**Analysis of Knowledge Discover in Databases Papers on Data Mining Models**

Wenke Huang, whuang67

1. **Introduction**

The KDD stands for Knowledge Discovery in Databases which is one of the largest and best Data Science conferences. Since 1989, many highly cited research papers were published in KDD. The technology and research discussed at KDD is often many years ahead of any other conference.

The dataset of this project contains the first 100 papers (“kdd15-p9” to “kdd15-p1015”) of KDD in the year of 2015 which is provided by Professor Meng Jiang. In this project, we would like to perform our analysis by using four different data mining tasks which include Frequent Pattern Mining, Classification, Clustering Analysis and Outlier Detection.

**Keywords:** Frequent Pattern Mining, Classification, Clustering Analysis, Outlier Detection

1. **Data Preprocessing and Data Cube Construction**

In this section, we would like to briefly introduce how we process our dataset and convert it from textual papers into an understandable format, and what dimensions we include in our data cube.

* 1. **Data Processing**

Currently, we have several methods that can help us build our data mining models by using text message after they are transformed, such as Term Frequency-Inverse Document Frequency (TF-IDF) and truncated Singular Value Decomposition (truncated SVD) which will be used later.

TF-IDF is a useful tool that can reflect how important a word is to a document in a collection or corpus based on their term frequencies and inverse document frequencies. Additionally, we will also try truncated SVD which is a factorization of a real or complex matrix. Particularly, truncated SVD works on TF-IDF matrices. Under this situation, it is known as Latent Semantic Analysis (LSA).

Totally, I finished processing the first 100 papers of the 224 papers of the year of 2015 (“kdd15-p9” to “kdd15-p1015”). I did not keep going further to process more because of data cleaning issues which will be discussed in the next part.

* 1. **Data Cleaning**

It is easily to notice that our textual dataset is not very clean. We have several special characters due to our textual dataset being converted from PDF files directly. We would like to remove them to lower or eliminate the influence that they will bring to us.

Additionally, I also removed all numerical characters. Because numerical characters in KDD papers always represent the examples that scholars used, the index of sections, tables and graphs, et cetera, which do not affect the main idea of papers. For example, let us suppose that there are two papers mainly talking about K-Means algorithm, but using different examples. The distance between these two papers may be increased significantly because of the difference scales of variables of these two different examples.

For the previously two issues, I would like to perform Regular Expression to extract only English alphabets. Apart from them, I also converted all upper-case letters to lower-case ones, or two words with same meanings may be considered as two distinct ones because of their cases.

There are some other minor issues in our dataset, like middle name versus middle initial, et cetera. I try to correct them by using my hands and eyes since they are easily recognized. For example, if we sort the dataset in either ascending or descending order, they tend to be together.

* 1. **Data Cube**

Our data cube here uses Star Schema where the fact table in the middle connected to a set of dimension tables here. The fact table which contains our measure count and keys to each related dimension tables of this data cubes is shown below.

Table 1: fact table of our data cube

|  |
| --- |
| Title-key |
| Author-key |
| Category-key |
| Whole-paper-key |
| count |

The dimension tables of both fact tables will not be shown here since we do not have any hierarchy in the dimensions of either data cubes. We can see that the data cubes operations including roll-up, drill-down, slice, dice and pivot are all feasible here.

The explanations of meaning of different dimensions are listed below.

* Title: The title of the paper.
* Author: The writers of a certain paper. Usually, there are at least two authors in one paper.
* Whole-paper: The whole text message of the paper.
* Category: The categories of data mining topics that a paper belongs to. We manually categorize papers into two: non-database (0) and database (1).

1. **Tasks Definition**

In this part, we would like to introduce the four data mining tasks which are Frequent Pattern Mining, Classification, Clustering Analysis and Outlier Detection.

* 1. **Frequent Pattern Mining**

Pattern stands for a set of items, subsequences or substructures that are strongly correlated or occur frequently together in a data set. For example, bread and butter usually are purchased together in the supermarket, and Computer Scientists are always good at C/C++ Programming Language.

In this project, we would like to use Apriori Algorithm here to help us find the frequent patterns. The principle of this algorithm is that if there is any itemset which is infrequent, its superset which cannot be frequent should not even be generated. The absolute and relative support here stand for the frequency of an itemset X and the fraction of transactions which contains X, respectively. The confidence stands for the conditional probability that a transaction which contains X also contains Y.

Because of the limitation of the support-confidence frame work, we would also try four additional dependent measures which are Lift, Chi-square, Cosine and Kulczynski in this project. Lift and Chi-Square among these four are not a null invariant measure while the rest two are. Null invariance stands for value not changing with the number of null transactions. These four measures can be given as:



* 1. **Classification**

Classification is the problem of identifying which of a group of categories one instance belongs to by using the classifier that we have. The classifier is trained based on the training dataset which contains both instances and their corresponding class label. Hence, it is one of the supervised machine learning algorithms, and usually.

There are many existed machine learning algorithms which can help us build possible classifiers. In this project, we would like to use Naïve Bayes algorithm to fit our classifier. In Naïve Bayes, we assume that given the class label, all features are conditionally independent. Here, we would like to derive the maximum posteriori which can be calculated from Bayes’ theorem. We can also notice that the priori which is the denominator is a constant, which means that comparison between joint distributions can give us the exactly same result. If feature A is a categorical one,  is the number of tuples in  which have value  for feature A divided by number of tuples of  in our whole training dataset. If feature A is continuous one, its conditional distribution can be computed based on Normal distribution .

We would like to extract 80 papers to be our training sets while the rest 20 are our testing sets. Training and testing sets will be split randomly. We will use the training set to fit our classifier and make predictions based on both training and testing ones. ROC curve (with AUC) will used as our metric here. Apart from it, we will also try to obtain the confusion matrix including accuracy, recall, precision and F1 score. We will mainly focus on the testing performance while training performance will help us detect if we over- or under-fit our model.

* 1. **Clustering Analysis**

Cluster Analysis which is one of the main task of Data Mining is a kind of algorithms that help us group (cluster) a set of objects. In Clustering Analysis, we would like to maximum the difference among different clusters while keep the intra-class similarity as high as we could. Unlike Classification which have discussed before, Clustering Analysis is one of the unsupervised learning since there is no pre-defined class label here.

In the following part, we would like to perform K Means clustering algorithm to cluster the datasets that we have in hand. In K Means clustering, we first should set our initial centroids. The number of initial centroids should be equal to the number of clusters (Let’s suppose k clusters here) that we would like to have. Second, we form k clusters by assigning points in our training set to their corresponding closest centroid. Then, we re-calculate the centroids of each clusters, and update our clusters by repeating the previous steps based on our updated centroids. We will stop until if the points in the cluster do not change or other convergence criterion is satisfied.

Apart from K-Means algorithm, we would also performance Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm. One of the differences between this algorithm and K-Means is that DBSCAN eliminates noise points and makes each group of connected core points into a separate cluster. There are two parameters “Eps” and “MinPts” which help us define core points, border points and outliers in this algorithm.

* 1. **Outlier Detection**

Outlier Detection is the process of detecting outliers from a given set of data. An outlier can be defined as observations that derivates from the given dataset. There are many reasons which may result in outliers, such as, data entry mistakes, unexpected facts, et cetera.

Here, we will borrow the DBSCAN algorithm which we mentioned in part 3.3. Because while we group each connected core points into different clusters, the points which cannot be assigned into any cluster are considered as outliers in this case. Outliers may not be exactly far away from other points. The criteria are that the number of points within Eps-neighborhood is less than “MinPts” and at the same time, they are not density-connected to any other points.

1. **Data Mining**

In this section, we would like to implement those three data mining tasks and the corresponding problems which we are interested in.

* 1. **Frequent Pattern Mining**

First, like we have mentioned before, we can easily notice that usually more than one scholars contribute to one published paper. Furtherly, we would like to discover if some scholars tend to research together. For example, knowing that two professors made one publication together last year, we would like to see if they made some other publications together, too. However, most scholars will publish only one paper in KDD in one year.

In this case, we will use Apriori Algorithm to help us find the frequent patterns. As Apriori Algorithm says, firstly we scan the whole dataset and get the one-item sets. Here, we would like to set our relative minimum support to be 2%, which means that only those authors who published at least three papers will be considered. Because for an author who only co-published one or two papers in 2015, the confidences of himself to all his co-workers are definitely less than 0.5 which makes less sense here. Then, we will generate our two-item sets based on the results here. Like the Apriori Algorithm, we only focus on the one-item sets which are frequent and ignore those infrequent ones. The relative minimum support is also 2% here. After getting two-item sets, we stop and do not focus on their subsets because we would like to focus mainly on their association here.

The minimum confidence here are set to be 0.5. Both relative support and confidence here are for association rules of Author 1 to Author 2. There are 17 frequent patterns that follow our conditions and they can be found in the attached files. Part of the results are listed below. We round the results to 4 decimal places.

Table 2: Frequent Patterns (support and confidence)

|  |  |  |  |
| --- | --- | --- | --- |
| Author 1 | Author 2 | Support | Confidence |
| Bo Zhao | Jiawei Han | 0.02 | 1 |
| Bo Zhao | Jing Gao | 0.02 | 1 |
| Takanori Maehara | Ken-ichi Kawarabayashi | 0.02 | 1 |
| Yaliang Li | Jiawei Han | 0.02 | 1 |
| Wei Fan | Hanghang Tong | 0.02 | 0.6667 |
| … | … | … | … |

From the Table 2, we can clearly find some associations among different authors. For example, Dr. Bo Zhao has two publications, both of which were done with the help of Professor Jiawei Han, and Dr. Wei Fan has three publications, two of which are with Professor Hanghang Tong.

Additionally, I would like to take a further look at our results above. We are about to calculate four more measures which are Lift, Chi-square, Cosine and Kulczynski to furtherly test association rules between different authors.

The results are listed below. Here, we only include the author pairs which have been printed out in the previous table. Here, Lift and Chi-square are rounded to 2 decimal places while Cosine and Kulczynski are rounded to 4.

Table 3: Frequent Patterns (Lift, Chi-square, Cosine, Kulczynski)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Author 1 | Author 2 | Lift | Chi-Square | Cosine | Kulczynski |
| Bo Zhao | Jiawei Han | 25 | 48.98 | 0.7071 | 0.75 |
| Bo Zhao | Jing Gao | 50 | 100 | 1 | 1 |
| Takanori Maehara | Ken-ichi Kawarabayashi | 33.33 | 65.99 | 0.8165 | 0.8333 |
| Yaliang Li | Jiawei Han | 25 | 48.98 | 0.7071 | 0.75 |
| Wei Fan | Hanghang Tong | 22.22 | 43.08 | 0.6667 | 0.6667 |
| … | … | … | … | … | … |

We can easily read from this table that all Lifts and Chi-squares are much greater than 1 and 0, respectively, which tells us a strongly association between the two authors in the corresponding pairs. We think these two results are less persuasive. It is because that among the 100 KDD papers of 2015 that we processed, very few researchers published more than three papers, which illustrates that null transactions are very large here. Here, null transaction stands for papers that were finished by neither Author 1 nor Author 2. Then, we move our concentration to Cosine and Kulczynski. We can read from their formulas which are mentioned in part 3 that these two measures are not influenced by null transactions, which makes our results more trustable.

* 1. **Classification**

In this part, we would like to build a classifier to classify our 100 papers into database-related and database-unrelated ones. The Category is our pre-classed variable that we would like to predict while the Whole-paper is our feature. Among the 100 papers that we processed, 49 of them are related with topics of database while the others are not.

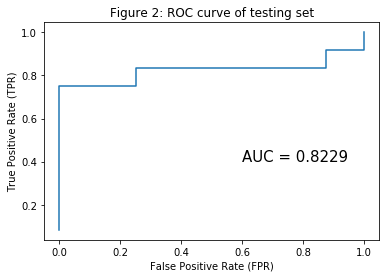
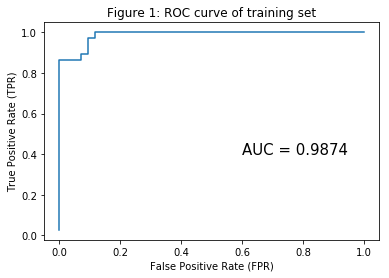
At the beginning, we would like to randomly split our datasets into training and testing sets, which we have mentioned in the previous part. The training set will contain 80 papers while the testing set contains the rest 20. We also need to check whether this is a class-balanced binary classification problem. Class-balanced dataset means that we have rare positive example while numerous negative ones.

Table 4: Class-Imbalanced Datasets Check

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | total |
| training | 43 | 37 | 80 |
| testing | 8 | 12 | 20 |
| all | 51 | 49 | 100 |

The results above look good since the proportion of 0 and 1 are roughly 50% and 50% in any of these three datasets. Furthermore, we know that the error costs of both classes are basically equal since we do not have any preference on either of them.

The classifier that we built is based on Naïve Bayes algorithm. We fit our model by using our training set and evaluate its performance by using both training and testing sets. The metric that we used here is ROC curve (with AUC).



Our ROC curves and their corresponding AUCs look great. The horizontal and vertical axes represent False Positive Rate (FPR) and True Positive Rate (TPR), respectively. The plots tell us that both rates are greater than 0.82. We may think that the performance here is acceptable.

Apart from ROC curve plots above, we will also find the confusion matrices including accuracies, recalls, precisions and F1 scores.

Table 5: Confusion Matrix of Training Sets

|  |  |  |  |
| --- | --- | --- | --- |
|  | predicted: 0 | predicted: 1 | total |
| actual: 0 | TN = 43 | FP = 0 | 43 |
| actual: 1 | FN = 5 | TP = 32 | 37 |
| total | 47 | 32 | 80 |

Table 6: Confusion Matrix of Testing Sets

|  |  |  |  |
| --- | --- | --- | --- |
|  | predicted: 0 | predicted: 1 | total |
| actual: 0 | TN = 4 | FP = 4 | 8 |
| actual: 1 | FN = 2 | TP = 10 | 12 |
| total | 6 | 14 | 20 |

Table 7: Accuracy, Specificity, