Modeling and Understanding the Evolution of Innovation in Academia

CS229 Project Final Report

Mengjie Cheng*, Lantao Mei, and Ziyue Wang

Abstract: *In this paper, we study the evolution of innovation in academia by studying the evolution of terminology frequencies of academic publications from Aminer Papers. Our research involves more than 23 million papers distributed during 50 years, from 1967 to 2016. We not only select words that frequently appear in the keyword list, but also use a novel framework named Autophrase to select terms from abstracts. We applied K-means, K-Spectral Centroid Clustering, and Haar Incremental method to classify the frequency distributions of these terms. We evaluated the clustering quality using F value and the sum of the squared distances between cluster centers. Finally, we used decision tree and lasso algorithm to explore what determines the popularity of terminologies in academia.*

1 Introduction

1.1 Motivation

Innovation is one of the core values of development. So far, many works focus on modeling innovation qualitatively, but they may be highly biased by the limitation of human cognition and lacking observations. With the accessibility of big data nowadays, we are hoping to use more data-driven approaches like machine learning and natural language processing to model "evolution of innovation," more specifically, concentrating on understanding and modeling the evolution of scientific terms.

1.2 Objectives

How many growth patterns are there in academic terms? What does each popularity growth pattern look like? And what determines the popularity of terms in academia? So to tackle this problem, we aim to select time-series data of each scientific term with the same length (50 years) using natural language processing, and classify them. In this way, all the scientific terms in the same group have a similar time-series pattern, and the centers of each cluster could be considered as representative patterns of the group. With the clusters and centers, we could explore the key ingredients that determine whether a term is popular or not.

*This project is inspired and extended from the Diffusion of Innovation project led by Professor Daniel A. McFarland in Graduate School of Education

1.3 Raw Data

Our raw data includes 166,192,182 papers from Aminer (https://aminer.org/open-academic-graph) [5, 6]. Among these papers, we focus on $\sim$ 23,000,000 papers. These papers are published between 1967 and 2016, and each of them is a JSON object with an id, a title, year, a keyword list, and an abstract.

We calculate the frequency of 3582 terms from the entire $\sim$ 23,000,000 papers, because we want the frequencies to be as accurate as possible. However, when selecting these terms, we only run on a sample (subset) of these papers, because as long as the terms we select are meaningful our work can proceed. Besides, the recognition features for decision tree and LassoCV are selected from a limited number of sample papers, because generating features like average author citation, may take huge computation time.

2 Methods and Experiments

2.1 Select Terms and Calculate Frequency Distributions

There are two major steps to get the frequency distributions of the terms. First, we select 1000 - 2000 terms that are likely to be important for our date. At this step, we only select from a small sample of our data, because it takes a large amount of computational time to judge whether a word or phrase is important or not. Second, we calculate each term's frequency of each year using the entire Aminer data (more than 23 million papers).

2.1.1 Select Terms from Keyword Lists

We select terms from two sources. To begin with, there is a keyword list for each of the 23 million papers. We choose 50,000 papers as a sample, and calculate the most frequent terms in this sample. Specifically,

$$S(term) = \sum_{y=1967}^{2016} \frac{N_{list}(term, y)}{N(y)} \quad \text{(among } \sim\text{50,000 papers)} \tag{1}$$

where $N_{list}(term, y)$ is the number of papers with their keyword lists that contain that term at year y, and N(y) is the total number of terms at year y. We take 2000 terms with the highest scores as our sample terms. The denominator $N(y)$ is used as a normalization, since the total numbers of papers are different in different years.

2.1.2 Nature Language Processing (NLP): select terms from Abstracts

Besides using the terms from the keyword lists, we also use NLP techniques to select both terms from abstracts in the data set. AutoPhrase is adopted for its ease of use and for it requires limited, shallow linguistic analysis[2]. We use AutoPhrase to select key phrases from "abstract". The result is shown in the "Results and Evaluation" section.

AutoPhrase is a novel automated phrase mining framework to select quality phrases without additional manual labeling effort using two new techniques *Robust Positive-Only Distant Training* and *Part-of-speech Guided Phrasal Segmentation* respectively [2]. The first technique uses quality phrases from general knowledge bases such as Wikipedia together with the input corpora to train several base classifiers. Those classifiers will separate the input text corpora into either positive pool (high-quality phrases) or negative pool (the rest phrases). Next, random forest is applied to compute the score of each phrase. A decision tree is created where no two positive and negative phrases share identical feature values in the 2K training set. And the phrase quality score is computed as the proportion of all decision trees that predict a particular phrase as a quality phrase [2]. The second technique is used after the above step to separate phrase into a complete semantic unit. The corpus is processed as a POS-tagged word sequence and is partitioned into m segments induced by a boundary index sequence. Viterbi Training (Hard EM in literature) [2] is used to find the best segmentation which is the one that maximizes the joint probability of word sequence and boundary index sequence. The phrase score is reestimated using the result of POS-guided phrasal segmentation.

2.1.3 Frequency Distributions

Using the terms we obtained from 2.1.1 and 2.1.2, we calculate frequency distributions.

For each term from the keyword lists, we calculate the frequency using the following formula.

$$x_{list}(term) = \frac{N_{list}(term, y)}{N(y)} \quad \text{(among } \sim \text{20,000,000 papers)} \tag{2}$$

Similarly, for terms from abstracts, the frequency is given by

$$x_{abst}(term) = \frac{N_{abst}(term, y)}{N(y)} \quad \text{(among } \sim \text{20,000,000 papers)} \tag{3}$$

where N_{abst} is the number of papers with abstracts that contain that term.

2.1.4 Gaussian Kernel Smoothing

We use Gaussian kernel smoothing[1] to smooth the distributions we obtain from 2.1.3. This method is described as following.

Given a distribution $x = (x_1, ..., x_m)^T$, (in our case, m=50), we define a kernel k for any of two elements in f. That is,

$$ker(x_i, x_j) = e^{\frac{-(x_i - x_j)^2}{2\sigma^2}} \tag{4}$$

where sigma is chosen to be 1 (year), which is the width of our bins. Then, each element of the smoothed data $x_s = (x_{s1}, ..., x_{s1})$ is given by

$$x_{si} = \frac{\sum_{j=1}^{m} ker(x_i, x_j) x_j}{\sum_{j=1}^{m} ker(x_i, x_j)} \tag{5}$$

2.2 Baseline: K-means Algorithm

Firstly, we use k-means clustering algorithm. After initializing the cluster centroids randomly, we repeat two steps until convergence: i) Assigning each training example $x^{(i)}$ to the closest cluster centroid. ii) Moving each cluster centroid μ_j to the mean of the points assigned to it. By repeating these two steps, k-means minimize the sum of the squared Euclidean distances between the members of the same cluster. But there are several problems involved when coping with time-series data using k-means clustering. First, if two-time series have very similar shape but different overall volume, they should still be considered similar. Thus, scaling the time series on the y-axis should not change the similarity. Second, different items appear and spike at different times. Thus, translating time series on the time axis should not change the similarity between the two time series[1].

2.3 K-Spectral Centroid Clustering

To solve the two problems we briefly mentioned above, we come to the K-Spectral Centroid Clustering(K-SC)[1] to do our term clustering. Firstly, we adopt a distance measure that is invariant to scaling and translation of the time series[4].Given two time series x and y, the distance d(x, y) between them is defined as follow:

$$d(x,y) = \min_{\alpha,q} \frac{\|x - \alpha y_q\|}{\|x\|} \tag{6}$$

This measure helps to find the optimal alignment(translation q) and the scaling coefficient α. Like K-means algorithm, we start the K-SC algorithm with a random initialization of the cluster centers. We repeat two steps until convergence: i)Assigning each training example $x^{(i)}$ to the closest cluster centroid based on the distance we just defined. ii)Updating the new cluster center be the minimized of the sum of $d(\mu_i, \mu_j)^2$ over all $x_i \subset C_k$ corresponding to the smallest eigenvalue λ_m of matrix M.

$$\mu_k = argmin_\mu \sum_{x_i \subset C_k} d(x_i, \mu_j)^2 = argmin_\mu \frac{\mu^T M \mu}{\|\mu\|^2} \tag{7}$$

$$M = \sum_{x_i \subset C_k} (I - \frac{x_i x_i^T}{\|x_i\|^2}) \tag{8}$$

2.4 Haar Incremental Algorithm for K-SC

We were intended to use Haar incremental method[1] to speed up our K-SC method. Haar transformations are used to compress our data into lower dimensions. First, we run the K-SC algorithms on the compressed and rougher datasets for several iterations. Second, instead of using random initial cluster assignments, then we use the results from the results of the compressed dataset as initial assignments. Since these results are approximations of the final results, Haar incremental K-SC should lead to a smaller computational time.

