An Evaluative Measure of Clustering Methods Incorporating Hyperparameter Sensitivity

Siddhartha Mishra[1], **Nicholas Monath**[†,1], **Michael Boratko**[1], **Ari Kobren**[2], **Andrew McCallum**[1]
[1]College of Information and Computer Sciences, University of Massachusetts Amherst
{siddharthami,nmonath,mboratko,mccallum}@cs.umass.edu
[2]Oracle Labs
ari.kobren@oracle.com

Abstract

Clustering algorithms are often evaluated using metrics which compare with ground-truth cluster assignments, such as Rand index and NMI. Algorithm performance may vary widely for different hyperparameters, however, and thus model selection based on optimal performance for these metrics is discordant with how these algorithms are applied in practice, where labels are unavailable and tuning is often more art than science. It is therefore desirable to compare clustering algorithms not only on their optimally tuned performance, but also some notion of how realistic it would be to obtain this performance in practice. We propose an evaluation of clustering methods capturing this ease-of-tuning by modeling the expected best clustering score under a given computation budget. To encourage the adoption of the proposed metric alongside classic clustering evaluations, we provide an extensible benchmarking framework. We perform an extensive empirical evaluation of our proposed metric on popular clustering algorithms over a large collection of datasets from different domains, and observe that our new metric leads to several noteworthy observations.

1 Introduction

Whether the application is in biology (Stassen et al. 2020) or personalization (Zaheer et al. 2019), jet physics (Greenberg et al. 2021) or automatic knowledge-base completion (Vashishth, Jain, and Talukdar 2018), clustering is a widely used tool. It is used for data analysis (Dubes and Jain 1980), visualization (Rasmussen and Karypis 2004), and to directly solve tasks such as entity resolution (Steorts, Hall, and Fienberg 2016), community detection (Riedy et al. 2011) and image segmentation (Chuang et al. 2006). In many such settings, there is not an explicit ground-truth clustering, and practioners may try a variety of different clustering algorithms and hyperparameter settings for those algorithms until they are able to find meaningful patterns in their data. Developing evaluations of models in alignment with their real-world performance can be challenging. This is particularly difficult for clustering, where the fundamental goal is to discover the underlying clusters and, thus, having access to such cluster assignments for purposes of evaluation renders the entire enterprise irrelevant.

[†] Now at Google.

When new clustering algorithms are introduced, evaluation is often done with respect to clustering benchmark datasets with ground truth labeled clusters (Campello, Moulavi, and Sander 2013; Bateni et al. 2017; Kobren et al. 2017, inter alia). The quality of the predicted clustering is measured in comparison to the ground truth using metrics like adjusted Rand index (Rand 1971) or normalized mutual information (Lancichinetti, Fortunato, and Kertész 2009). While these metrics can effectively describe the accuracy and representational capacity of clustering methods, the fact that there are other criteria on which a clustering method can be evaluated. While these metrics are quite important, they only give a limited sense of how practical a given algorithm is in real-world settings.

In the aforementioned applications, a practitioner needs to experiment with different settings of an algorithm. Ideally, an algorithm would give high-quality results without much work, tuning of hyperparameters would be straightforward or even automated so that the algorithm is usable by someone who does not deeply understand the algorithm's details or even clustering in detail. However, to the best of our knowledge, no such metric for clustering incorporates this aspect of *usability*.

In this paper, we evaluate the extent to which clustering methods empirically benefit from extensive hyperparameter tuning. To this end, we present a metric that captures the usability of a given algorithm. We model the selection of hyperparameters as a stochastic process, and consider the expected value of a clustering objective (e.g., adjusted Rand index) for the *best* clustering produced after t steps of hyperparameter selection. We consider both an *uninformed* setting, where the practitioner selects hyperparameters uniformly randomly from a reasonable range, as well as *informed*, where the stochastic process is guided by Bayesian optimization as a proxy for how hyperparameters might be tuned by researchers when reporting optimal performance. In both settings, the extent to which this metric changes as t increases is a good indication of the *ease-of-tuning*.

In order to evaluate the quality of our metric, we compare 5 commonly used clustering algorithms: mini-batch k-means, BIRCH, DBSCAN, HDBSCAN, and HAC. We conduct extensive experiments on over 50 datasets, and find that

- The performance of algorithms often changes significantly with more substantial tuning.

- The ranking of algorithms can change, depending on whether Bayesian hyperameter tuning was used.
- Most algorithms have a single hyperparameter which is predominantly responsible for this variation in performance, which we identify.

Evaluating on such a large scale establishes that this metric is reliably informative across a wide range of datasets. In order to make this level of rigorous analysis accessible to researchers proposing novel clustering methods, we designed and provide an extensible evaluation framework and a curated selection of datasets on which to evaluate.

2 Background

Given a dataset $\boldsymbol{X}$ of n points, clustering is the task of partitioning $\boldsymbol{X}$ into disjoint subsets $\mathcal{C}$ which cover $\boldsymbol{X}$, i.e.

$$\forall C_i, C_j \in \mathcal{C}, C_i \cap C_j = \emptyset \quad \text{and} \quad \bigcup_{C \in \mathcal{C}} C = \boldsymbol{X}.$$

The partition $\mathcal{C}$ is also called a **clustering**. A **clustering method**, $\mathcal{A}$, takes as input a dataset $\boldsymbol{X}$ as well as **method specific hyperparameters** ϕ and outputs a partition $\mathcal{C}$. For example, the popular clustering method mini-batch k-means (Sculley 2010) has hyperparameters such as the batch size and convergence tolerance.

