# Scalable HDBSCAN Clustering with kNN Graph Approximation

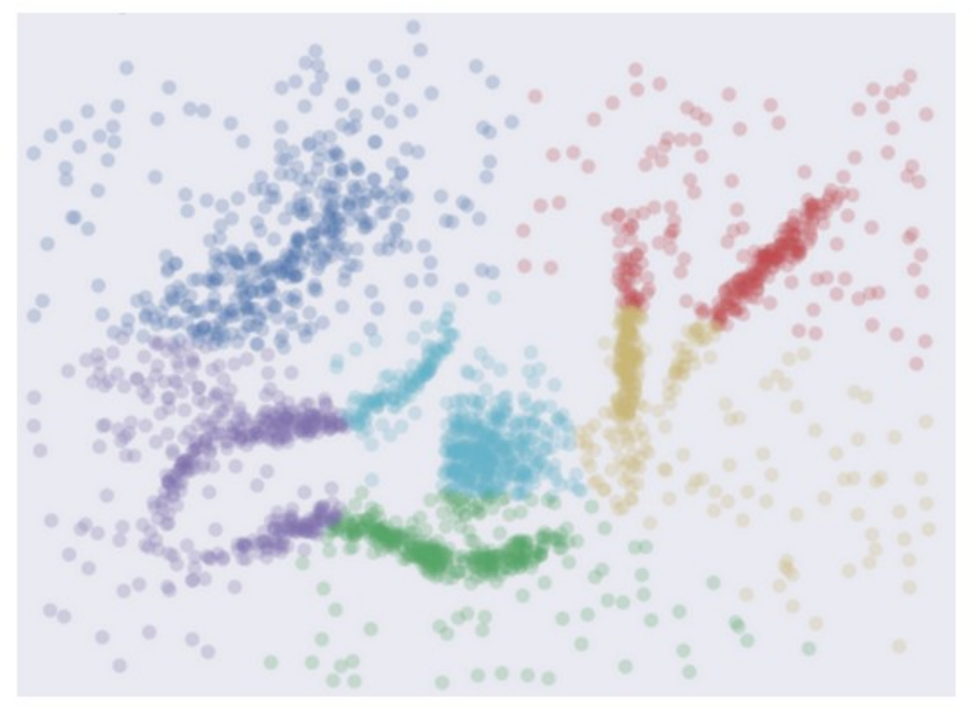
Jacob Jackson, Aurick Qiao, Eric P. Xing

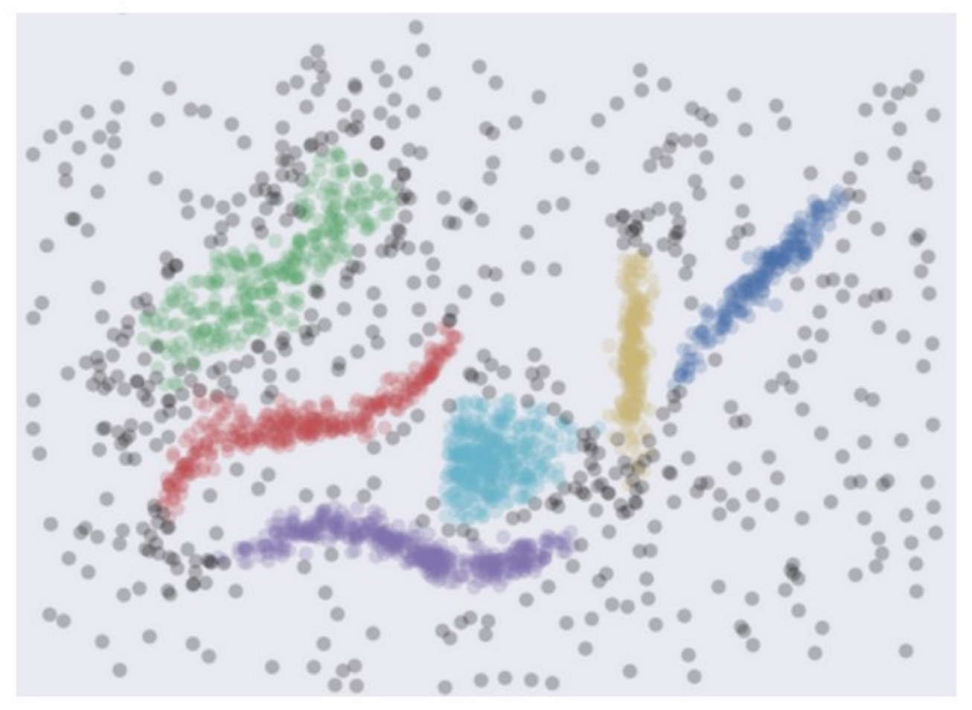
Petuum Inc.

