \circ f_\theta(\mathbf{x})), \tag{5}$$

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

$$\mathcal{L}_{\text{clust}}(f_\theta(\mathcal{X}), \mathcal{R}) = \sum_{x \in \mathcal{X}} \sum_{k=1}^{K} \delta^E(f_\theta(\mathbf{x}), \mathbf{r}_k)\, G_k(f_\theta(\mathbf{x}), \alpha; \mathcal{R}) \tag{6}$$

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:

$$G_k(f_\theta(\mathbf{x}), \alpha; \mathcal{R}) = \frac{\exp\big(-\alpha \cdot \delta^E(f_\theta(\mathbf{x}), \mathbf{r}_k)\big)}{\sum_{k'=1}^{K} \exp\big(-\alpha \cdot \delta^E(f_\theta(\mathbf{x}), \mathbf{r}_{k'})\big)}. \tag{7}$$

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

Following prior deep clustering works [12,23,30,31], we initialize the auto-encoder parameters through pretraining by first only optimizing the reconstruction loss of the auto-encoder. In the pretraining of SD2C-Doc, we also include the constraint-enforcing term (second term in Problem 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 $\{\mathbf{s}_k\}_{k=1}^K$.[2] 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.

4 Experiments

The experiments we performed to evaluate the proposed SD2C framework 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[3] dataset, referred to as *20NEWS*; the Reuters-21578[4] dataset, referred to as *REUTERS*, from which, similarly to [8,16,17,19], we use only the 10 largest (and highly imbalanced) categories; the Yahoo! Answers dataset [32], referred to as *YAHOO*, from which we use only the test set comprising 60,000 documents evenly split into 10 classes; the DBPedia dataset [32], 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 [32] and referred to as *AGNEWS*, from which we use the training set, composed of 120,000 documents evenly split into 4 classes. After preprocessing, which includes removing stop words and words made of less than 2 characters, Porter stemming and discarding the empty documents, the number of documents in 20NEWS, REUTERS, YAHOO, DBPEDIA, AGNEWS are respectively 18,846, 7,964, 59,978, 70,000, 120,000. 20NEWS and REUTERS contain the documents with the greatest and most varied length whereas DBPEDIA, YAHOO, and AGNEWS are made of rather short documents.

Baselines and SD2C 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 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 (*-e) or the cosine (*-c) distance is used for δ^I, respectively.

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

[2] 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.

[3] http://qwone.com/~jason/20Newsgroups/.

[4] http://www.daviddlewis.com/resources/testcollections/reuters21578/.

- *KM*, *AE-KM* and *DKM*: KM corresponds to k-Means [21] applied on the same input for documents as the one used for SD2C (average of documents' word embeddings); AE-KM first trains an auto-encoder on the collection and then applies k-Means to the document embeddings learned by the auto-encoder; DKM is the deep k-Means algorithm[5] presented in [23] which we also study under the two variants DKM-e and DKM-c.[6]
- *NN*: This method is similar to the 'on-the-fly' nearest neighbor-like classification described in [6]. 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 Sect. 3).
- *STM*: In our experiments, we ran the Java implementation of the Seed-guided Topic Model [17] provided by the authors[7] 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[8].

Seed Word Selection. Recent works on dataless text classification [8,16,17,19] 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 [8]. 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:

$$\text{score}(w,k) = \left(\text{tf}_k(w) - \frac{1}{K-1} \sum_{\begin{cases} k'=1 \\ k' \neq k \end{cases}}^{K} \text{tf}_{k'}(w) \right) \times \text{idf}(w), \tag{8}$$

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.

[5] https://github.com/MaziarMF/deep-k-means.
[6] Seed words are not utilized in these approaches.
[7] https://github.com/ly233/Seed-Guided-Topic-Model.
[8] Very recently, another topic modeling approach, the Laplacian Seed Word Topic Model (LapSWTM), was proposed in [19]. 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.

Table 1. Macro-average results in terms of accuracy (ACC) 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.

Metric	KM	AE-KM	DKM-e	DKM-c	NN	STM	SD2C-Doc-e	SD2C-Doc-c	SD2C-Rep-e	SD2C-Rep-c
ACC	61.4	64.0	**65.5**	65.0	72.6	73.3	73.6	**75.9**	75.6	74.1
ARI	45.4	48.8	**50.4**	48.9	50.9	53.6	56.2	**57.1**	55.7	53.5

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.

