Stephen Hawking, one of the most brilliant minds of our time, recently suggested that humans will not survive another thousand years without escaping to another earth-*like* planet. NASA in its effort to find extra-terrestrial life looks for planets which are *similar* to earth. It also tries to find stars which are *similar* to our sun. Our universe has billions of galaxies with each containing billions of planets and stars.

Given a huge list of objects, "grouping similar objects together and separating dissimilar objects" is a fundamental problem in science and nature. In biology, you need to find similar proteins. In genetics, you want to find similar DNA sequences. Websites like Facebook and Google want to show similar ads to similar users; Netflix wants to suggest similar movies to similar users. This problem is called **clustering**. In computer science, it arises in different contexts like image analysis, text analysis, speech processing and much more. The task of clustering is challenging due to the following reasons.

1. *Computationally expensive*

NP-Hardness is a popular notion in computer science and mathematics to characterize computationally 'difficult' or expensive problems. Researchers have shown that the clustering problem is NP-Hard. Informally, this means that there exist datasets on which any clustering algorithm will take a very long time to find a solution. Hence heuristics are used in practice. These heuristics have been successful in few applications like cancerous data detection, search engines, etc. However, a lot is still left to be desired given the vast applicability of clustering algorithms.

One of the reasons is that, despite their prevalence, clustering algorithms are very poorly understood! The heuristics used in practice have little or no formal guarantees. When would a particular algorithm succeed, when could it fail? Current literature has little or no answer to these fundamental questions.

How do we tackle computational complexity of clustering?

In preliminary work, my co-authors and I defined a mathematical notion of *clusterable* or *nice* data. We proposed a new clustering algorithm. We then proved that if the input has the nice properties (as defined by our notion), then our proposed algorithm finds the correct solution. Our notion is much less restrictive than previously defined notions. We expect that many datasets in practice (that one might wish to cluster) would satisfy this assumption.

In the proposed work, we would like to find datasets that satisfy our mathematical property. This will confirm that our model and our notion is realistic. We would also like to investigate whether popular clustering algorithms find the correct solution if the data satisfies our clusterability notion. If not then we would like to modify existing algorithms so that they work under our assumptions.

2. *Under-specificity*

Given a dataset, often there are multiple intuitive ways to group the datapoints. Consider the problem of clustering algorithm a dataset of human faces into two groups. Now, one solution could be to have a group of men's faces and another group of women's. Another solution could be to group all the smiling faces together and all the non-smiling or sad faces in a separate group. However, without knowledge of intended application, the algorithm has no way to prefer one choice over the other. This is called under-specificity or lack of complete information.

How do we tackle under-specificity in clustering?

In preliminary work, my co-authors and I addressed this challenge by incorporating domain knowledge into the clustering task. We introduced a new framework of human-supervised clustering. The algorithm works as usual but can ask queries from an expert (if it gets confused). We prove that access to even very few such answers can turn an otherwise computationally expensive problem into a feasible one. Our framework is much more realistic than previous frameworks.

In the proposed work, we would like to extend this framework to handle more realistic situations. We currently assume that the expert is perfect and doesn't make any mistakes. We would like to handle situations when the expert makes a mistake or chooses not to answer a query (because he is confused). We would also like to implement our algorithms and holistically analyze it against massive real world datasets.