# Dialog Clustering: a Framework for Automatic Text Clustering[⋆]

Massarenti Nicola[1][0000−0002−8882−4252] and Lazzarinetti Giorgio[1][0000−0003−0326−8742]

Noovle S.p.A, Milan, Italy https://www.noovle.com/en/

**Abstract.** Despite the recent advances in Natural Language Processing techniques, when it comes to unsupervised applications, such as text clustering, many critical issues arise, as evaluating the results of an unsupervised algorithm or making an unsupervised algorithm work in an automatic fashion. In this research we propose an innovative framework based on machine learning for the creation of a system capable of addressing these critical issues. The framework consists of four main steps: a step for converting text into embedding, a step to reduce the dimensionality of the generated embedding, a step for clustering data and, finally, a step to evaluate the results obtained and selecting the best clustering produced. Each of these steps uses different models equipped with a mechanism for identifying and automatically selecting hyperparameters. Our framework guarantees good performance in clustering text without an a-priori knowledge of the data. Observing the results it can be seen how the subdivisions of the dataset vary according to the models used in the different steps of the proposed system and how groups focused on specific themes are identified.

**Keywords:** Natural Language Processing · Text Clustering · Dimensionality Reduction · Text Embedding.

## 1 Overview

In recent years, thanks to the advances in Natural Language Processing (NLP) techniques [1–3], an ever increasing number of text applications, as part-of-speech tagging [6], text classification [4], document summarization [4], named entity recognition [7], text clustering [9], machine translation and chatbots, have been developed [5]. Among these, the use of chatbots has been very popular since its inception in 1960. In the past several years, giant companies have invested

⋆ Activities were partially funded by Italian ”Ministero dello Sviluppo Economico”, Fondo per la Crescita Sostenibile, Bando “Agenda Digitale”, D.M. Oct. 15th, 2014 - Project n. F/020012/02/X27 - “Smart District 4.0”.

in artificial intelligence and developed conversational engines (such as Google DialogFlow, Amazon Lex, Azure Bot Service) that allow customers to create their own personalized dialog system [8]. However, the design of the conversational flow is complex, often error-prone and incomplete [65]. To enhance their creation many companies make use of existing unstructured corpora of text, coming from real conversations with customers but, the exploration and exploitation of such corpora is difficult, since data is unstructured and noisy. A better understanding of such data can be obtained by using unsupervised machine learning algorithms, such as clustering, which, indeed, aims at detecting patterns and getting insights from data. Text clustering applications, given their unsupervised nature, however are not always directly measurable and need to involve humans to explain the results. Thus, the goal of this research is that of developing a methodology for creating text clustering addressing these issues and trying to automate a process that has until now been manual. This research is driven by the business need of a partner company that aims at creating an application to continuously improve an existing conversational customer service agent able to assist customers to manage and maintain the machine they produce. The goal is that of collecting all the chatbot's fallbacks (i.e. questions to which the chatbot is not able to give and answer) and use them subsequently to derive new intents to re-train the chatbot. Thus, we design a methodology to automatically create and evaluate clusters of text coming from conversations, in order to derive from the corpus new possible intents. The developed framework is composed of a multi-step pipeline with an embedding step for creating text embeddings, a dimensionality reduction step, a clustering step and a final evaluation to select and evaluate the clusterings produced. The rest of this paper is organized as follows. In Chapter 2 an overview of the state of the art is presented, with a focus on text embedding, dimensionality reduction and clustering. In Chapter 3 the pipeline that enables the creation of the automatic clustering framework is defined and in Chapter 4 the experimental results obtained are shown. Finally in Chapter 5 some conclusive marks.

## 2 State of the Art

In the context of text clustering [9], the critical points for the realization of a clustering system are three: the conversion of the text into a set of features, a.k.a embedding, the dimensionality reduction of the features and the clustering algorithm itself [10]. In the following we focus on these main issues.

### 2.1 Text Embedding

The representation of words and documents is a fundamental activity in NLP applications and for years the predominant methodology for this task has been that of the Vector Space Model (VSM) [11]. The intuition is to produce an encoding of words and documents in t-dimensional vectors, where each element represents a unique term contained in the documents. VSM can be performed considering global information (i.e. all the corpus of text) often via a co-occurrence matrix, or local information (i.e. words and their context within a sentence) via the use of neural networks. One of the first approaches to global information-based VSM is the calculation of the term frequency - inverse document frequency (TF-IDF) statistic which, given a corpus of texts, weights the values of the t-dimensional vector with respect to the word in the document, and with respect to the frequency in all documents of the corpus [12]. More recently GloVe [14] has been introduced, which performs an encoding of semantic relationships between words by leveraging the intuition that co-occurrence relationships between word pairs are more informative than simple counting. On the other hand, the first important contributions to modern local information-based VSM were the continuous bag-of-words (CBOW) and the skip-gram (SG) models [15]. These models have been used for the development and dissemination of Word2Vec [16, 17] a model that leverages neural networks' hidden layers representation of the feature. In 2016 Facebook Inc published FastText [18], where they proposed to use the n-grams [19, 20], thus including in the embedding the morphology and the composition of words. In 2018 three important models have been published: ELMo [21, 22], a bidirectional Long Short Term Memory (LSTM) model, Universal Sentence Encoder (USE) [48], a model developed specifically for the production of sentence embedding and Bidirectional Encoder Representations for Transformers (BERT) [23], a Google's model pre-trained on Wikipedia and BooksCorpus [49] that uses transformers [24, 25] in a bidirectional architecture based on Recurrent Neural Network (RNN) that implements an attention mechanism for contextual representations of sentences. Following BERT there have been several models trained on other corpora, as AlBERTo [26], a model specifically for the italian language.

