Unsupervised Clustering on the CORA Dataset

Joel Stansbury

Introduction

This paper describes a procedure for identifying clusters within a directed graph of one-hot vectors, specifically, the cora dataset, which is a database of papers (represented as one-hot vectors) along with citations between them represented as a directed edge list. The strategy taken is one of feature engineering, in which, the citations and one-hot encodings are combined to produce a single input vector. Gaussian Mixture Model is then used to identify clusters and achieves an accuracy of 69%.

Code: https://github.com/JoelStansbury/IR-pa3

DATA

The cora dataset consists of one-hot vectors representing the text within 2,708 scientific publications. A one-hot vector is a representation of text wherein every position of the array represents a word in the vocabulary. If the i^{th} element has a value of 1, this means that the i^{th} word in the vocabulary is present in the text. The number of times that word occurred in the document is unknown.

Each paper belongs to one of seven categories. Additionally, an edge list, representing citations between the publications, is provided.

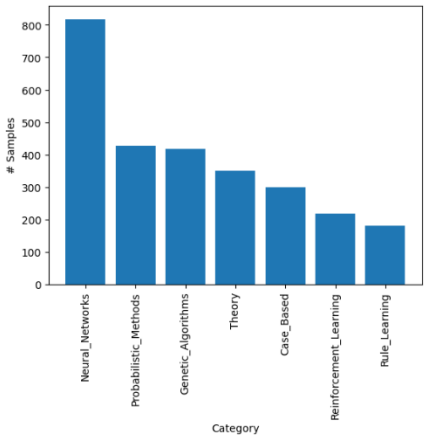

Figure 1: Distribution of categories.

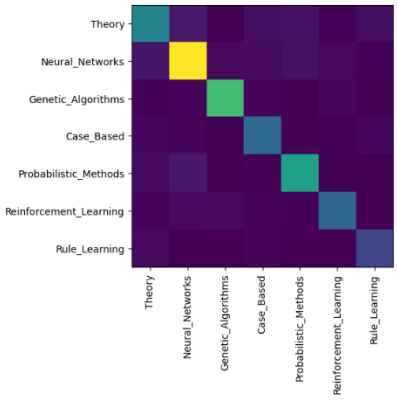

Figure 2: Distribution of in-group vs out-group citations. Rows indicate the distribution of papers which cite papers from the category identified by the row label, while columns indicate the distribution of categories which are cited by papers within the category identified by the column label. Evidently, most citations occur within the same category.

METHODS

Our approach is to model each paper as not only the collection of words contained within the document, but also the collection of words contained within the documents which are cited. This includes immediately adjacent citations (1^{st} degree) as well as citations separated by n intermediate documents (n^{th} degree).

The words within the paper should probably bear more weight on the meaning of the paper than the words within the paper's citations. To account for this, we introduce a decay multiplier (m).

The reference vector is calculated as

$$\boldsymbol{x'_i} = \frac{\mathrm{m}}{N}\sum_{j\in \mathrm{C_i}} \boldsymbol{x_j} + \boldsymbol{x'_j}$$

- $\boldsymbol{x'_i}$: reference vector for document $\boldsymbol{i}$, with information about cited documents.
- $\boldsymbol{C_i}$: set of indices for documents which are referenced by document $\boldsymbol{i}$.
- $\boldsymbol{N}$: number of documents cited by document $\boldsymbol{i}$.
- $\boldsymbol{x_j}$: Document vector for document $\boldsymbol{j}$.
- $\boldsymbol{x'_j}$: Reference vector for document $\boldsymbol{j}$, which is cited by document $\boldsymbol{i}$.

Note that the multiplier (m) is applied recursively such that the impact of an n^{th} degree citation is attenuated by a factor of m^n.

This function does not explicitly handle circular dependencies. The cora dataset contains 220 circular dependency cycles (for example, documents 648112 and 648106 both cite each other). Without explicitly accounting for these, the diffusion function will omit some information from one or more of the members of each cycle (depending on the order of traversal). It is unclear what impact this has on the performance of the resulting vectors.

```
def diffuse(X, adjacency_matrix, m=0.7):
    X_prime = X.copy() * 0
    A = adjacency_matrix.astype(np.bool_)
    roots = list(np.where(A.sum(axis=0) == 0)[0])
    seen = set()
    N = A.sum(axis=0)
    while len(roots):
        i = roots.pop(0)
        children = set(np.where(A[i])[0])
        parents = set(np.where(A[:, i])[0])
        if not i in seen:
            seen.add(i)
            roots += list(children.union(parents) - seen)
            for j in children:
                X_prime[j] += (X_prime[i] + X[i]) * m / N[j]
    return normalize(X_prime)  # scale length to 1.0
```

The diffuse function returns a separate matrix which represents the weighted citation information for each document. To augment the document vectors, we simply add the two together. Experiments involving *weighted* sums of these two matrices did not appear to significantly affect performance.

The decay multiplier, on the other hand, does. Tests showed that between 0.6 and 0.95 produced the best accuracy scores.

The preprocessing pipeline is as follows

1. Dimensionality reduction of one-hot vectors to obtain ***X_pca***
2. Normalization of ***X_pca*** such that the length of each vector is 1.0 to obtain ***X_norm***.
3. Calculate the reference diffusion vectors ***X_ref*** from passing ***X_norm*** into the diffuse function along with the citation adjacency matrix.
4. Add ***X_ref*** to ***X_norm*** and normalize the result to obtain ***X***.

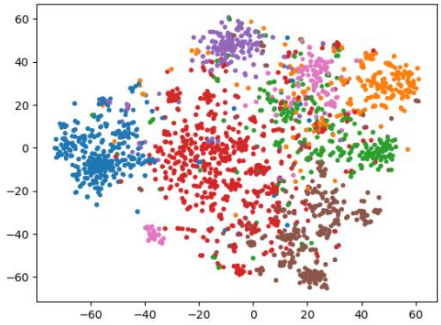

Figure 3: T-SNE representation of the augmented document vectors (m=0.9).

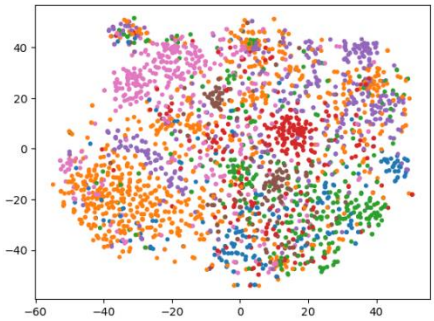

Figure 4: T-SNE representation of the original one-hot vectors.

MODEL

The classification algorithm used is scikit-learn's Gaussian Mixture Model algorithm with n_components set to 7.

RESULTS

Vectors	Accuracy	NMI	ARS
X_pca	0.34	0.15	0.1
X_norm	0.48	0.25	0.23
X_norm + X_ref	0.69	0.5	0.44

After applying the Hungarian algorithm (scipy's linear_sum_assignment) to align the predicted classes with the target classes, The labels produced after GMM clustering achieve an accuracy of 69%.