Clustering Startups Based on Customer-Value Proposition

Daniel Semeniuta
Stanford University
dsemeniu@stanford.edu

Meeran Ismail
Stanford University
meeran@stanford.edu

***Abstract*— K-means clustering is a popular unsupervised learning technique for grouping closely related data points. This project utilizes K-means to group together startups by text description into related working industry. Additional techniques such as dimensionality reduction via singular-value decomposition are implemented to reduce the space of the feature set and document-term matrix and improve speed and efficiency of the algorithm. This paper also presents a novel metric for measuring the error of K-means clustering in a reduced dimensionality space.**

I. Introduction

Countless new startups are born every single day, and venture capitalists are always on the lookout to find which one will be the next big thing. To do this, they must learn an incredible amount about the startups they want to invest in. One piece of information that is especially valuable to investors is the industry that a startup is in and the industry competition it faces. As such, classifying startups by industry function is an important tool in investing; however, doing this for the many thousands of startups that are formed every day is impossible by hand. We thus want to use machine learning to cluster companies by customer value proposition, given nothing more than short one to two lines describing what the company does.

The input to our work is a file of text descriptions of various startups scraped from Pitchbook and Crunchbase. This thus results in feature matrix of sparse vectors, describing whether each word in our dataset appears in each individual startup's text description. Due to our massive dataset of 70,000 companies resulting in a feature set of 90,000 words, we implemented dimensionality reduction via singular-value decomposition to bring our feature set down to a much more manageable size. This dataset is thus much more workable. The vectors representing the location of each company in the reduced dimension space is then clustered together using k-means clustering into 20 clusters. Each cluster is meant to be representative of an industry or grouping of similar startups based on their text description.

This clustering was part of a larger system which was combined with our project for CS221. The overarching purpose of our project is to be able to take a text description of a startup and return a classification into a clustering of related companies.

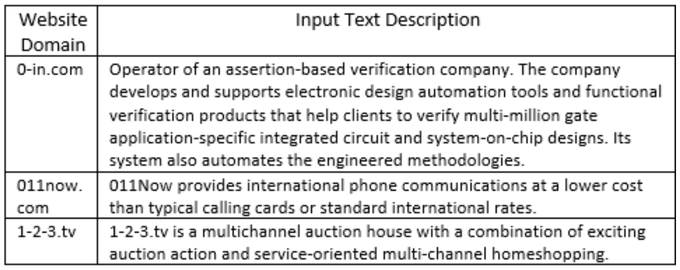
II. Related Work

Our project is essentially attempting to group similar text descriptions together through clustering. Other attempts at measuring similarity in text documents have been researched before. Anna Huang from The University of Waikato, Hamilton, New Zealand, looked at text clustering using different measures of distance and similarity.[2] She took a different direction than our work in focusing on the different measures of similarity between texts. Though we considered going beyond Euclidean distance, we realized the difficulties in doing direct comparison on the word features due to our massive data set.

Additionally, knowing the quality of various clustering algorithms is helpful for figuring out which clustering algorithm best works for our purposes. Yong Gyu Jung, Min Soo Kang and Jun Heo compared the performance of K-means and expectation maximization algorithms.[3] Their results showed that both clustering achieved high accuracy, higher than a classification algorithm they were using as a baseline. However, the K-means algorithm was much more accurate, on the scale of 94% versus 87% for EM clustering. The trade-off they found was in speed, with EM running much quicker. They suggested further optimizations to reduce the operation time, which we carried out in our use of singular-value decomposition for dimensionality reduction.

III. Dataset and Features

Our dataset is a comma separated values file of text descriptions of startups scraped from Pitchbook or Crunchbase. The original CSV contains a lot of unnecessary information for our project, as well as similar information for each startup coming from either Pitchbook or Crunchbase. The categories and industries provided by Pitchbook and Crunchbase were too noisy to use as a meaningful ground truth for our classification. Also, many of the examples in the original dataset either did not have provided descriptions, or the provided descriptions were too short to utilize in a meaningful way. Finally, if the example had a description provided from both Pitchbook and Crunchbase, then we only wanted to use one description. Fig. 1 shows a few examples of the text descriptions we were inputting into our system from the dataset.



Website Domain	Input Text Description
0-in.com	Operator of an assertion-based verification company. The company develops and supports electronic design automation tools and functional verification products that help clients to verify multi-million gate application-specific integrated circuit and system-on-chip designs. Its system also automates the engineered methodologies.
011now.com	011Now provides international phone communications at a lower cost than typical calling cards or standard international rates.
1-2-3.tv	1-2-3.tv is a multichannel auction house with a combination of exciting auction action and service-oriented multi-channel homeshopping.

Fig. 1: Example of startups in dataset after cleaning the CSV

After cleaning, we had dataset resulted in a total of around 70,000 examples. Our feature set taken from this is around 90,000 words, representing all the various vocabulary which appears in the text descriptions of the dataset. Preprocessing we did on our data included removal of stop-words such as ‘and,’ ‘or,’ and ‘the’ from our feature set. The final output of our dataset is a document-term matrix of our examples. To arrive at this document-term matrix, we utilize a vectorizer from the scikit-learn package.[4] First, we utilized a count vectorizer to directly map the amount of appearances of each term in the corpus in each example. After, we utilized a tf-idf vectorizer to weight the terms. We utilized each of the document-term matrices in our clustering.

IV. Methods

We have two primary steps in our method: dimensionality reduction then unsupervised clustering. Due to the massive size of our feature set (over 70,000 startups, and over 90,000 unique words), it would be massively computationally expensive to run our clustering on the actual dataset. Thus, prior to clustering, we had to employ some type of dimensionality reduction on the document-term matrix in order to make clustering computationally possible.

Given m startups and n words (i.e. a $m \times n$ feature space), the goal of this dimensionality reduction is to represent the m startups in terms of the k dimensional vector (where $k <<< n$ is the number of dimensions we choose to reduce our space down to), where each of the k dimensions is some sort of mapping from the original n dimensions. Ideally, the dimensions we reduce to should serve two purposes: first, we want the new dimensions to be a reasonably accurate interpretation of the original feature space. Second, we also want to denoise the original feature matrix, so that different words that imply similar semantics are mapped to the same dimensions so that they are more likely to be clustered together (i.e. ”healthcare”, ”hospital”, and ”medical” are semantically very closely aligned, so ideally we don’t want them to be represented as 3 separate features).

Thus, Deerwester et. al. (1990) propose *latent semantic analysis* (LSA) precisely as a means of accomplishing the two aforementioned goals. LSA works by performing singular value decomposition (SVD) on the document-term matrix, and then using that decomposition to perform dimensionality reduction; under full singular value decomposition, given $X \in \mathcal{R}^{m \times n}$, we decompose so that

$$X = U \Sigma V^T$$

where $U \in \mathcal{R}^{m \times \min(m,n)}$ and $V \in \mathcal{R}^{n,min(m,n)}$ are orthogonal matrices, and $\Sigma \in \mathcal{R}^{min(m,n) \times \min(m,n)}$ is a diagonal matrix of *singular values* σ_j such that $\Sigma_{i,i} \geq \Sigma_{i+1,i+1}$ for all i. To reduce to k dimensions (as is necessary for LSA), let $\Sigma^* = \Sigma_{1:k,1:k}$ (i.e. let Σ^* consist of only the k largest singular values).

Then, if we define $U^* = U_{*,1:k}$ so that $U^* \in \mathcal{R}^{m \times k}$ and $V^* = V_{*,1:k}$ so that $V^* \in \mathcal{R}^{n \times k}$, then a reduced dimensionality approximation of X is given by

$$X \approx U^* \Sigma^* {V^*}^T$$

The k singular values we have exist to scale the k dimensions we've reduced our feature space to, effectively weighting our new dimensions in terms of importance to the original document-term matrix. Thus, in order to represent X in terms of the properly scaled new dimensions, if we let $X^* = U^* \Sigma^*$, then we have k correctly scaled "features" for each of the m startups. It has been shown that among commonly used dimensionality reduction techniques, LSA using SVD had the best "representational richness" (i.e. it does the best job of uncovering latent semantic associations) and was also the most scalable to large datasets [1], which is why we used it for dimensionality reduction. After reducing the dimensionality, we have a data set which is easier to cluster together. On this new, reduced feature space, we run k-means clustering with Euclidean distance to group together our training examples into groups representative of related working industry.