## Motivation

HDBSCAN is a clustering algorithm with some desirable properties:

- Can find clusters of irregular shapes.
- Tolerant of noise.
- Works with any distance metric.


*Example clustering using k-means*


*Example clustering using HDBSCAN*

But it scales **quadratically** with the size of the data, making it impractical to use in many instances.

We use the following ideas to make HDBSCAN fast enough to run on **millions of points**:

- Replace exact kNN graph construction with an approximation using the **NN-Descent** algorithm.
- Use the kNN graph instead of the all-pairs distance graph when extracting the clusters.
- Implement in a distributed framework and optimize the constant factor.

## Project Goals

Create a clustering algorithm which is:

- **Flexible.** HDBSCAN supports any distance metric and can find clusters of irregular shapes.
- **Reliable.** Outliers have limited impact because HDBSCAN is tolerant of noise points.
- **Efficient.** Our implementation runs on a dataset of 400,000 300-dimensional points in 10 minutes when restricted to a single thread.
- **Scalable.** Using 4 machines, each with 4 cores, we can improve the runtime on this dataset by 4x to 2.5 minutes.

## NN-Descent

**Iteratively improve** the approximation of the kNN graph by considering neighbors of neighbors as candidates to replace existing edges.

## Distance metrics

Distance metrics such as **cosine similarity** are essential for meaningfully comparing some kinds of data. HDBSCAN supports these distance metrics.

## Scalability

Our implementation can exploit **multiple cores** and **multiple machines** to obtain substantial performance improvements.