2.1 Evaluation Metrics used in Clustering

An **evaluation metric**, $\mathcal{M}(\dots)$, measures the quality of a predicted clustering $\mathcal{C}$ either in terms of a target partition, $\mathcal{C}^\star$, or properties of the dataset $\boldsymbol{X}$. Metrics defined against a target partition $\mathcal{M}(\mathcal{C}, \mathcal{C}^\star)$ include Rand index (Rand 1971), pairwise F1 (Barnes 2015), and normalized mutual information (Lancichinetti, Fortunato, and Kertész 2009), among others. Such metrics are known as *external* metrics, as they make use of external information (in this case, cluster labels). This is in contrast with *internal* metrics, such as k-means cost, DP-means (Jiang, Kulis, and Jordan 2012), or correlation clustering cost (Ailon, Charikar, and Newman 2008), which are defined on a clustering and the dataset, i.e. $\mathcal{M}(\mathcal{C}, \boldsymbol{X})$.

Internal evaluations provide some measure of comparison between clustering algorithms, however they implicitly make assumptions about the structure of the data to be clustered and can be biased towards algorithms which make a similar assumption (Estivill-Castro 2002). External metrics are often preferred to internal, as they can be more representative of actual task performance, particularly for use cases such as information retrieval (Schütze, Manning, and Raghavan 2008). However, since external metrics require ground truth labels, they allow methods to be excessively tuned in ways that are not possible in practice.

Our work will measure the performance of a clustering algorithm, $\mathcal{A}$, with respect to a metric $\mathcal{M}(\dots)$ taking difficulty of tuning into account. In our empirical analysis (Section 4), we present results where the metric $\mathcal{M}(\dots)$ is adjusted Rand index, however our approach is general enough to support any metric (external or internal).

2.2 Hyperparameters in Clustering Methods

Hyperparameters of clustering methods, ϕ, are those inputs to an algorithm, $\mathcal{A}$, that define how a method is performed. It is distinguished from a **model parameter** in that the latter is learned from the data, whereas hyperparameters are often selected by the practitioner. For example, the cluster centroids of a method like k-means would be considered model parameters, whereas the learning rate would be a hyperparameter. As we show in Section 4, the performance of clustering methods can be greatly impacted by the specific choice of hyperparameters.

2.3 Bayesian Optimization

Bayesian optimization is a method for finding the global optimum of noisy black-box functions, and is particularly applicable to situations where evaluating the function at many points is relatively expensive. The fundamental idea of Bayesian optimization is to model the objective function $f(x)$ (for which the form may be unknown) as a random function using a prior with properties amenable to optimization. Most often, this prior is a Gaussian process, which can be fit to a given set of observations $\mathbf{O}_k = \{(x_1, f(x_1)), \dots, (x_k, f(x_k))\}$, and for which the posterior enables efficient informed sampling of potential points to evaluate according to a specified **acquisition function**, most commonly **expected improvement**,

$$\mathrm{EI}(x) = \mathbb{E}[\max(f(x) - f(x^+), 0)],$$

where x^+ is such that $f(x^+)$ is the largest value observed thus far. Bayesian optimization can be adapted to handle ordinal variables (by rounding) and categorical variables (by one-hot encoding). In our setting, we use Bayesian optimization for hyperparameter tuning, maximizing the function $f(\phi) = \mathcal{M}(\mathcal{A}(\boldsymbol{X}; \phi), \mathcal{C}^*)$.

2.4 Functional ANOVA

While Bayesian optimization can effectively be used as a black-box optimization procedure, this is not entirely satisfying, as it leaves one without any understanding about how the hyperparameters impact the performance. Often, it is the case that a few hyperparameters are much more important than others (Bergstra and Bengio 2012), for example, and it would be beneficial for practitioners to be aware of this. Functional analysis of variance (fANOVA) decomposes the variance $\mathbb{V}$ of a function f into additive components $\mathbb{V}_U$ for each subset U of it's inputs (Sobol 1993) . It is not possible to perform this decomposition analytically for an arbitrary function f, however Hutter, Hoos, and Leyton-Brown (2014) demonstrate that such a decomposition of the variance is possible if the function is modeled by a random forest. We use their approach to analyze the relative importance of hyperparameters for each clustering method (see Table 2).

3 Proposed Evaluation

In this section we explicitly define our proposed method of evaluating clustering algorithms in accordance with the way they would be used in practice. We define a class of metrics for this evaluation, explain the motivation for their definitions, and the technical details involved in implementing them. First, however, we motivate the need for yet another metric for comparing clustering algorithms.

Clustering algorithms are typically compared using their optimal performance when their hyperparameters are tuned on an external metric. This may yield significantly different results than that which would be achievable in practice without access to ground-truth labels (see Section 4). Ideally, therefore, it would be beneficial to have an evaluation metric which allows for the use of ground-truth labels, so as to avoid the issues related to internal metrics, while also penalizing excessive tuning using these ground-truth labels, since this would not be feasible for practitioners seeking to apply these methods in real-world applications. Available computational resources also play a role, as infinite compute would allow one to simply try all possible settings of hyperparameters, and thus it would be of further benefit if this metric provided some notion of the extent of tuning required. With these guiding principles in mind, we now turn to the definition of our proposed metric.

The first thing one may reasonably consider when presented with the objective of evaluating the quality of algorithms irrespective of their hyperparameter settings is simply to take their expected performance, i.e. $\mathbb{E}_\Phi[\mathcal{M}(\mathcal{A}(\boldsymbol{X};\Phi),\mathcal{C}^*)]$, under some reasonable prior on Φ, and indeed without some selection criteria or even heuristic for comparing two proposed clusterings this is the best estimate one could hope for.

Practitioners are often able to assess the relative quality of two different proposed clusterings, however, by manually inspecting their output, and thus the quantity of interest is actually the maximum performance one can typically achieve with a fixed amount of effort. Thus, given a compute budget of t, we can model the selection of t settings of hyperparameters as a stochastic process $\{\Phi_i\}_{i\in\{1,\dots,t\}}$, and consider the expectation of maximum performance, i.e.

$$\mathrm{EoM}(t) = \mathbb{E}_{\{\Phi_i\}}\left[\max_{i\in\{1,\dots,t\}} \mathcal{M}(\mathcal{A}(\boldsymbol{X},\Phi_i),\mathcal{C}^*)\right]. \quad (1)$$

Since we are taking the max over a larger and larger set of random variables, EoM is monotonically increasing with respect to t, regardless of the exact nature of the stochastic process $\{\Phi_i\}$.