V. Evaluation Metric

We had various hyper parameters in our work. The most obvious one which we fine-tuned for was the number of dimensions which to reduce our dataset to. To do so we came up with our own error metric which combined the error of the reconstruction loss in k-means clustering, with the loss of complexity in dimensionality reduction, measured by the inverse of the explained variance in the reduced dimensions (i.e. the more that the reduced dimensions capture the original variability of the data, the more representative of the original data they are and so the error should be low). Essentially, our metric is meant to represent a tradeoff between dimensionality reduction error and clustering error with respect to our hyperparameter, the number of dimensions reduced to; if the number of dimensions reduced to is extremely low, then our clustering error will be small (because we are in a smaller space), but our dimensionality reduction error would be high, due to the difficulty of representing a 90,000-word space in just a few dimensions. Conversely, an extremely high number of dimensions would correspond to a low dimensionality reduction error (as our new space would be highly similar to our old space), but the clustering error would be very large, due to the fact that we're now working in a much larger space, so Euclidean distances are longer. Putting this tradeoff together, let $X \in \mathcal{R}^{m \times n}$ be our document-term matrix (m documents, n terms), U, Σ, V be our SVD decomposition, K be the number of clusters we have and let $\mu_k \forall k \in [1, K]$ be the centroid location of cluster k. Then we define the following error metric:

$$Error = RSS_{X,K} \times \frac{Var(X)}{Var(X_{transformed})}$$

Where:

$$RSS_{X,K} = \sum_{i=0}^{m} \min_{k \in [1,K]} (x_{i*}V - \mu_k)^2$$

$$Var(M \in \mathcal{R}^{m \times n}) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{i=1}^{n} (x_{ij} - \bar{x}_M)^2$$

$$\bar{x}_{M \in \mathcal{R}^{m \times n}} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{i=1}^{n} X_{ij}$$

VI. Experiments/Results/Discussion

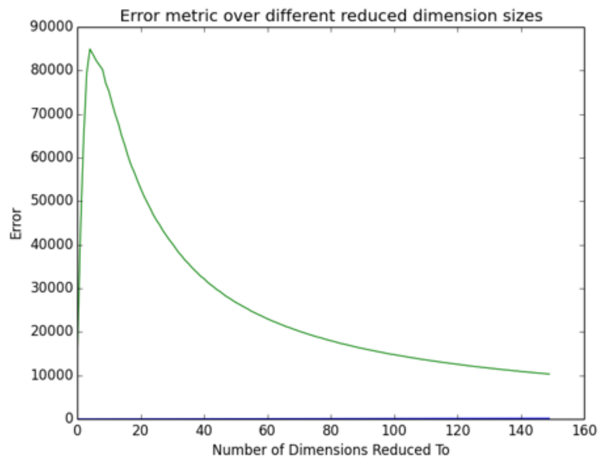


Fig. 2: Error using count vectorization

When performing eye checks of the clustering of the dataset when it was count vectorized, we noticed that there may be issues in the how certain words have an outsize effect on clustering.

Our goal is to be able to group startups together based on working industry or related working area. Examples of this could be health/healthcare or finance tools, due to our intuition that, for example, healthcare startups would use the word health a lot while others would not, and thus cluster together. However, we noticed when looking at random samples of startups from our dataset which were clustered together, we weren't getting the results we expected. Startups which had the word 'provider' or 'company' in their description were getting grouped together. In the context of our project, these words are meaningless, yet they have an outsize effect on our clustering. Startups end up clustered together just because they all say company or provider and don't have meaningful separation aside from that.

To move forward from such an issue, we needed to weight words in their clustering. Our first idea was to utilize term frequency-inverse document frequency, or tf-idf, in building our document-term matrix. The purpose of utilizing tf-idf is to properly weight words which are more important to the whole of a document. We don't necessarily know if 'provider' is appearing across the entire document. If it is though, tf-idf will give words that appear with high frequency across the entire document a much lower weight compared to words which appear in select few documents.