However, the dimension of our data is only 50, and thus the original K-SC algorithm is already fast ($\sim$ 50 seconds). Therefore, we did not observe a significant boost in the speed of the incremental method.

2.5 Decision Tree

After exploring the popularity growth patterns of various scientific terms, we continue to explore our next question concerning the key ingredients of a term being popular or not. We come up with eight candidate features2 to represent two important indicators:past success and recognition. Then we try to use decision tree to identify the importance of our features.

We use the class DecisionTreeClassifier from scikit learn [7]. Given training vectors $x_i \subset R^n$, i=1,2,3...l, and a label vector $y \subset R^l$, a decision tree recursively partitions the space such that the samples with the same labels are grouped together. Let the data at node m be represented by Q. For each candidate split θ = (j, t_m) consisting of a feature j and threshold t_m, partition the data into $Q_{left}(\theta)$ and $Q_{right}(\theta)$ subsets. The impurity at m is computed using an impurity function H()

$$G(Q,\theta) = \frac{n_{left}}{N_m H(Q_{left}(\theta)))} + \frac{n_{right}}{N_m H(Q_{right}(\theta)))} \tag{9}$$

Then we select the parameters that minimize the impurity.

$$\theta^* = argmin_\theta G(Q,\theta) \tag{10}$$

And our measure of impurity for classification is Gini

$$H(X_m) = \sum_k p_{mk}(1 - p_{mk}) \tag{11}$$

where $p_{mk} = 1/N_m \sum_{x_i \subset R_m} I(y_i = k)$

2.6 Lasso

We run a lasso algorithm with exactly the same data that we used in the decision tree model. We want to compare the ingredient results of these two models. The reason that we are using lasso algorithm instead of other classification model is that our goal is to identify the importance of our features, rather than predicting the classifications according to the features.

We use the function "LassoCV" from scikit learn [7]. The loss function is given by

$$\frac{||y - Xw||_2^2}{2n} + \lambda ||w||_1 \tag{12}$$

where w are the parameters and n is the total number of samples. We normalize our features by setting the parameter "normalize" to be "True", so that the weights are comparable. The regularization parameter λ is optimized using cross validation (CV) by this function.

3 Results and Evaluation

3.1 NLP using AutoPhrase

The results of AutoPhrase is reasonable. For instance, given the abstract of Aminer papers, the top 10 key phrases with the highest scores: clay minerals, Latin America, southeast Asian, programming languages, spallation neutron source, Persian gulf, histone acetyltransferase, intelligence quotient, analytical chemistry and shock waves.

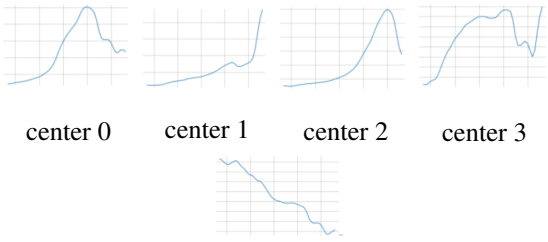

Figure 6: Results of Incremental K-SC: Terms from Both. On each plot, the horizontal axis represent 50 years while the y axis is the occurring frequency during that year

Methods	$d(x_i, \mu_k)^2$	F
K-means: Terms from Absts	1.87	471.05
K-SC: Terms from Absts	4.55	208.26
Inc K-SC: Terms from Absts	4.54	208.24
K-means: Terms from Kw Lists	0.14	442.09
K-SC: Terms from Kw Lists	3.52	198.13
Inc K-SC: Terms from Kw Lists	3.59	199.48
Kmeans: Terms from Both	0.13	949.24
K-SC: Terms from Both	4.18	471.23
Inc K-SC: Terms from Both	4.18	471.23

Table 1: Cluster Quality Evaluation. 'Kw' refers to 'Keyword' and 'Abst' refers to 'Abstract'.