## How It Works

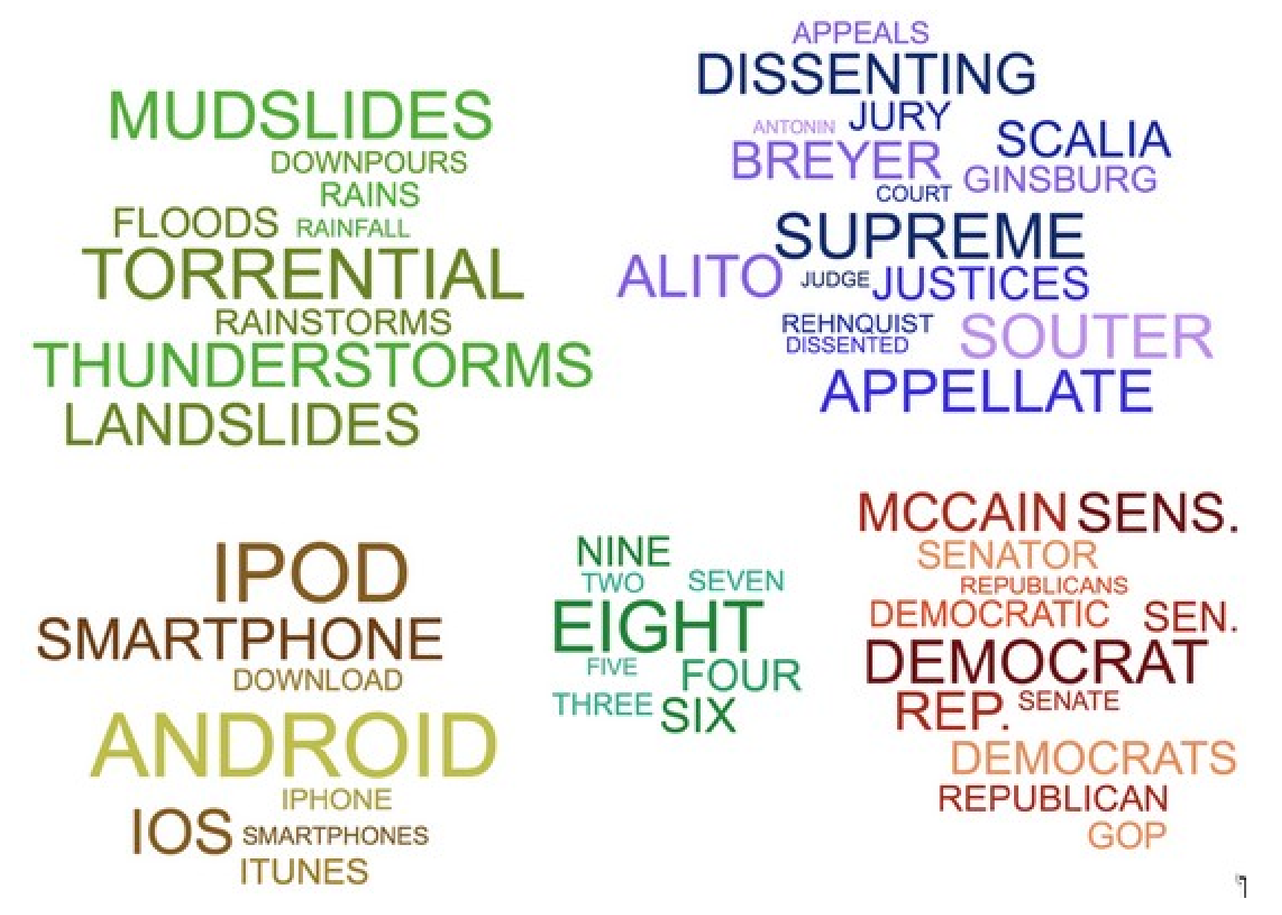
### How HDBSCAN Works

The traditional implementation of HDBSCAN consists of two main phases:

1. Construct the k-nearest-neighbors (k-NN) graph, with an undirected edge connecting each point p to the k most similar points to p. Use the k-NN information to define a new metric called the mutual reachability distance between all pairs of points in the data, which reduces the sensitivity to outliers.
2. Find the minimum spanning tree (MST) connecting all points in the data according to the mutual reachability distance. This tree represents a hierarchy of clusters, and the individual clusters can be extracted using a simple heuristic.

### How NN-Descent Works

We use **NN-Descent** to approximate step 1 of HDBSCAN, where we construct the k-NN graph. It is based on the principle that the neighbor of a neighbor is likely to be a neighbor. Each point p keeps a list of k other points, which are the k closest points to p that have been found so far. In each update step, two randomly chosen a and b in the neighbor list of a point are compared. If b is closer to a than the farthest point in a's neighbor list, then the farthest point in a's neighbor list is replaced by b. Repeating this update improves the accuracy of the k-NN graph approximation with each iteration.