Architecture and Hyperparameters. The auto-encoder used in our experiments on all datasets is similar to the ones adopted in prior deep clustering works [12,23,30,31]. 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 auto-encoder embedding space. Neural networks' weights are initialized based on the Xavier scheme [11]. The SD2C, DKM, and AE-KM models are trained with the Adam optimizer [13] 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 auto-encoder pretraining and model finetuning are fixed to 50 and 200, respectively, as in [23]. We also use the inverse temperature $\alpha = 1000$ from [23] for the parameterized softmax-based differentiable reformulations of SD2C models. The balancing hyperparameters λ_0 and λ_1 of SD2C-Doc and SD2C-Rep were both set to 10^{-5}. We experimented with different word embedding techniques including word2vec [22], doc2vec [15], and FastText [4] trained either on an external large corpus (e.g., Google News) or individually on the datasets used in the experiments. We found that training the word embedding models on the experiments' collections consistently improved in terms of clustering performance on external corpus-based training. Among the word embedding techniques we tested, word2vec and FastText performed evenly and significantly better than doc2vec. Since word2vec is faster to train than FastText, which operates at the character level, we chose the former technique (in practice, Gensim[9] word2vec Python implementation) trained on each of our experiments' datasets to compute the word embeddings. The word embedding size was fixed to 100. The Skip-Gram model was trained with a window size of 50 words on 20NEWS and 10 words on other datasets. Note that a word2vec model is trained once for each dataset so that all approaches rely on the same word embeddings.

[9] https://radimrehurek.com/gensim/.

Table 2. Seed-guided constrained clustering results with 3 seed words per cluster. Bold results denote the best, as well as not significantly different from the best, results. Underlined SD2C results indicate a significant improvement over STM.

Model	20NEWS		REUTERS		YAHOO		DBPEDIA		AGNEWS	
	ACC	ARI	ACC	ARI	ACC	ARI	ACC	ARI	ACC	ARI
NN	72.3±0.0	53.2±0.0	79.0±0.0	58.3±0.0	54.7±0.0	26.5±0.0	79.9±0.0	64.7±0.0	77.3±0.0	51.9±0.0
STM	65.7±0.9	47.5±1.0	**83.0±0.7**	**66.3±1.2**	57.1±0.1	29.3±0.2	**80.9±0.4**	**72.7±0.4**	79.7±0.2	55.8±0.3
SD2C-Doc-e	**80.5±0.6**	**66.4±0.7**	66.3±3.7	53.0±3.2	60.4±0.3	**34.3±0.3**	76.1±0.2	63.0±0.2	**84.8±0.2**	**64.4±0.5**
SD2C-Doc-c	77.0±1.5	61.7±2.2	78.1±1.8	60.2±1.1	**61.1±0.8**	**34.4±1.3**	79.4±1.9	66.3±2.1	**84.1±1.1**	**63.3±2.3**
SD2C-Rep-e	*76.1±0.3*	60.1±0.5	80.2±0.8	59.9±0.8	60.2±0.3	**33.5±0.4**	80.3±0.5	67.0±0.6	81.1±0.3	58.1±0.5
SD2C-Rep-c	72.1±1.5	55.7±1.7	81.4±0.7	61.1±0.9	57.8±1.7	29.7±2.7	79.8±1.4	66.1±1.8	79.4±2.0	55.0±3.6

4.1 Results

We measure the clustering performance in terms of clustering accuracy (ACC) and adjusted rand index (ARI), which are standard clustering metrics [5]. Table 1 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) have markedly higher ACC and higher ARI than those which do not (KM, AE-KM, DKM). Among these latter methods, DKM is the best ones (as a comparison, DCN and IDEC, mentioned in Sect. 3, 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.

We further provide in Table 2 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 not statistically different from the best result). As one can note, the proposed SD2C models compare favorably against STM, the strongest baseline. Indeed, all SD2C approaches significantly outperform STM on 20NEWS, and SD2C-Doc-e/c as well as SD2C-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 SD2C methods, SD2C-Doc-c yields the best performance overall (as shown in Table 1).

Runtime. We further compared, in Table 3, the efficiency of STM and the SD2C methods on a machine with eight i7-7700HQ CPUs at 2.80 GHz, 16 GB RAM, and an NVIDIA GeForce GTX 1070 (only used for the deep learning approaches). The runtime of the SD2C approaches is lower than that of STM on most datasets. STM was only faster on AGNEWS (between 2 and 3 times), yet far slower on 20NEWS (about 10 times). This discrepancy can be explained by the fact that STM's complexity is dominated by the total number of tokens