3.2 Unsupervised Learning

We use different clustering number K such as 3,4,5...8 and find that when the clustering number equals 5, the patterns of clustering is the most reasonable. The result of incremental K-SC clustering is shown in Fig. 6 Cluster 0 shows a sharp increase and then moderate decline pattern, which includes 608 terms such as "predictive models", "computational geometry", etc. Cluster 1 displays pattern that increases sharply in most recent years, which includes 593 terms such as "translational research", "risk assessment", etc. Cluster 2 has a peak during later time periods, which includes 364 terms such as "heavy metal", "teaching", etc. Cluster 3 includes a sharp decay during later time periods, which includes 1190 terms such as "directed graph", "normal distribution", etc. Cluster 4 shows a continuously decay, which includes 827 terms such as "ammonium sulfate", "fracture mechanics", etc. Examples of distributions of specific terms can be found in Fig. 8. By inspection, the clustering is reasonable.

To further evaluate the performance of each clustering method discussed in section 2, we computed two performance metrices:

(a) Objective function F as defined in Equation 13.

$$F = \sum_{k=1}^{K} \sum_{x_i \subset C_k} d(x_i, \mu_k)^2 \tag{13}$$

It can be seen from this equation that Function F value is strongly influence by the length of the term list (abstracts, keyword lists, both). Therefore, comparing F value from different term list is meaningless.

(b) The sum of the squared distances between the cluster centers: $d(x_i, \mu_k)^2$. Since the number of centers is independent of the length of term list, this performance metric does not depend on term list length. Function F measures the compactness of the cluster, while the distances between the cluster centers measure the diversity of clusters[1]. Therefore, a good clustering has a low value of F (computed using the same term list) and high value of sum of squared distances between cluster centers. Table 1 presents our quality evaluation results. It can be seen that the performance of incremental KSC is approximately the same as regular KSC which comfirms our statement in 2.4. And both incremental KSC and regular KSC has better performance than K-means.

3.3 Decision Tree and Lasso

Our result of decision tree is shown in Fig. 7, we could see that the most important feature is (3)prob_change_k3 and then comes with (7)avg_venueAvgCitation, (8)avg_paper_freq, (0)paper_counts and (4)avg_paper_citation. The weights of the eight features from LassoCV are given in table 3. Among all the features, (2)prob_change_k1, (3)prob_change_k3, (4)avg_paper_citation, (7)avg_venueAvgCitation and (8)avg_paper_freq have higher weights, while the decision tree shows (0), (3), (4), (7), and (8) to be the important features. The reason for the exceptions (0 and 2) is that the decision tree is not a linear model; instead, it classify through paths. In spite of the exceptions, the LassoCV model still verifies the reliability of the decision tree model.

4 Conclusions

In this project, we apply K-means and K-SC algorithms, and verified that K-SC as well as incremental K-SC algorithms perform better than K-means in solving time-series clustering problems. Furthermore, we did not observe significant effect on short feature vectors (50) for Haar incremental K-SC. We apply decision tree algorithm to demonstrate that both past success and recognition can have influence on the pattern of a given term and past success is more important than recognition. Lasso algorithm is then applied to v

Past Success	(0) paper_count	# of publications where a term is mentioned
	(1) norm_paper_count	Proportion of # of publications where a term is mentioned to all publications
	(2) prob_change_k1	Relative changes of probability occurring in publications in 1 year
	(3) prob_change_k3	Relative changes of probability occurring in publications in 3 years
	(8) avg_paper_freq	Average freqency that a term is mentioned each year
Recognition	(4) avg_paper_citation	Average # of citaions for supporting papers
	(5) avg_author_citationAvg	Average # of citaions for all authors who wrote about it
	(6) avg_author_citationTotal	Total # of citaions for all authors who wrote about it
	(7) avg_venue_citationAvg	Average # of citaions for all venues where it got written