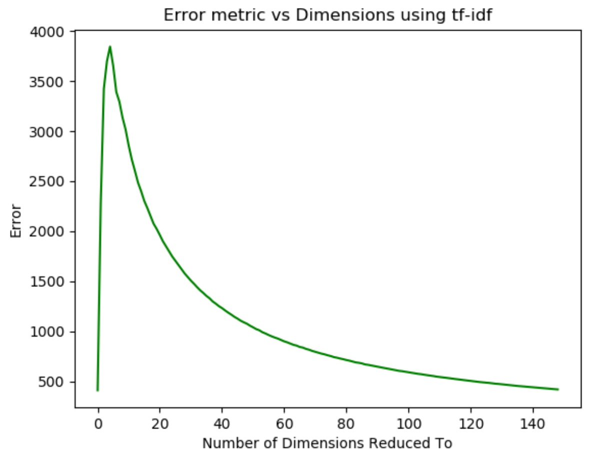


Fig. 3: Error using tf-idf vectorization

Applying tf-idf to the document-term matrix prior to dimensionality reduction and clustering it by k-means reduced the error according to our metric by 20 to 30 times for each dimension (see figs. 2 & 3). Utilizing tf-idf led to a much smaller error per dimension.

One aspect we did not experiment with is the number of clusters into which we were grouping our dataset. Our original thought process was that the number of industries is a relatively inflexible parameter. Thus, all our runs of k-means clustering utilized a parameter of 20 clusters. However, upon further reflection it is thought that the amount of clusters which we are grouping our dataset into could have variable meaning. When there are fewer clusters, the groupings would be more general. Clusters may be representative of larger industries, such as health or mobile. A greater amount of clusters could potentially mean more specific delineation between clusterings. An industry such as health and wellness could be further split into medical device suppliers, mobile life coach applications, and artificial intelligence for radiology.

VII. Conclusion/Future Work

Our project, while initially imagined as a simple clustering problem, grew into examining the trade-offs of dimensionality reduction with clustering. Utilizing dimensionality reduction led to massive efficiency gains in our model. We were able to utilize the full extent of our dataset. We were able to label 70,000 startups with clusterings of related companies.

Unfortunately, the dataset we scraped did not have consistent labelings of industry or product focus for startups. Future work may include scraping and verifying by eye test or utilizing human given labelings for a subset of the dataset to utilize as a ground truth. We could then more easily confirm the accuracy of the clustering algorithm by verifying that startups in the same clustering have the same or similar ground truths.

Additionally, we weren't able to try out clustering algorithms beyond K-means because the evaluation metric we used depended on the loss function from K-means. Thus, using a different error metric for clusters that didn't rely on the K-means objective would allow us to try other algorithms such as Gaussian mixture models, DB-SCAN, and others.

Finally, our project is only a specific application of a model which could be applied to all sorts of problems related to grouping text by meaning. Potential uses include grouping films into genre by plot synopsis, restaurants into cuisine types by menu items, and more. This project is only a portion of the possibilities.

Contributions

Both Meeran Ismail and Daniel Semeniuta contributed to the brainstorming process in which algorithms to implement and the intended input and output of the model.

Meeran focused primarily on the initial architecture for the clustering algorithms, and building the pipeline by which a list of startup descriptions was converted to reduced dimensions, and then clustered. He was also responsible for describing the mathematical motivations behind the SVD and Error Metric on the poster and final report.

Daniel focused primarily on the presentation of the project through the poster and final report. He explained the motivation behind the project as well as the dataset, and the application of count and tf-idf vectorization. He also experimented with small aspects of the codebase, such as applying tf-idft vectorization instead of count vectorization over a range of dimensions.

References

[1] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6), 1990.

[2] Anna Huang. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, pages 49–56, 2008.

[3] Yong Gyu Jung, Min Soo Kang, and Jun Heo. Clustering performance comparison using k-means and expectation maximization algorithms. *Biotechnology & Biotechnological Equipment*, 28(sup1):S44–S48, 2014. PMID: 26019610.

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