### 2.2 Dimensionality Reduction

Dimensionality reduction is the process of transforming multidimensional variables from vector spaces (VS) with a large number of dimensions to VSs with a significantly reduced number of dimensions so that the intrinsic and most significant properties of the variables are preserved [28]. One of the best known dimensionality reduction techniques is Principal Component Analysis (PCA) [27, 29] a statistical technique that determines the principal components (PC) by carrying out a linear transformation of the variables such as to maximize the variance [30]. PCA has proven to be effective in many applications even if, however, since it does not take into account the distribution of data sets it has been extended to Kernel PCA which is the reformulation of PCA obtained by applying a kernel function, used in order to take advantage of the kernel trick [31, 32]. Other effective approaches are Isomap [33] and UMAP [34], which are manifold-based dimensionality reduction methods. Finally, more recently, Autoencoders [35], based on multiple feed-forward neural networks have also been applied to reduce the dimensionality of the data. The main criticality in the use of autoencoders to reduce dimensionality lies in the training process of the neural network which requires a large dataset.

## 2.3 Clustering

Clustering is an unsupervised machine learning technique that aims at grouping data points into two or more sets so that data in the same set are closer to each other with respect to the data points in the other set [36, 37, 39]. The clustering models are mainly divided into 8 categories as described in Table 1.

**Table 1.** Clustering Algorithms per category.

| Clustering category | Description | Algorithms |
|---|---|---|
| Partition based | base their theory by designating the data center as the corresponding cluster center | K-means [39] |
| Hierarchy based | build hierarchical relationships between the data to be clustered and, on the basis of these, they separate the datar | BIRCH [40], CURE and ROCK [36] |
| Fuzzy theory based | based on the idea that the discrete value of belonging to a cluster, 0, 1, is to be replaced with the continuous interval [0, 1] in order to describe the relationships of belonging of data to clusters | FCM [41] and FCS [42] |
| Distribution based | base the clustering of data on the statistical distribution of features | Gaussian Mixture Model (GMM) [43] |
| Density based | base their output on the idea that regions with high density are those that contain data belonging to the same cluster | DBSCAN [44], OPTICS [45] and Mean-shift [46] |
| Graph theory based | based on graphs, where nodes are the data and links are the relationships between the data | CLICK [36] |
| Grid based | change the original data space into a grid structure | STING and CLIQUE [47] |
| Fractal based | based on fractal theory | FC [36] |

Partition based models have low complexity but they have poor performance in case of non-convex data or with many outliers and, as hierarchy based models, they require the number of clusters to be defined in advance. Fuzzy theory based models, on the other hand, have poor scalability and strong sensitivity to the hyperparameters of the models. On the contrary, the distribution based models have a higher scalability but they strongly depend on the hyperparameters too. The same holds for density based models, which however are very efficient and suitable for data with arbitrary topologies. Graph theory based models have high accuracy and efficiency of clustering, however the complexity increases significantly as the complexity of the graph increases. Grid-based models are highly scalable and characterized by low complexity but the output is very sensitive to the granularity (mesh size). Finally, fractal theory based models have the advantage of having linear complexity with the number of data, however they are not enough performing to compete with the other clustering techniques.

## 3 Methodology

To understand how to create a system that, starting from a corpus of text, trains different clustering models using hyper parameter optimization techniques and selects the best ones based on a series of combined evaluation metrics, it is firstly necessary to focus on the dataset provided to implement the solution.

### 3.1 Dataset Description

The available dataset is associated with a virtual agent structured in intents, according to the business needs of the partner company.The dataset is composed of 1297 phrases, used by the conversational engine for training the model, representative of 140 intents with an average number of training phrases for each intent equal to 9.26. The main topics concern document management, the resolution of problems encountered with machinery, contract management and restoration operations.

### 3.2 Clustering Pipelines

The results of the clustering system are computed by pipelines $P_i$, which are defined by the set of models implemented and their configurations. More formally:

$$P_i = \left\{E_e^l, D_d^m, C_c^n\right\} \tag{1}$$

where $E_e \in \{E_1, ..., E_E\}$ is the e-th embedding model and $E_e^l \in \left\{E_e^1, ..., E_e^L\right\}$ is the l-th configuration of the model $E_e$; $D_d$ is the d-th dimensionality reduction model and $D_d^m$ is the m-th configuration; $C_c$ is the c-th clustering model and $C_c^n$ is the n-th configuration. At the end of the executions of all the pipelines different clusterings are obtained which are analyzed and evaluated in order to choose the ones that best adhere to the data, that is, those that have the best evaluation.