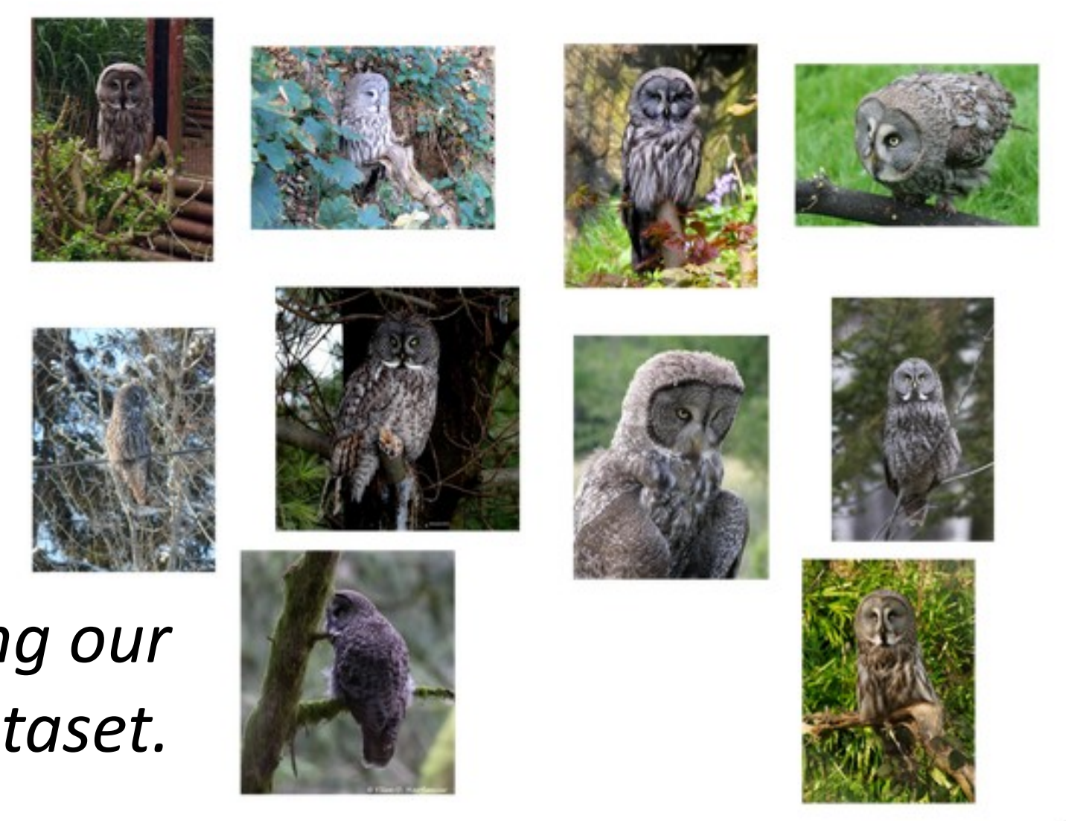


*Examples of clusters obtained by running our implementation on a word embedding dataset.*

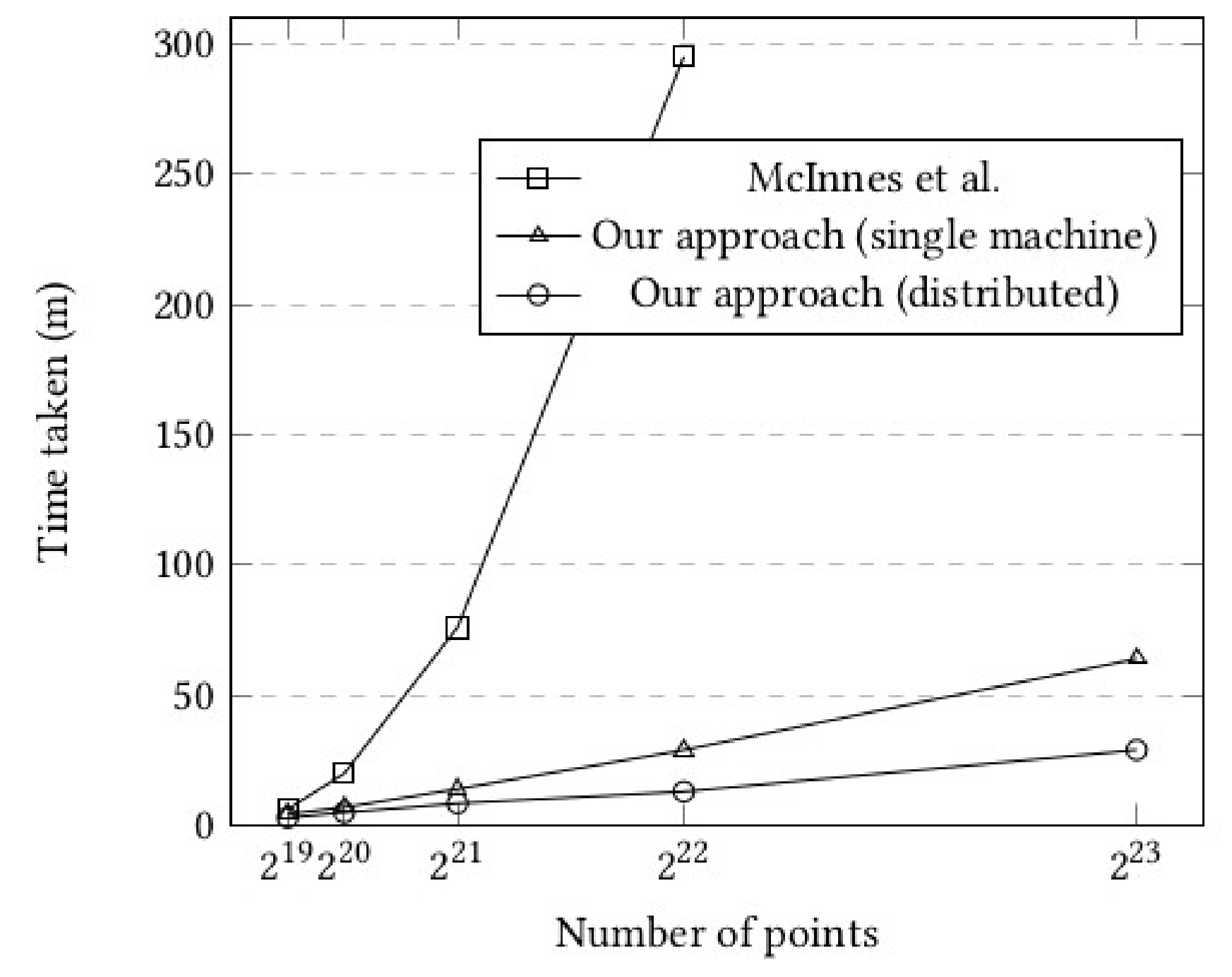
### Our modifications to HDBSCAN

We use NN-Descent to approximate step 1 of HDBSCAN without having to compute the exact k-NN graph, as this would take $O(n^2)$ time.

We also optimize step 2 by assuming that every edge edge in the MST comes from the k-NN graph. Although this assumption is not always true, points which belong in the same cluster typically have nearest neighbors in the cluster, so it is usually true for the edges we care about: that is, the edges connecting two points which belong in the same cluster.


*An example cluster of owls from running our implementation on an image dataset.*

## Performance Results



### Comparison with Scikit HDBSCAN (McInnes et al.)

- Scikit HDBSCAN is exact, while our implementation is approximate.
- The runtime of Scikit HDBSCAN scales quadratically, while ours is empirically $O(n^{1.14})$.

## Accuracy Results

Selected clusters from the result of running on a word embedding dataset containing 400,000 points with dimension 300:

- loaf loaves rigatoni tofu spaghetti orzo tacos snacks fettuccine penne dente linguine couscous sausage pasta cheese bread noodles sandwiches
- infinitive imperfective noun nominative pronoun dative genitive pronouns verb participle verbs subjunctive nouns accusative
- phi kappa epsilon theta sigma

We compared with an exact HDBSCAN implementation on synthetic datasets with 1,000 clusters.

- The similarity between our clustering and the exact clustering was over 0.89 in all instances
- Similarity measured by FM score (1 means identical).

## Conclusion and Future Work

- Using our implementation, it's possible to cluster millions of points in under an hour.
- Ideas for future work:
  - Theoretical justification for why our approach performs well.
  - Investigate alternatives to NN-Descent for kNN approximation.