Table 2: Features that are used for Decision Tree and LassoCV

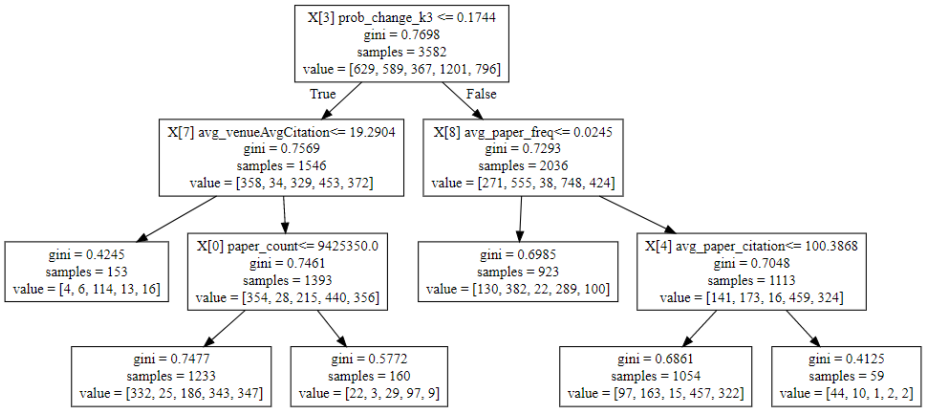

Figure 7: Decision Tree. The view of this tree is generated from http://www.webgraphviz.com/.

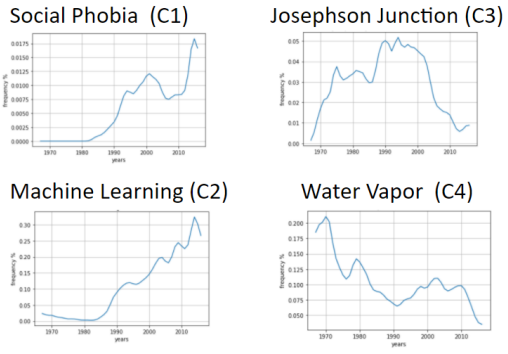

Figure 8: Several example distributions of terms. Ci indicate the clusters that the terms are belong to from Fig. 6. Ci means the term is belong to the i^{th} cluster.

No.	Features	Weights
0	paper_count	-2.32e-08
1	norm_paper_count	0
2	prob_change_k1	1.51e-02
3	prob_change_k3	-2.51e-02
4	avg_paper_citation	-1.94e-02
5	avg_author_citationAvg	5.28e-04
6	avg_author_citationTotal	-6.72e-04
7	avg_venueAvgCitation	1.10e-02
8	avg_paper_freq	5.74e-01

Table 3: LassoCV Results

Contributions

Mengjie Cheng: Motivation and objectives, K-means algorithm, K-SC algorithm, feature generation, results and evaluation, decision tree.

Lantao Mei: Parse raw data, NLP to select terms from abstract, results and evaluation.

Ziyue Wang:selecting terms from keyword lists, calculation of frequency distributions, Guassian kernel smoothing, incremental K-SC, Decision Tree, LassoCV.

Acknowledgement

We thank all the instructors of Fall 2017 CS229 machine learning class, including Prof. Andrew Ng, Prof. Dan Boneh and all the teaching assistants, for bringing us such a meaning experience during this quarter.

References

[1] Yang, Jaewon, and Jure Leskovec. "Patterns of temporal variation in online media." Proceedings of the fourth ACM international conference on Web search and data mining. ACM, 2011.

[2] Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R Voss, Jiawei Han, "Automated Phrase Mining from Massive Text Corpora", submitted to TKDE, under review. arXiv:1702.04457

[3] Jialu Liu*, Jingbo Shang*, Chi Wang, Xiang Ren and Jiawei Han, "Mining Quality Phrases from Massive Text Corpora, Proc. of 2015 ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'15), Melbourne, Australia, May 2015.

[4] K. K. W. Chu and M. H. Wong. Fast time-series searching with scaling and shifting. In PODS 99, 237248, 1999.

[5] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. ArnetMiner: selection and Mining of Academic Social Networks. In Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'2008). pp.990-998. [PDF] [Slides] [System] [API]

[6] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In Proceedings of the 24th International Conference on World Wide Web (WWW 15 Companion). ACM, New York, NY, USA, 243-246.

[7] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.