### 3.3 Embedding

The first step of the proposed methodology is represented by the application of a set of the embedding models to the input dataset. This step is carried out over a dataset cleaned via stop words elimination and pre-processed via stemming, in order to reduce the noise present in the data. At this step three different embeddings are carried out: BERT, AlBERTo and USE. These models, indeed, represent the state of the art for embedding production and, moreover, they are equipped with tools suitable to support their industrialization. Indeed all the models are written in Python [51] and uses the TensorFlow framework [52] which, through theTensorFlow Hub module [53] (except for ALBERTo [56]), allows to download the pre-trained models (BERT [54], ALBERTo and USE [57]) and the pre-processing [55, 56] used for tokenization. BERT expects the inputs to be pre-processed and structured according to specific requirements. For each input phrase it is required to divide the words into sequences of tokens and to add the token [CLS] at the beginning of the sequence and the token [SEP] at the end. BERT uses the WordPiece tokenizer [50], whose vocabulary is initialized with all individual characters and augmented with the most frequent and probable word combinations. AlBERTo implements both the model and the tokenization and pre-processing. USE is a model based on the Transformer architecture that transforms the sentences into 512-dimensional embeddings.

## 3.4 Dimensionality reduction

The second step of the proposed methodology is represented by the application of a set of dimensionality reduction models to each of the embedding produced at the previous step. At this step three different dimensionality reduction techniques are implemented PCA, Kernel PCA and UMAP. With respect to hyperparameters selection, these model can be of two different types: subject to hyper parameter tuning (auto-tunable), i.e. there is a methodology to automatically select the optimal hyperparameters, or not subject to hyperparameters tuning (non-auto-tunable), i.e. there is not a methodology to automatically select the optimal hyperparameters. The same considerations hold for the clustering algorithms. Following the execution of the dimensionality reduction model, the standardization of the features is performed to project the data into a uniform and independent space from the pipeline for the next step of clustering.

**PCA** The first dimensionality reduction model used by the proposed system uses the implementation of the Incremental PCA [58] which, differently from PCA, does not process all the dataset in a single run, but computes the PC incrementally, thus limiting the use of the required memory without placing constraints on the size of the dataset. This method is non-auto-tunable, thus to determine the number of principal components to reduce the dimensionality, we select all and only the main components that describe the 90% of the variance. The data is processed in blocks of 5000.

**Kernel PCA** As mentioned, Kernel PCA is an extension of the PCA dimensionality reduction model based on the use of kernel methods. Kernel PCA is auto-tunable, i.e. the best configuration among all those available can be automatically searched. The methodology applied for the identification and automatic selection of hyperparameters of the kernel PCA algorithm is a variant of the one presented in [59]. Specifically, after defining $K_1, ...K_K$ kernel PCA configurations to be explored during hyper-parameter tuning, the set of configurations is selected such that:

$$K_{\bar{k}} | \bar{k} = arg \max_{i|1 \leq i \leq K} score(Ki) \tag{2}$$

That is, defined the original dataset $\boldsymbol{X}$, for each configuration $K_i, 1 \leq i \leq K$ the i-th configuration is applied such that:

$$\boldsymbol{I_i} = kernelPCA(K_i, \boldsymbol{X}) \tag{3}$$

Once the image $\boldsymbol{I_i}$ of $\boldsymbol{X}$ is obtained through the kernel function defined by the configuration $K_i$, the eigenvalues and eigenvectors of $\boldsymbol{I_i}$ are computed. If the number of eigenvalues $N_i$ required to describe 90% of the variance of $\boldsymbol{I_i}$ is greater than the number of components of the original feature space, a very low score is assigned, otherwise, the new feature space $\boldsymbol{I_i^{N_i}}$ obtained by applying Kernel PCA with the configuration $K_i^{N_i} = (K_i, N_i)$ is calculated. The selection of the best set of hyperparameters is done by evaluating the reconstruction error. An inverse image $\boldsymbol{Z_i}$ of the variable $\boldsymbol{I_i^{N_i}}$ is defined such that:

$$\boldsymbol{Z_i} = H^{-1}(\boldsymbol{I_i^{N_i}}) \tag{4}$$

where $H^{-1}$ is the function that determines the inverse image. The reconstruction error is calculated as:

$$\boldsymbol{E_i} = d(\boldsymbol{X}, \boldsymbol{Z_i}) \tag{5}$$

where $d(\boldsymbol{X}, \boldsymbol{Z_i})$ is the Manhattan distance. Following the calculation of the reconstruction error, the score associated with the configuration is calculated as follows:

$$K_i|score_i = -\sum_{1\leq n\leq|\boldsymbol{X}|}\sum_{1\leq m\leq N_i} \boldsymbol{E_i}(n, m) \tag{6}$$

An application problem that arises with Kernel PCA in cases where the size of the input is high is how to mediate the management of parallelization of executions with the management of available memory. To meet this need we compute an heuristics based on the available memory, the number of available processors and the size of the input.

**UMAP** Finally, UMAP [34], is a dimensionality reduction model based on the theoretical framework of Reimann geometry and topological algebra that builds a high-dimensional graph where the weights of the links between two nodes represent their probability of being connected. The connectivity of the graph is determined by defining a circle around each node and connecting those with overlapping circles. UMAP guarantees the conservation of the local structure through a mechanism that binds the connection of each point to the one closest to it. Following the construction of the high-dimensional graph, UMAP optimizes its structure to reduce the components and creates an analogous graph of reduced dimensionality as similar as possible to the one originally created. This module is non-auto-tunable, thus we set two configurations chosen empirically: the first foresees to implement the model with the parameter *n neighbors* equal to 5 and *n components* equal to 30, the second foresees to set *n neighbors* equal to 15 and *n components* equal to 50.

## 3.5 Clustering

The third step of the methodology is that of the computation of a set of clustering algorithms for each dataset produced by the dimensionality reduction step. Given the large number of models available for clustering, for each clustering method, it is necessary to have a methodology for automatic selection of the optimal configuration of the hyperparameters. For this reason, we decided to implement only auto-tunable clustering algorithms.

**K-means** K-Means [39] bases its clustering strategy on the updating of cluster centroids through an iterative process carried out until convergence. The process of convergence of the algorithm performs initialization (1), assignment of clusters (2) and updating of centroids (3). While the initialization, which involves the random assignment of the centroids, is performed only once, the step of assigning the data to the nearest centroid and that of recomputing the position of the centroids are carried out until the difference between the last centroids obtained and those calculated at the previous iteration is negligible, i.e. when convergence is reached. K-Means, requires the definition of the number of clusters in advance, thus we implement an optimization process to determine the best number of clusters. The process involves performing different clustering each with a different and progressive number of clusters. For each clustering the distances of the points from the centroid of the cluster to which they have been assigned are calculated. Finally, the Within Cluster Sum of Squares (WCSS) value of each configuration is calculated as the sum for all clusters of the sum of the square Manhattan distance from each point to the centroid of the cluster it is assigned. Given the set of WCSSs for each clustering produced (with different number of clusters) the elbow point is sought through the implementation of the kneedle algorithm [60], then the best configuration is identified and, consequently, the optimal number of clusters.

**OPTICS** OPTICS [45] is based on the following strategy: a point belongs to a cluster if it is close to other points belonging to that cluster. The key parameter of OPTICS is minPts which identifies the minimum number of points required to define a cluster. Thanks to minPts, OPTICS defines three types of points: core points, border points and outliers. The core points are those points that are close to at least minPts points, the border points are those reachable from a core point but which have no minPts points. The outliers, on the other hand, are the points that are neither core points nor border points. To define minPts we develop an automatic mechanism that involves the execution of different configurations that explore the clustering determined with different minPts and then the assignment of a score determined through the analysis of a general evaluator as described in Paragraph 3.6.

**Spectral Clustering** Spectral Clustering [39] is a graph based model that determines how to cluster data by analyzing the spectrum of the similarity matrix. Given an enumerated set of data points, the similarity matrix may be defined as a symmetric matrix $A$, where $A_{i,j} > 0$ represents a measure of the similarity between data points with indices $i$ and $j$. The general approach to spectral clustering is to use K-means on the relevant eigenvectors of the Laplacian matrix of $A$. The eigenvectors that are relevant are the ones that correspond to the smallest eigenvalues of the Laplacian except for the smallest eigenvalue which will have a value of 0. One of the open themes of Spectral Clustering is managing data at multiple scales. For this reason we introduce a local scaling mechanism, originally presented in [61], to improve data representation. The introduction of local scaling involves calculating the affinity of two points using the Manhattan distance and the local scale obtained by applying k-nearest neighbor [62] with $k = 7$ as described in [61]. Each clustering produced is then associated with a score assigned by a general evaluator as described in Paragraph 3.6: at the end of the iterations the clustering with the best score is chosen.

## 3.6 Clustering evaluation and selection

The unsupervised clustering system presented in this research aims at determining the best $N$ clustering among those produced by the $I$ pipelines. In order to achieve this goal we designed a general evaluator, who determines the goodness of each clustering with respect to the characteristics of cohesion and separation. Moreover, we designed a relative cohesion evaluator, which determines the cohesion of clusters with respect to a clustering set and a relative separation evaluator, which determines the separation between clusters against a clustering set. The procedure for selecting the best clustering involves to select, among all clusterings produced by the pipelines, just the first 10 with the best score assigned by the general evaluator. The relative cohesion evaluator and the relative separation evaluator are also computed to give the user the possibility of furtherly exploring the results and selecting the best clustering according to their needs. As general evaluator we use the Silhouette [63] defined as:

$$sil(x_i) = \frac{b(x_i) - a(x_i)}{\max\{a(x_i), b(x_i)\}} \tag{7}$$

where $a(x_i)$ is the average distance of $x_i$ from the other points belonging to the same cluster and $b(x_i)$ represents the distance between $x_i$ and the nearest cluster, calculated as the average distance of $x_i$ from all points belonging to the closest cluster. The range of values returned by the silhouette metric is [-1,1], where good clustering for the point is achieved when $sil(x_i) \to 1$ . The silhouette score of all clustering is obtained by averaging the silhouette scores of the points in the dataset.