3.1 Random Search

In the simplest case, a practitioner may simply pick hyperparameters uniformly randomly from an acceptable range, in which case the $\{\Phi_i\}$ are i.i.d. random variables. We denote this setting as $\mathrm{EoM_R}$, and note that $\mathrm{EoM_R}$ increases from average to maximum achievable performance, i.e. [1]

$$\mathrm{EoM_R}(1) = \mathbb{E}_\Phi[\mathcal{M}(\mathcal{A}(\boldsymbol{X};\Phi),\mathcal{C}^*)],$$
$$\lim_{t\to\infty} \mathrm{EoM_R}(t) = \operatorname*{ess\,sup}_{\phi} \mathcal{M}(\mathcal{A}(\boldsymbol{X};\phi),\mathcal{C}^*)$$

In practice, this expectation can be estimated in via Monte Carlo sampling. Since we would like to estimate $\mathrm{EoM_R}(t)$ for different values of t, however, we simply sample a large number of Φ, and then repeatedly subsample t evaluations, averaging the max of these t values over T trials.

[1]ess sup is the essential supremum, i.e. the least upper bound ignoring sets of measure zero.

This approach can handle both discrete and continuous hyperparameters, however in the event that a method has fewer hyperparameter settings than t (eg. HAC, which has 8 values for the hyperparameter) it is no longer a reasonable assumption that the practitioner would continually evaluate settings of hyperparameters they have already tried. In this setting, we consider the $\{\Phi_i\}$ to not be i.i.d. but rather sampled from the set of valid hyperparameters *without* replacement.[2]

Taking the expectation of maximum performance with respect to an external evaluation avoids the issues present when relying on internal evaluations, while also preventing any excessive hyperparameter tuning based on this external evaluation, and thus is more in alignment with how models may perform in practice. Inspecting the output of $\mathrm{EoM_R}$ for different values of t also provides a characterization of the extent to which additional computational resources would provide performance benefits.

3.2 Bayesian Optimization Search

On the opposite end of the spectrum, we could consider the selection of hyperparameters by a researcher who has a much deeper understanding of how they impact the model and also the capability to hand-tune such hyperparameters on the external evaluation. This setting more accurately captures the optimal performance which is conventionally reported.

To this end, we tune the models using Bayesian hyperparameter optimization, as described in Section 2.3. We seed the Gaussian process using 10 random evaluations. While imperfect, this prior can be viewed as incorporating a researcher's greater understanding of the model's hyperparameters and characteristics of the dataset. We then sample a hyperparameter $\Phi_1 = \phi_1$ via the expected improvement acquisition function. We re-fit the Gaussian process, including the new observation $(\phi_1, \mathcal{A}(\boldsymbol{X},\phi_1))$, and sample again. This defines a stochastic process $\{\Phi_i\}$ which is no longer comprised of i.i.d. random variables. We denote the expectation of maximum with respect to this stochastic process $\mathrm{EoM_B}$.

While intended as a proxy for comparison with the way in which researchers would typically report algorithm performance, we note that considering the value of this metric for different values of t also provides a useful measure of how easy various algorithms would be to tune using a heuristic which is reasonably correlated with $\mathcal{M}$, eg. calculating $\mathcal{M}$ on some hand-labeled subset of the data.

4 Experiments

The purpose of this experiment section is to demonstrate how our proposed metric can be used to evaluate clustering algorithms. We show that it can be used to deduce interesting properties of the hyperparameters of clustering algorithms as well as directly compare the algorithms themselves. We also

[2]Practitioners would also obviously avoid evaluating the exact same values of hyperparameters in general, however as long as there is at least one continuous random variable (the case for all methods other than HAC) we have for any $i \neq j$, $P(\Phi_i = \Phi_j) = 0$.

perform fANOVA analysis to assess importance of hyperparameters for natural language datasets. We begin by providing a description of our provided benchmarking framework [3]. We then show its application to evaluating well known clustering algorithms under our proposed metric.

We provide an extensible framework to evaluate our proposed metrics on several clustering methods over datasets from different domains.

4.1 Datasets

We use classification datasets in our experiments since our proposed metric requires the use of external clustering measures. We have selected datasets in each domain with a diverse number of clusters, features and instances. We zero-norm the features and perform Z-score normalization (standardization). We find that unit normed features outperform non-unit normed features, and thus preprocess all datasets in this manner.

Generic We use a subset of 56 datasets suitable for clustering from OpenML CC-18 (Bischl et al. 2019; Vanschoren et al. 2013), a benchmark of classification datasets. We use the python library (Feurer et al. 2019) (as recommended) to load the datasets. Each dataset has predefined features as well as classification labels. We use the class labels as the cluster labels for the computation of our metric. Examples of the datasets included in this subset are wdbc (Street, Wolberg, and Mangasarian 1993), a dataset for Breast Cancer diagnosis and first-order-theorem-proving(Bridge, Holden, and Paulson 2014), a dataset for predicting which five heuristics give the fastest proof for a theorem when used by a first-order prover.

Natural Language Processing We select three popular NLP document text classification datasets, AGNews (Zhang, Zhao, and LeCun 2016), DBpedia (Auer et al. 2007), and YahooAnswers (Dror et al. 2011). We use SentenceBERT (Reimers and Gurevych 2019) to get sentence-level embeddings. We consider three different pretrained encoder models to represent the documents: Average of Glove embeddings (Pennington, Socher, and Manning 2014), DistilRoBERTa - which is a distilled version of RoBERTa (Liu et al. 2019) and MPNet (Song et al. 2020).

4.2 Clustering Methods

Recall that the purpose of this experiment is to demonstrate the value of our proposed metric for evaluating clustering algorithms. With this goal in mind, we select six classic and frequently used clustering algorithms. These are algorithms that many practitioners would select and that would likely be familiar to readers, who could match their own understanding of these methods to the conclusions drawn with our metric. Our hope is that analysis similar to the analysis done here would be adopted by researchers proposing new clustering methods. We experiment with the following algorithms:

- **Mini-batch K-Means** (**MBKM**) (Sculley 2010) A scalable optimization of the k-means objective using a mini-batch alternative to Lloyd's algorithm (Lloyd 1982). Hyperparameters include the mini-batch size, the number of iterations, the tolerance to stopping condition, number of random k-means++ initializations (Arthur and Vassilvitskii 2006), amount of dataset to use when initializing, center re-assignment criteria, and stopping criteria.
- **DBSCAN** (Ester et al. 1996). A seminal density-based method which has two hyperparameters: a radius, epsilon, and a min number of points criteria. Clusters are built by first labeling points as "core points" if they have more than the minimum number of points within epsilon. Edges are then added between core points within epsilon of one another, and all points within epsilon of core points which are connected via these edges are considered to be in the same cluster.
- **HDBSCAN** (Campello, Moulavi, and Sander 2013). A hierarchical density-based algorithm that transforms point-wise distances based on core point designations in a way similar to robust single linkage (Chaudhuri and Dasgupta 2010), builds a single-linkage style clustering and condenses based on a density based criteria.
- **BIRCH** (Zhang, Ramakrishnan, and Livny 1996) works by first building a cluster tree of parametric branching factor in an incremental manner using a threshold to determine when two points can sit at the same tree node and when the tree needs to grow to separate the points as siblings. After the tree is built, a 'rebuilding' phase re-clusters into a set of flat predicted clusters.
- **HAC** (Sneath, Sokal et al. 1973; Murtagh 1983, inter alia). The classic bottom-up greedy algorithm which sequentially merges the nearest two clusters according to the specific *linkage function*, the lone hyperparameter for this method.

We summarize the algorithms and hyperparameters, including the corresponding bounded search spaces used in our experiments, and descriptions in Table 1. From SciKit-Learn (Pedregosa et al. 2011), we use **MBKM**, **BIRCH**, **DBSCAN**. From scipy (Virtanen and Gommers 2020), we use **HAC**. We use the **HDBSCAN** implementation[4].

4.3 Framework Details

We use Ax (Bakshy et al. 2018) for experiment management. The metadata for the experiments such as optimization parameters, list of datasets, models and range of hyperparameters can be configured using a YAML configuration file based on Hydra (Yadan 2019) , which provides an easy to use CLI. The framework can easily be extended to include new models, it allows the definition of custom evaluation metrics as well as the utilities to plot the visualizations shown in the paper. For distributed hyperparameter optimization, we use Ray Tune (Liaw et al. 2018). The framework supports any custom BOTORCH (Balandat et al. 2020) based model, which is a PyTorch (Paszke et al. 2019) based library for Bayesian Optimization.

4.4 Results

We provide the following results for the experiments we performed:

[3]https://github.com/mishra-sid/clustering_hyperparameters

[4]https://github.com/scikit-learn-contrib/hdbscan

Algorithm	Hyperparameter	Range/Choices	Description
MBKM	`batchSize`	$\{1, \ldots, 1024\}$	Batch size.
	`maxIter`	$\{1, \ldots, 1000\}$	Max number of iterations.
	`tol`	$[0.0, 1.0]$	Tolerance
	`nInit`	$\{1, \ldots, 100\}$	Number of initializations
	`initSize`	{num_clusters, …, num_instances}	Size of initialization
	`reassign`	$[0.0, 1.0]$	Number of re-assignments
	`maxNoImprv`	$\{2, \ldots, 100\}$	Number of iters allowed without improvement
HAC	`linkFunc`	['single', 'complete', 'average', 'centroid', 'median', 'ward', 'weighted']	Linkage function
BIRCH	`threshold`	$[0.001, 0.999]$	Threshold
	`branch`	$\{5, \ldots, 100\}$	Branching factor.
DBSCAN	`eps`	$[0.001, 0.999]$	Max distance for neighboring pts.
	`minPts`	$\{2, \ldots, 100\}$	Min. neighbors for a core point.
HDBSCAN	`eps`	$[0.001, 0.999]$	Max distance for neighboring pts.
	`minPts`	$\{2, \ldots, 100\}$	Min. points in a cluster.
	`minClusterSize`	$\{2, \ldots, 100\}$	Min. cluster size.

Table 1: Algorithms considered for experiments, their hyperparameters, and the range of values searched over

In Figure 1, we plot our proposed metrics EoM_R and EoM_B aggregated over all generic / natural language datasets.

In Figure 2, we plot specific examples of a few specific datasets each in the Generic and Natural Language domains to demonstrate certain properties of the clustering algorithms observed while tuning them.

In Table 2, we tabulate the hyperparameter importances using fANOVA analysis on evaluations for all algorithms mentioned in Section 4. (Note that HAC only has a single hyperparameter, and thus was omitted from this analysis.)

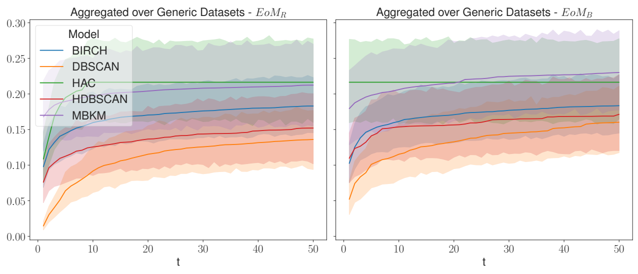
4.5 Discussion

Ranking of Metrics In Figure 1(a) for generic datasets, we observe that the relative performance of the algorithms changes. HDBSCAN, in particular, seems to benefit from hyperparameter tuning, and with sufficient tuning MBKM actually outperforms HAC. The aggregate results for natural language datasets depicted in Figure 1(b) do not show as significant a shift between EoM_R and EoM_B, however we do observe some differences depending on the text encoders in the next paragraph. We also note that the ranking of methods may change significantly between datasets which are of different levels of clustering difficulty. In Figure 2(a), we see that most algorithms are able to do quite well on this dataset, and the ranking is predominantly in agreement with our aggregate results. In Figure 2(b), however, all models yield $\text{EoM}(t) < 0.1$, and we see the relative performance differs drastically with, somewhat surprisingly, birch performing best.

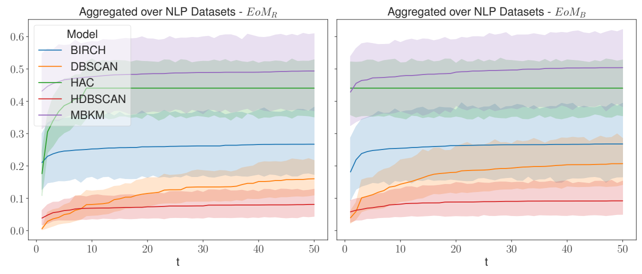
Effect of Encoders in NLP Datasets In Figure 2 (c) and (d), we observe that the performance of clustering algorithms on natural language datasets is dependent on the encoding model used. In particular, GloVe embeddings are easier to cluster than using a BERT based models as the pre-trained encoder. We believe this is because GloVe provides a better sentence-level representation more appropriate to be used by distance functions used to capture similarity between data points in clustering methods. Particularly in the case of DBSCAN, which uses a cosine distance function, performance is improved significantly after selecting effective hyperparameters in both uninformed and informed cases.

Density-based Methods In Figure 1 (a) and (b), we observe that **HDBSCAN** performs better than **DBSCAN** on generic datasets, whereas in case of natrual language datasets **DBSCAN** performs better. We believe this is because the advantage of **DBSCAN** supporting cosine distance, which is better than Euclidean distance for distance measures between word embeddings. This fundamental difference seems to outweigh the algorithmic improvements offered in **HDBSCAN** without the support for cosine distance measure. Among all algorithms, **DBSCAN** benefits the most from having access to labels, i.e. tuned by an expert practitioner.

fANOVA Analysis In the analysis summarized in Table 2, we observe that algorithms like **DBSCAN** and **BIRCH** have their `eps` and `threshold` as the hyperparameter of highest importance across all datasets respectively. Both of these hyperparameters are used as a criteria for providing an upper-bound of distance for datapoints to be considered part of or merged into a single cluster. For **MBKM**, the combination of `nInit` and `initSize` has the highest importance, which implies that the performance of the resulting clusters are relatively more dependent on initialization. It is crucial to tune parameters such as these since it causes a large variance in performance across evaluations. In the case of **HDBSCAN**, there is no fixed observable pattern of relative importance of hyperparameters across datasets. Thus, **HDBSCAN** would be relatively difficult to tune, as it is important for the practitioner to select effective hyperparameters for all of `eps`, `minClusterSize` and `minPts`.

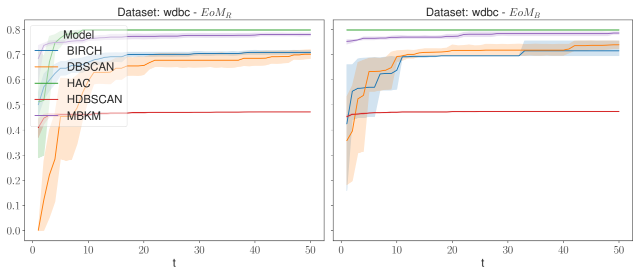


(a) Aggregated over Generic Datasets

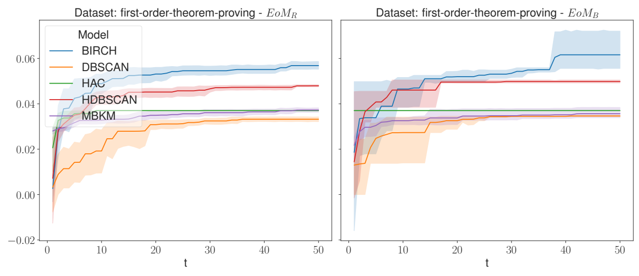


(b) Aggregated over NLP Datasets

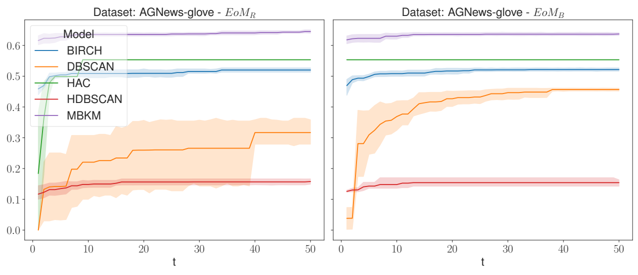
Figure 1: **Evaluation via** EoM. Clustering methods are distinguished by color. 95% confidence intervals are also depicted.



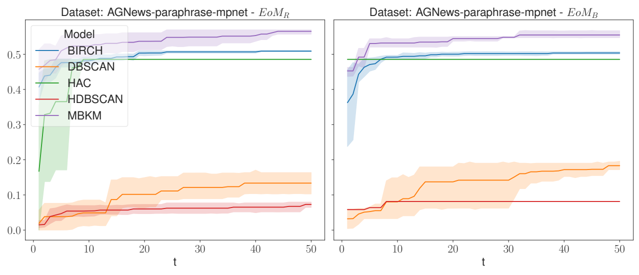
(a) Generic: wdbc Dataset



(b) Generic: First-order-theorem Dataset



(c) NLP : AGNews Dataset with Glove Embeddings



(d) NLP : AGNews Dataset with MPNet Embeddings

Figure 2: **Evaluation via** EoM. Clustering methods are distinguished by color. 95% confidence intervals are also depicted.

Ease of Tuning As mentioned previously, inspecting the rate at which EoM increases for various t provides a reasonable assessment of the overall ease of tuning for a given method. This is most obvious when observing the graph of HAC - since there are only 8 possible choices for the linkage function, we see that HAC rapidly obtains it's optimal performance, at which point the graph plateaus. Note that ease of tuning is distinct from overall performance - in the wdbc dataset presented in Figure 2(a) we see that, while ranking last, HDBSCAN is arguably quite easy to tune, as it achieves this performance by $t = 10$. While one would not want to focus exclusively on ease of tuning, therefore, inspecting the plots of $\mathrm{EoM_R}$ would allow researchers to choose an algorithm which reliably obtains a reasonable level of performance within their budget. Inspecting the plots of $\mathrm{EoM_R}$ would allow more experienced practitioners a sense of how hard various methods would be to tune even with more familiarity with the method and the dataset. Comparing the relative performance between these two metrics may suggest that, for certain datasets and use-cases, it is worth hand-labeling some subset of the data to allow for explicit tuning on a surrogate heuristic metric.

5 Related Work

Benchmarks for Clustering There have been a number of benchmarks proposed for clustering such as for synthetic data for K-means (Fränti and Sieranoja 2018), graph-data (Emmons et al. 2016), univariate data (El Abbassi et al. 2021), and document clustering (Sinka and Corne 2002). Van Mechelen et al. (2018) carefully outlines criteria for building such a benchmark. Beyond clustering there are many efforts for building large collections of datasets in machine learning (Vanschoren et al. 2013; Dua and Graff 2017; Lhoest et al. 2021, inter alia).

Bridging Theory & Practice in Clustering Work has considered ways to understand how theoretical complexity results relate to the kinds of clustering problems faced by prac-

	Dataset	AGNews			DBpedia			YahooAnswers		
Algorithm	**Hyperparam.**	MPNet	RoBERTa	Glove	MPNet	RoBERTa	Glove	MPNet	RoBERTa	Glove
MBKM	`batchSize`	2.226	4.175	0.788	1.832	0.859	0.730	8.598	3.830	1.934
	`initSize`	4.647	8.633	3.931	**17.681**	**26.400**	4.014	2.515	3.403	4.934
	`maxIter`	4.326	1.284	8.645	3.201	3.618	4.043	3.678	2.201	3.201
	`maxNoImprv`	1.533	1.833	3.138	1.535	2.404	2.116	2.428	4.602	5.258
	`nInit`	**21.323**	**10.128**	**26.487**	11.391	11.566	**17.841**	**11.947**	**14.456**	**9.834**
	`reassign`	2.265	5.281	2.295	3.136	3.421	3.576	3.161	2.930	2.156
	`tol`	1.501	5.123	2.733	6.288	1.077	3.040	3.346	3.861	3.080
BIRCH	`branch`	5.721	12.066	5.877	14.576	8.881	22.611	14.131	15.005	12.741
	`threshold`	**76.903**	**53.878**	**65.075**	**43.376**	**53.086**	**47.236**	**59.315**	**57.779**	**46.800**
DBSCAN	`eps`	**37.569**	**37.324**	**50.991**	**58.152**	**78.664**	**65.721**	**53.201**	**53.459**	**63.155**
	`minPts`	6.478	8.604	11.112	10.377	2.717	9.358	5.998	16.365	4.416
HDBSCAN	`eps`	0.263	1.577	9.126	2.520	**50.794**	**67.428**	3.462	2.098	0.482
	`minClusterSize`	4.928	12.548	10.097	18.541	2.263	2.135	13.577	**26.155**	**72.431**
	`minPts`	**79.091**	**51.026**	**52.467**	**32.549**	6.540	4.480	**32.570**	23.911	10.668

Table 2: Importances of hyperparameters (in percent) of Clustering Algorithms in NLP Datasets

tioners. (Ben-David 2015, 2018). Other work has advocated focusing the evaluation of clustering based on the use case along side discussing a plethora of important open-ended questions (Von Luxburg, Williamson, and Guyon 2012).

Hyperparameters in Clustering Tuning hyperparameters in clustering has been considered by (Shalamov et al. 2018; Blumenberg and Ruggles 2020, inter alia) . However these works are not focused on designing a metric which incorporates the difficulty of tuning, as in this work.

Clustering Metrics Many external metrics have been proposed for clustering (Rand 1971; Maulik and Bandyopadhyay 2002; He et al. 2004, inter alia) . Specific metrics for tasks like coreference resolution (Bagga and Baldwin 1998; Recasens and Hovy 2011) have also been proposed. Designing internal metrics (e.g., clustering costs) has a long history and is widely studied. Often such costs are designed to automatically discover the number of clusters (Pelleg, Moore et al. 2000; Jiang, Kulis, and Jordan 2012) or for other kinds of clustering (e.g., hierarchical) (Dasgupta 2016).

6 Conclusion

In this paper we conducted a thorough evaluation of the hyperparameter sensitivity of 5 commonly used clustering methods. In doing so, we also presented a new metric for evaluating clustering methods. Our metric captures the difficulty of finding suitably effective hyperparameters in a given compute budget by measuring the expected best performance achieved after t evaluations. We analyze the proposed metric by using it to investigate the performance of several frequently-used clustering algorithms on a collection of over 50 datasets. Under this evaluation, we discover that the order of the optimal method may change depending on the level of sophistication used for tuning. In particular, with sufficient tuning, MBKM may outperform HAC, which is far easier to tune. Future work could consider how to use the distributions over hyperparameters that are built using this work to suggest hyperparameters for unseen datasets.

7 Acknowledgments

This work supported in part by the Center for Data Science and in part the Center for Intelligent Information Retrieval and in part by the National Science Foundation under Grants No. 1763618, and University of Southern California subcontract no. 123875727 under Office of Naval Research prime contract no. N660011924032. Some of the work reported here was performed using high performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Ailon, N.; Charikar, M.; and Newman, A. 2008. Aggregating inconsistent information: ranking and clustering. *Journal of the ACM (JACM)*, 55(5): 1–27.

Arthur, D.; and Vassilvitskii, S. 2006. k-means++: The advantages of careful seeding. Technical report, Stanford.

Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R.; and Ives, Z. 2007. DBpedia: A Nucleus for a Web of Open Data. In Aberer, K.; Choi, K.-S.; Noy, N.; Allemang, D.; Lee, K.-I.; Nixon, L.; Golbeck, J.; Mika, P.; Maynard, D.; Mizoguchi, R.; Schreiber, G.; and Cudré-Mauroux, P., eds., *The Semantic Web*, 722–735. Berlin, Heidelberg: Springer Berlin Heidelberg.

Bagga, A.; and Baldwin, B. 1998. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, 563–566. Citeseer.

Bakshy, E.; Dworkin, L.; Karrer, B.; Kashin, K.; Letham, B.; Murthy, A.; and Singh, S. 2018. AE: A domain-agnostic platform for adaptive experimentation. In *Conference on Neural Information Processing Systems*, 1–8.

Balandat, M.; Karrer, B.; Jiang, D. R.; Daulton, S.; Letham, B.; Wilson, A. G.; and Bakshy, E. 2020. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*.

Barnes, M. 2015. A Practioner's Guide to Evaluating Entity Resolution Results. *arXiv preprint arXiv:1509.04238*.

Bateni, M. H.; Behnezhad, S.; Derakhshan, M.; Hajiaghayi, M. T.; Kiveris, R.; Lattanzi, S.; and Mirrokni, V. 2017. Affinity clustering: Hierarchical clustering at scale. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6867–6877.

Ben-David, S. 2015. Clustering is easy when.... What? *arXiv preprint arXiv:1510.05336*.

Ben-David, S. 2018. Clustering-what both theoreticians and practitioners are doing wrong. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Bergstra, J.; and Bengio, Y. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2).

Bischl, B.; Casalicchio, G.; Feurer, M.; Hutter, F.; Lang, M.; Mantovani, R. G.; van Rijn, J. N.; and Vanschoren, J. 2019. OpenML Benchmarking Suites. arXiv:1708.03731.

Blumenberg, L.; and Ruggles, K. V. 2020. Hypercluster: a flexible tool for parallelized unsupervised clustering optimization. *BMC bioinformatics*, 21(1): 1–7.

Bridge, J. P.; Holden, S. B.; and Paulson, L. C. 2014. Machine Learning for First-Order Theorem Proving. *Journal of Automated Reasoning*, 53(2): 141–172.

Campello, R. J.; Moulavi, D.; and Sander, J. 2013. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, 160–172. Springer.

Chaudhuri, K.; and Dasgupta, S. 2010. Rates of convergence for the cluster tree. In *NIPS*, 343–351. Citeseer.

Chuang, K.-S.; Tzeng, H.-L.; Chen, S.; Wu, J.; and Chen, T.-J. 2006. Fuzzy c-means clustering with spatial information for image segmentation. *computerized medical imaging and graphics*, 30(1): 9–15.

Dasgupta, S. 2016. A cost function for similarity-based hierarchical clustering. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, 118–127.

Dror, G.; Koren, Y.; Maarek, Y.; and Szpektor, I. 2011. I Want to Answer; Who Has a Question? Yahoo! Answers Recommender System. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, 1109–1117. New York, NY, USA: Association for Computing Machinery. ISBN 9781450308137.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository.

Dubes, R.; and Jain, A. K. 1980. Clustering methodologies in exploratory data analysis. *Advances in computers*, 19: 113–228.

El Abbassi, M.; Overbeck, J.; Braun, O.; Calame, M.; van der Zant, H. S.; and Perrin, M. L. 2021. Benchmark and application of unsupervised classification approaches for univariate data. *Communications Physics*, 4(1): 1–9.

Emmons, S.; Kobourov, S.; Gallant, M.; and Borner, K. 2016. Analysis of Network Clustering Algorithms and Cluster Quality Metrics at Scale. *PLOS ONE*, 11(7): 1–18.

Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X.; et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, 226–231.

Estivill-Castro, V. 2002. Why so many clustering algorithms: a position paper. *ACM SIGKDD explorations newsletter*, 4(1): 65–75.

Feurer, M.; van Rijn, J. N.; Kadra, A.; Gijsbers, P.; Mallik, N.; Ravi, S.; Mueller, A.; Vanschoren, J.; and Hutter, F. 2019. OpenML-Python: an extensible Python API for OpenML. *arXiv*, 1911.02490.

Fränti, P.; and Sieranoja, S. 2018. K-means properties on six clustering benchmark datasets.

Greenberg, C. S.; Macaluso, S.; Monath, N.; Dubey, A.; Flaherty, P.; Zaheer, M.; Ahmed, A.; Cranmer, K.; and McCallum, A. 2021. Exact and Approximate Hierarchical Clustering Using A*. *UAI*.

He, J.; Tan, A.-H.; Tan, C.-L.; and Sung, S.-Y. 2004. On quantitative evaluation of clustering systems. In *Clustering and information retrieval*, 105–133. Springer.

Hutter, F.; Hoos, H.; and Leyton-Brown, K. 2014. An efficient approach for assessing hyperparameter importance. In *International conference on machine learning*, 754–762. PMLR.

Jiang, K.; Kulis, B.; and Jordan, M. 2012. Small-variance asymptotics for exponential family Dirichlet process mixture models. *Advances in Neural Information Processing Systems*, 25: 3158–3166.

Kobren, A.; Monath, N.; Krishnamurthy, A.; and McCallum, A. 2017. A hierarchical algorithm for extreme clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 255–264.

Lancichinetti, A.; Fortunato, S.; and Kertész, J. 2009. Detecting the overlapping and hierarchical community structure in complex networks. *New journal of physics*, 11(3): 033015.

Lhoest, Q.; del Moral, A. V.; von Platen, P.; Wolf, T.; Jernite, Y.; Thakur, A.; Tunstall, L.; Patil, S.; Drame, M.; Chaumond, J.; Plu, J.; Davison, J.; Brandeis, S.; Scao, T. L.; Sanh, V.; Xu, K. C.; Patry, N.; McMillan-Major, A.; Schmid, P.; Gugger, S.; Liu, S.; Lesage, S.; Debut, L.; Matussière, T.; Delangue, C.; and Bekman, S. 2021. huggingface/datasets: 1.11.0.

Liaw, R.; Liang, E.; Nishihara, R.; Moritz, P.; Gonzalez, J. E.; and Stoica, I. 2018. Tune: A Research Platform for Distributed Model Selection and Training. *arXiv preprint arXiv:1807.05118*.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692.

Lloyd, S. 1982. Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2): 129–137.

Maulik, U.; and Bandyopadhyay, S. 2002. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on pattern analysis and machine intelligence*, 24(12): 1650–1654.

Murtagh, F. 1983. A survey of recent advances in hierarchical clustering algorithms. *The computer journal*, 26(4): 354–359.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830.

Pelleg, D.; Moore, A. W.; et al. 2000. X-means: Extending k-means with efficient estimation of the number of clusters. In *Icml*, volume 1, 727–734.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543.

Rand, W. M. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336): 846–850.

Rasmussen, M.; and Karypis, G. 2004. gCLUTO–An Interactive Clustering, Visualization, and Analysis System.

Recasens, M.; and Hovy, E. 2011. BLANC: Implementing the Rand index for coreference evaluation. *Natural Language Engineering*, 17(4): 485–510.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv:1908.10084.

Riedy, E. J.; Meyerhenke, H.; Ediger, D.; and Bader, D. A. 2011. Parallel community detection for massive graphs. In *International Conference on Parallel Processing and Applied Mathematics*, 286–296. Springer.

Schütze, H.; Manning, C. D.; and Raghavan, P. 2008. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

Sculley, D. 2010. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, 1177–1178.

Shalamov, V.; Efimova, V.; Muravyov, S.; and Filchenkov, A. 2018. Reinforcement-based method for simultaneous clustering algorithm selection and its hyperparameters optimization. *Procedia Computer Science*, 136: 144–153.

Sinka, M. P.; and Corne, D. W. 2002. A large benchmark dataset for web document clustering. *Soft computing systems: design, management and applications*, 87: 881–890.

Sneath, P. H.; Sokal, R. R.; et al. 1973. *Numerical taxonomy. The principles and practice of numerical classification.*

Sobol, I. 1993. Sensitivity estimates for nonlinear mathematical models. *Math. Model. Comput. Exp*, 1(4): 407–414.

Song, K.; Tan, X.; Qin, T.; Lu, J.; and Liu, T.-Y. 2020. MPNet: Masked and Permuted Pre-training for Language Understanding. arXiv:2004.09297.

Stassen, S. V.; Siu, D. M.; Lee, K. C.; Ho, J. W.; So, H. K.; and Tsia, K. K. 2020. PARC: ultrafast and accurate clustering of phenotypic data of millions of single cells. *Bioinformatics*, 36(9): 2778–2786.

Steorts, R. C.; Hall, R.; and Fienberg, S. E. 2016. A Bayesian approach to graphical record linkage and deduplication. *Journal of the American Statistical Association*, 111(516): 1660–1672.

Street, W. N.; Wolberg, W. H.; and Mangasarian, O. L. 1993. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical image processing and biomedical visualization*, volume 1905, 861–870. International Society for Optics and Photonics.

Van Mechelen, I.; Boulesteix, A.-L.; Dangl, R.; Dean, N.; Guyon, I.; Hennig, C.; Leisch, F.; and Steinley, D. 2018. Benchmarking in cluster analysis: A white paper. *arXiv preprint arXiv:1809.10496*.

Vanschoren, J.; van Rijn, J. N.; Bischl, B.; and Torgo, L. 2013. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations*, 15(2): 49–60.

Vashishth, S.; Jain, P.; and Talukdar, P. 2018. Cesi: Canonicalizing open knowledge bases using embeddings and side information. In *Proceedings of the 2018 World Wide Web Conference*, 1317–1327.

Virtanen, P.; and Gommers, R. e. a. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272.

Von Luxburg, U.; Williamson, R. C.; and Guyon, I. 2012. Clustering: Science or art? In *Proceedings of ICML workshop on unsupervised and transfer learning*, 65–79. JMLR Workshop and Conference Proceedings.

Yadan, O. 2019. Hydra - A framework for elegantly configuring complex applications. Github.

Zaheer, M.; Ahmed, A.; Wang, Y.; Silva, D.; Najork, M.; Wu, Y.; Sanan, S.; and Chatterjee, S. 2019. Uncovering hidden structure in sequence data via threading recurrent models. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 186–194.

Zhang, T.; Ramakrishnan, R.; and Livny, M. 1996. BIRCH: an efficient data clustering method for very large databases. *ACM sigmod record*, 25(2): 103–114.

Zhang, X.; Zhao, J.; and LeCun, Y. 2016. Character-level Convolutional Networks for Text Classification. arXiv:1509.01626.