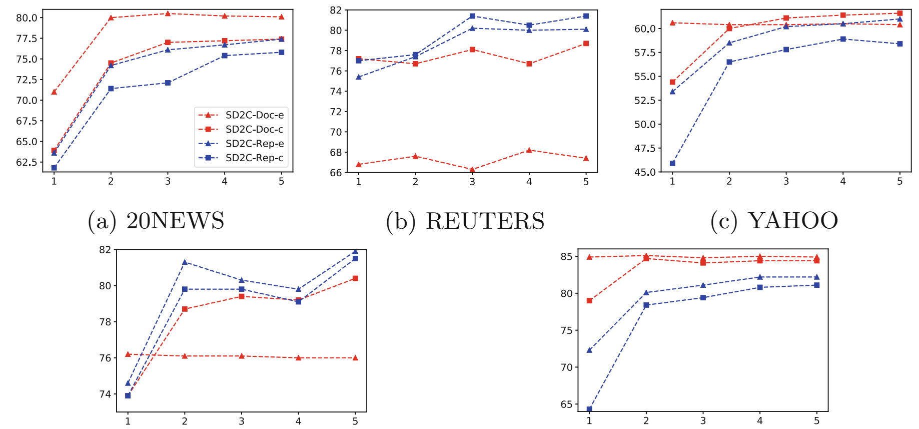
Table 3. Execution time per run (in seconds) for each model on 20NEWS, REUTERS, YAHOO, DBPEDIA, and AGNEWS.

Model	20NEWS	REUTERS	YAHOO	DBPEDIA	AGNEWS
NN	15	3	24	37	22
STM	2008	177	762	686	338
SD2C-Doc	221	72	549	736	1048
SD2C-Rep	186	60	459	590	808

Table 4. Seed-guided constrained clustering results with manual seed words.

Model	20NEWS		REUTERS	
	ACC	ARI	ACC	ARI
STM	66.4±0.2	48.4±0.3	83.2±0.4	68.4±0.8
SD2C-Doc-e	**78.5±0.8**	**64.9±0.9**	67.8±0.4	56.8±2.3
SD2C-Doc-c	75.6±1.0	60.6±1.8	75.5±2.2	61.2±1.7
SD2C-Rep-e	75.2±0.5	60.0±0.4	82.9±0.7	67.7±1.5
SD2C-Rep-c	73.7±0.6	57.5±0.9	**84.2±0.8**	**71.2±1.6**

(large for a small number of documents in 20NEWS), whereas SD2C models only depend on the number of documents (large with few tokens per document in AGNEWS). As to the SD2C models, SD2C-Rep runs faster than SD2C-Doc. This is due to the complexity of the constraint-enforcing loss term being lower for the former than for the latter.



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

(d) DBPEDIA (e) AGNEWS

Fig. 2. 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.

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 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 Fig. 2. We observe that using more seed words leads to notable improvements in most cases – with the exception of SD2C-Doc-e, which seems to be less influenced by the number of seed words. 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 SD2C approaches.

Comparing Automatic and Manual Seed Words. In order to check that the method we retained to automatically extract seed words is appropriate, we also computed the results obtained by STM and the SD2C methods using the manual seed words available for 20NEWS and REUTERS and presented in, *e.g.*, [8,17,28] (denoted as $\mathbb{S}^D$ in the latter). The corresponding list of seed words contain in average 5.1 words per category for 20NEWS and 6.8 words per category for REUTERS. The procedure to constitute these lists of descriptive seed words is detailed in [8]. Table 4 summarizes the results obtained with such seed words. These results first show that the scores obtained by the different methods using the manual seed words are close to the ones obtained with the automatically selected ones. For example, the difference in ACC for STM amounts to only 0.7 points on 20NEWS and 0.2 points on REUTERS. This shows that the automatic selected seed words are a reasonable substitute to manual seed words for evaluation purposes. In addition, SD2C methods still significantly outperform STM on 20NEWS. SD2C-Rep-c is here significantly better, even though the difference is not important, than STM on REUTERS – this is in line with our comment on Table 2 on the small differences between STM and SD2C on REUTERS.

5 Conclusion

We have introduced in this paper the SD2C framework, the first attempt, to the best of our knowledge, to constrain document clustering with seed words using a deep clustering approach. To do so, we have integrated constraints associated to seed words in the Deep k-Means optimization problem [23], modifying either the document embeddings, the cluster representatives or the input representations to make them closer to the seed words retained. The new methods thus derived have been evaluated on five text collections widely used for text classification purposes. For this evaluation, we have proposed a simple method to automatically select seed words that behaves comparably to manual seed words for evaluation purposes.

Several perspectives for this work can be envisaged. First of all, it is possible to extend the current framework with a 'garbage' cluster to collect documents that do not fit well within the clusters defined by the seed words. This can be useful in particular for document filtering [16]. Other types of autoencoders and other attention mechanisms can also be designed to try and improve the results of the SD2C methods. Combinations of the different approaches can also be studied so as to benefit from their respective strengths. Lastly, if the SD2C-Doc-c method overall outperforms the other approaches in terms of accuracy and adjusted rand index, we want to better understand when it is beneficial to bias the document representations and when to bias the cluster representative ones.

Acknowledgment. This research was partly funded by the ANR project LOCUST and the AURA project AISUA.