The relative cohesion evaluator determines the cohesion of the clusters produced by a model and normalizes this value with respect to the cohesion values of a clustering set. Given a set of clustering $R_1, ..., R_k$, defined $C^k$ the clusters created by the k-th clustering and $X$ the dataset shared by all clusterings, the cohesion associated with clustering $t$ is defined as:

$$\theta(R_t) = \sqrt[2]{\frac{1}{|C^t|} \sum_{C_i \in C^t} \big(\frac{1}{m_i} \sum_{x \in C_i} d(x, c_i)\big)^2} \tag{8}$$

where $m_i$ is the number of points belonging to the i-th cluster, $c_i$ is the centroid of the i-th cluster and $d(x, y)$ is the Manhattan distance. The relative cohesion is calculated using the following equation:

$$\theta_{rel}(R_t) = 1 - \frac{\theta(R_t) - min\{\theta(R_1), ..., \theta(R-k)\}}{max\{\theta(R_1), ..., \theta(R-k)\} - min\{\theta(R_1), ..., \theta(R-k)\}} \tag{9}$$

Similarly, the relative separation evaluator determines the separation between clusters and normalizes that value against the separation values of a clustering set. Thus, given $c_i$ the centroid of cluster $C_i$, the separation associated with clustering is

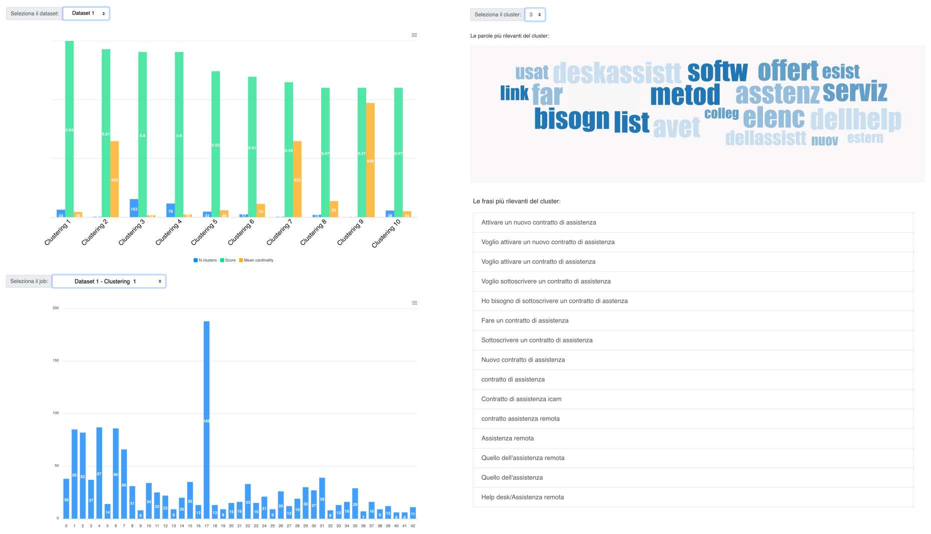
$$\lambda(R_t) = \sqrt[2]{\frac{1}{|C^t|} \sum_{C_i \in C^t} m_i d(c_i, c)^2} \tag{10}$$

from which to compute $\lambda_{rel}(R_t)$ in the same way as for relative cohesion.

## 4 Experimental results

As explained, the goal of this research is to create a framework for the automatic creation and evaluation of text clustering. The goal is not that of producing a unique clustering, but a set of clusterings with some descriptive evaluation metrics that the final user can exploit to explore the corpus of text and derive a set of intents for the re-training of a chatbot system or even for the first training. Indeed, the corpus of text comes from the chatbot's fallback, i.e. a set of questions to which the chatbot was not able to find an answer. Thus, in order to let the framework be usable, we design a web page that allows the user to upload the corpus of text, run the clustering methodology and visualize just the first 10 clusterings with best evaluation metrics. In order to visualize the results three main plots are shown in the web page: a bar plot that show for each clustering the number of clusters, the general evaluation score and the mean cardinality of clusters (Figure 1 top-left); a bar plot that for a specific clustering that can be selected show the cardinality of each cluster (Figure 1 bottom-left); a word cloud plot that for a specific cluster selected from the previous bar plot shows the main words of that cluster and the main sentences (Figure 1 top/bottom-right).

After running the framework with the input dataset we have analyzed the 10 clustering produced. It is clear that the evaluation of the entire methodology cannot be done in a quantitative way as for supervised machine learning algorithms. Indeed, even if the dataset used for the experiment has a ground truth (each sentence has been manually associated with an intent) we cannot state that the only acceptable results produced by the clustering algorithms is that defined by the ground truth, since the goal of unsupervised algorithms is indeed that of finding hidden structure in the original dataset. Thus, we manually explore each clustering produced to see if the results are coherent and it emerged that apart from some clustering with a very small number of large clusters, the other clustering produced actually manage to divide the dataset in group of text with the same meaning or related to the same topic. Indeed, of the


Seleziona il dataset: Dataset 1
Clustering 1
Clustering 2
Clustering 3
Clustering 4
Clustering 5
Clustering 6
Clustering 7
Clustering 8
Clustering 9
Clustering 10
N clusters
Score
Mean cardinality
Seleziona il job: Dataset 1 - Clustering 1
Seleziona il cluster: 3
Le parole più rilevanti del cluster:
usat deskassistt softw offert esist link far metod assтenz serviz bisogn list avet colleg elenc dellhelp dellassistt nuov estern

Le frasi più rilevanti del cluster:

Attivare un nuovo contratto di assistenza
Voglio attivare un nuovo contratto di assistenza
Voglio attivare un contratto di assistenza
Voglio sottoscrivere un contratto di assistenza
Ho bisogno di sottoscrivere un contratto di asstenza
Fare un contratto di assistenza
Sottoscrivere un contratto di assistenza
Nuovo contratto di assistenza
contratto di assistenza
Contratto di assistenza icam
contratto assistenza remota
Assistenza remota
Quello dell'assistenza remota
Quello dell'assistenza
Help desk/Assistenza remota

**Fig. 1.** Bar plot of the meaningful clustering produced by the pipeline.

10 kept clustering, 5 are meaningful and contains specific concept related to the use and maintenance of the machine. These 5 clusterings have approximately 30 to 100 clusters. Of these clusterings, three have less than 50 clusters, while two have respectively 78 and 103 clusters. These two clustering are even more fine, since they are able to distinguish even sentences with similar word but different meaning. As far as the other clustering with less than 30 clusters is concerned, instead, they contain some very large clusters which are noisy and some small clusters which are meaningful. Therefore, these clustering seem to be less useful than the others. Even though the evaluations are empirical, we can state that the methodology is able to produce meaningful clusterings and the framework created allows to exploit such clusterings to detect topics and derive intents.

To conclude, we also make some experiment to define the required hardware needed to run the methodology. In particular, we run the methodology over a Google Cloud Compute Engine [64] with 8 vCPUs and 32 GB of RAM. We test the performance of the framework in terms of execution time, CPU and RAM utilization with two different dataset, one with 5000 sentences and one with 20000 sentences. In Table 2 the results are shown. It can be seen that in the first case, the duration of the pipeline and the use of resources is limited and almost never reach 100%. In the second case, the use of resources extremely increases and reaches 100% of CPU utilization in 2 cases: Kernel PCA and Spectral Clustering. Moreover, for these two algorithms, the time extremely increases. As far as the created framework is concerned, in order to limit the use of resources, we set a limit to the input dataset to 20000 rows, to avoid exhausting resources. This limit is actually enough for the partner company and for the specific use case.

**Table 2.** Resource consumption

| | | **Dataset 5k** | | | **Dataset 20k** | | |
|---|---|---|---|---|---|---|---|
| | | Durata (h) | %CPU | %RAM | Durata (h) | %CPU | %RAM |
| **Embedding** | BERT | 0.2 | 82 | 12 | 0.82 | 80 | 14 |
| **Dimensionality Reduction** | PCA | 0.02 | 80 | 10 | 0.02 | 76 | 4 |
| | Kernel PCA | 0.13 | 100 | 15 | 6.25 | 100 | 50 |
| | UMAP | 0.02 | 13 | 3 | 0.03 | 15 | 3 |
| **Clustering** | K-means | 0.07 | 90 | 6 | 0.6 | 91 | 2 |
| | OPTICS | 0.07 | 16 | 5 | 0.57 | 100 | 8 |
| | Spectral Clustering | 1.07 | 13 | 6 | 12.3 | 17 | 12 |

## 5 Conclusion and future works

This study presents a methodology based on machine learning for the realization of an unsupervised clustering system. The identified methodology consists of an embedding step, composed of the non-auto-tunable models BERT, AlBERTo and USE, a dimensionality reduction step, that consists of the non-auto-tunable PCA and UMAP models and the auto-tunable Kernel PCA model, the clustering step, composed of the auto-tunable K-Means, OPTICS and Spectral Clustering models and a final evaluation step, where the metric used to select the best clusterings provides a score with respect to both cohesion and separation of clusters (further evaluations are obtained through metrics that focus only on cohesion and separation of clusters). Ultimately, the developed methodology shows good performance with respect to the clustering goal, even though performance considerations are empirical. In conclusion, the methodology is effective in dividing datasets into clusters, provides more than one clustering and accompanies the results with the metadata of the evaluators used. In general, it appears to be a performing system to be used in an industrial context. Its main potential and innovativeness lies in the automatic system for selecting the hyperparameters and the best clustering: through these mechanisms, the sequences of models and hyperparameters that are most adherent and suitable for the dataset are chosen. Thanks to this mechanism it is, indeed, possible to automatically obtain the clustering of data never explored and extract useful information for business logic. Even though the framework created works fine, some enhancement can be performed. In particular, from the clustering logic perspective, the pipeline can be enriched with other algorithms for the three different steps. However, the main enhancement to extend the framework are related to the creation of an auto scaling infrastructure that is able to keep the training time low by increasing the resources, so that the framework is no limited in the dimension of the input and does not require the user to wait for long for the creation of the clustering.

## References

1. Hindle, A., Barr, E., Gabel, M., Su, Z., Devanbu, P.: On the naturalness of software. Communications of the ACM **59**, 122–131 (2016)
2. Otter, D. W., Medina, J. R., Kalita, J. K.: A Survey of the Usages of Deep Learning for Natural Language Processing. IEEE Transactions on Neural Networks and Learning Systems **32**(2), 604–624 (2021)
3. Cai, T., Giannopoulos, A., Yu, S., Kelil, T., Ripley, B., Kumamaru, K., Rybicki, F., Mitsouras, D.: Natural Language Processing Technologies in Radiology Research and Clinical Applications. Radiographics: a review publication of the Radiological Society of North America **36**, 176–191 (2016)
4. Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Presa, R. M., Shyu, M.-L.,Chen, S.-C., Iyengar, S.: A Survey on Deep Learning: Algorithms, Techniques, and Applications. ACM Computing Surveys **51**, 1–36 (2018)
5. Zhang, X., Zhao, J., Lecun, Y.: Character-level Convolutional Networks for Text Classification. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, pp. 649–657. MIT Press, Cambridge, MA, USA (2015)
6. Plank, B.,Søgaard, A., Goldberg, Y.: Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pp. 412–418. Association for Computational Linguistics, Berlin, Germany (2016)
7. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural Architectures for Named Entity Recognition. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 260–270. Association for Computational Linguistics, San Diego, California (2016)
8. Ait-Mlouk, A., Jiang, L.: KBot: A Knowledge Graph Based ChatBot for Natural Language Understanding Over Linked Data. IEEE Access (2020)
9. Xu, D., Tian, Y.: A Comprehensive Survey of Clustering Algorithms. Annals of Data Science **2**, 165–193 (2015)
10. Abualigah, L., Khader, A. T., Al-Betar, M.,Hanandeh, E.:. Unsupervised Text Feature Selection Technique Based on Particle Swarm Optimization Algorithm for Improving the Text Clustering. In: Proccedings of First EAI International Conference on Computer Science and Engineering, EAI (2017)
11. Felipe, A., Geraldo, X.: Word Embeddings: A Survey. arXiv preprint, arXiv:1901.09069 (2019)
12. Qaiser, S., Ali, R.: Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. International Journal of Computer Applications **181**(1) (2018)
13. Arora, S., Li, Y., Liang, Y., Ma, T., Risteski, A.: A Latent Variable Model Approach to PMI-based Word Embeddings. Transactions of the Association for Computational Linguistics **4** 385-399 (2016)
14. Pennington, J., Socher, R., Manning, C.: Glove: Global Vectors for Word Representation. EMNLP **14** 1532-1543 (2014)
15. Goodfellow, I., Bengio, Y., Courville, A.: Deeplearning. MITpress (2016)
16. Google Code Archive - Long-term storage for Google Code Project Hosting. https://code.google.com/archive/p/word2vec/. Last accessed 14 Sept 2021.
17. Word2vec - Wikipedia. https://it.wikipedia.org/wiki/Word2vec. Last accessed 14 Sept 2021.
18. FastText - Facebook Research. https://research.fb.com/blog/2016/08/fasttext/. Last accessed 14 Sept 2021.

19. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2016)
20. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of Tricks for Efficient Text Classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, pp. 427–431. Association for Computational Linguistics, Valencia, Spain (2017)
21. Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 2227–2237. Association for Computational Linguistics, New Orleans, Louisiana (2018)
22. Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., Huang, X.: Pre-trained Models for Natural Language Processing: A Survey. Science China Technological Sciences **63**,1872–1897 (2020)
23. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (2018)
24. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I.: Attention is All You Need. ArXiv preprint, arXiv:1706.03762 (2017)
25. Hu, D.: An Introductory Survey on Attention Mechanisms in NLP Problems. In: Bi Y., Bhatia R., Kapoor S. (eds) Intelligent Systems and Applications. IntelliSys 2019. Advances in Intelligent Systems and Computing, vol 1038. Springer, Cham. https://doi.org/10.1007/978-3-030-29513-4
26. Polignano, M., Basile, P., de Gemmis, M., Semeraro, G., Basile, V.: ALBERTO: Italian BERT Language Understanding Model for NLP Challenging Tasks Based on Tweets. In: Proceedings of the Sixth Italian Conference on Computational Linguistics (CLiC-it 2019), volume 2481. CEUR (2019)
27. Cunningham, J., Ghahramani, Z.: Linear Dimensionality Reduction: Survey, Insights, and Generalizations. Journal of Machine Learning Research **16**, 2859–2900 (2015)
28. Xie, H., Li, J., Xue, H.: A survey of dimensionality reduction techniques based on random projection. ArXiv prerprint, arXiv:1706.04371 (2017)
29. Gadekallu, T., Reddy, P., Lakshman, K., Kaluri, R., Rajput, D., Srivastava, G., Baker, T.: Analysis of Dimensionality Reduction Techniques on Big Data. IEEE Access **8**, 54776–54788 2020
30. Jolliffe, I., Cadima, J.: Principal component analysis: A review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **374** (2016)
31. Kim, C., Klabjan, D.: A Simple and Fast Algorithm for L1-norm Kernel PCA. IEEE Transactions on Pattern Analysis and Machine Intelligence **42**(8), 1842 - 1855 (2019)
32. Van der Maaten, L., Postma, E., Herik, H.: Dimensionality Reduction: A Comparative Review. Journal of Machine Learning Research **10** (2009)
33. Yang, B., Xiang, M., Zhang, Y.: Multi-manifold Discriminant Isomap for visualization and classification. Pattern Recognition **55** (2016)
34. McInnes, L., Healy, J.: UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. ArXiv preprint, arXiv:1802.03426 (2018)