References

1. Aljalbout, E., Golkov, V., Siddiqui, Y., Cremers, D.: Clustering with deep learning: taxonomy and new methods. arXiv:1801.07648 (2018)
2. Basu, S., Davidson, I., Wagstaff, K.: Constrained Clustering: Advances in Algorithms, Theory, and Applications, 1st edn. Chapman & Hall/CRC, Boca Raton (2008)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
4. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguist. **5**, 135–146 (2017)
5. Cai, D., He, X., Han, J.: Locally consistent concept factorization for document clustering. IEEE Trans. Knowl. Data Eng. **23**(6), 902–913 (2011)
6. Chang, M.W., Ratinov, L., Roth, D., Srikumar, V.: Importance of semantic representation: dataless classification. In: Proceedings of AAAI, pp. 830–835 (2008)
7. Chang, Y., Chen, J., Cho, M.H., Castaldi, P.J., Silverman, E.K., Dy, J.G.: Multiple clustering views from multiple uncertain experts. In: Proceedings of ICML, pp. 674–683 (2017)
8. Chen, X., Xia, Y., Jin, P., Carroll, J.: Dataless text classification with descriptive LDA. In: Proceedings of AAAI, pp. 2224–2231 (2015)
9. Druck, G., Mann, G., Mccallum, A.: Learning from labeled features using generalized expectation criteria. In: Proceedings of SIGIR, pp. 595–602 (2008)
10. Gliozzo, A., Strapparava, C., Dagan, I.: Improving text categorization bootstrapping via unsupervised learning. ACM Trans. Speech Lang. Process. **6**(1), 1–24 (2009)
11. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of AISTATS, pp. 249–256 (2010)
12. Guo, X., Gao, L., Liu, X., Yin, J.: Improved deep embedded clustering with local structure preservation. In: Proceedings of IJCAI, pp. 1753–1759 (2017)
13. Kingma, D.P., Ba, J.L.: Adam: a method for stochastic optimization. In: Proceedings of ICLR (2015)
14. Ko, Y., Seo, J.: Learning with unlabeled data for text categorization using bootstrapping and feature projection techniques. In: Proceedings of ACL, pp. 255–262 (2004)
15. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of ICML, pp. 1188–1196 (2014)
16. Li, C., Chen, S., Xing, J., Sun, A., Ma, Z.: Seed-guided topic model for document filtering and classification. ACM Trans. Inf. Syst. **37**(1) (2018)
17. Li, C., Xing, J., Sun, A., Ma, Z.: Effective document labeling with very few seed words: a topic model approach. In: Proceedings of CIKM, pp. 85–94 (2016)
18. Li, C., Zhou, W., Ji, F., Duan, Y., Chen, H.: A deep relevance model for zero-shot document filtering. In: Proceedings of ACL, pp. 2300–2310 (2018)
19. Li, X., Li, C., Chi, J., Ouyang, J., Li, C.: Dataless text classification: a topic modeling approach with document manifold. In: Proceedings of CIKM, pp. 973–982 (2018)
20. Liu, B., Li, X., Lee, W.S., Yu, P.S.: Text classification by labeling words. In: Proceedings of AAAI/IAAI, pp. 425–430 (2004)

21. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of BSMSP, pp. 281–297 (1967)
22. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of NeurIPS, pp. 3111–3119 (2013)
23. Fard, M.M., Thonet, T., Gaussier, E.: Deep k-means: jointly clustering with k-means and learning representations. arXiv:1806.10069 (2018)
24. Niu, D., Dy, J.G., Ghahramani, Z.: A nonparametric Bayesian model for multiple clustering with overlapping feature views. In: Proceedings of AISTATS, pp. 814–822 (2012)
25. Niu, D., Dy, J.G., Jordan, M.I.: Multiple non-redundant spectral clustering views. In: Proceedings of ICML, pp. 831–838 (2010)
26. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: Proceedings of EMNLP, pp. 1532–1543 (2014)
27. Shental, N., Bar-Hillel, A., Hertz, T., Weinshall, D.: Computing Gaussian mixture models with EM using equivalence constraints. In: Proceedings of NeurIPS, pp. 465–472 (2003)
28. Song, Y., Roth, D.: On dataless hierarchical text classification. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence, AAAI 2014, pp. 1579–1585 (2014)
29. Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S.: Constrained K-means clustering with background knowledge. In: Proceedings of ICML, pp. 577–584 (2001)
30. Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: Proceedings of ICML, pp. 478–487 (2016)
31. Yang, B., Fu, X., Sidiropoulos, N.D., Hong, M.: Towards K-means-friendly spaces: simultaneous deep learning and clustering. In: Proceedings of ICML, pp. 3861–3870 (2017)
32. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Proceedings of NeurIPS, pp. 649–657 (2015)