35. Tschannen, M., Bachem, O., Lucic, M.: Recent Advances in Autoencoder-Based Representation Learning. ArXiv preprint, arXiv:1812.05069 (2018)
36. Xu, D., Tian, Y.: A Comprehensive Survey of Clustering Algorithms. Annals of Data Science **2** (2015)
37. Wong, K.-C.: A Short Survey on Data Clustering Algorithms. 2015 Second International Conference on Soft Computing and Machine Intelligence (ISCMI), pp. 64-68. IEEE (2015)
38. Singh, S., Srivastava, S.: Review of Clustering Techniques in Control System. Procedia Computer Science **173**, 272–280 (2020)
39. Jason X., K. L.: Power k-Means Clustering. ICML, 2019.
40. Lorbeer, B., Kosareva, A., Deva, B., Softić, D., Ruppel, P., Küpper, A.: Variations on the Clustering Algorithm BIRCH. Big Data Research **11** (2017)
41. Gosain, A., Dahiya, S.: Performance Analysis of Various Fuzzy Clustering Algorithms: A Review. Procedia Computer Science **79**, 100-111. (2016)
42. Yongli, L., Hengda W., Tianyi D., Jingli C., Hao, C.: Incremental fuzzy clustering based on a fuzzy scatter matrix. J. Inf. Process. Syst. **15**, 359-373 (2019)
43. Viroli, C., Mclachlan, G.: Deep Gaussian Mixture Models. Statistics and Computing **29** (2019)
44. Chen, Y., Tang, S., Bouguila, N., Wang, C., Du, J., Li, H.: A Fast Clustering Algorithm based on pruning unnecessary distance computations in DBSCAN for High-Dimensional Data. Pattern Recognition **83** (2018)
45. Schubert, E., Gertz, M.: Improving the Cluster Structure Extracted from OPTICS Plots. LWDA (2018)
46. Pulkit, M., Abhishek, M., Saket, A., Sushil, M.: Deep mean shift clustering. 2015 Second International Conference on Soft Computing and Machine Intelligence (ISCMI), pp. 64-68. IEEE (2019)
47. Suman, S., Rani, P.: A Survey on STING and CLIQUE Grid Based Clustering Methods. International Journal of Advanced Research in Computer Science **8**, 1510-1512 (2017)
48. Cer, D., Yang, Y., Kong, S.-Y. Hua, N., Limtiaco, N., John, R., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., Kurzweil, R.: Universal Sentence Encoder. ArXiv preprint, arXiv:1803.11175 (2018)
49. Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., Fidler, S.: Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. 2015 IEEE International Conference on Computer Vision (ICCV), 19-27 (2015)
50. Sennrich, R., Haddow, B., Birch, A.: Neural Machine Translation of Rare Words with Subword Units. ArXiv preprint, arXiv:1508.07909 (2016)
51. Welcome to Python.org. https://www.python.org/. Last accessed 14 Sept 2021.
52. TensorFlow. https://www.tensorflow.org/. Last accessed 14 Sept 2021.
53. TensorFlow Hub. https://www.tensorflow.org/hub. Last accessed 14 Sept 2021.
54. TensorFlow Hub. https://tfhub.dev/tensorflow/bert_multi_cased_L-12_H-768_A-12/3. Last accessed 14 Sept 2021.
55. TensorFlow Hub. https://tfhub.dev/tensorflow/bert_multi_cased_preprocess/1. Last accessed 14 Sept 2021.
56. GitHub - marcopoli/AlBERTo-it: AlBERTo the first italian BERT model forTwitter languange understanding. https://github.com/marcopoli/AlBERTo-it. Last accessed 14 Sept 2021.
57. TensorFlow Hub. https://tfhub.dev/google/universal-sentence-encoder-multilingual/3. Last accessed 14 Sept 2021.

58. Ross, D.,Lim, J., Lin, R.-S., Yang, M.-H.: Incremental Learning for Robust Visual Tracking. International Journal of Computer Vision **77**, 125-141 (2008)
59. Md Ashad, A., Kenji, F.: Hyperparameter selection in kernel principal component analysis. Journal of Computer Science **10**, 1139-1150 (2014)
60. Satopaa, V., Albrecht, J., Irwin, D., Raghavan, B.: Finding a Kneedle in a Haystack: Detecting Knee Points in System Behavior. 2011 31st International Conference on Distributed Computing Systems Workshops, 166–171 (2011)
61. Zelnik-Manor, L., Perona, P.: Self-Tuning Spectral Clustering. Adv. Neural Inf. Process. Syst. **17** (2004)
62. García-Pedrajas, N., Romero, J., Cerruela, G.: A Proposal for Local k Values for k-Nearest Neighbor Rule. IEEE Transactions on Neural Networks and Learning Systems **28**, 1–6 (2015)
63. Wang, F., Franco-Penya, H.-H., Kelleher, J., Pugh, J., Ross, R.: An Analysis of the Application of Simplified Silhouette to the Evaluation of k-means Clustering Validity. MLDM (2017)
64. Compute Engine documentation — Compute Engine Documentation. https://cloud.google.com/compute/docs. Last accessed 14 Sept 2021.
65. Feine, J., Morana, S., Maedche, A.: Designing a Chatbot Social Cue Configuration System. In: International Conference on Information Systems. Munich (2019)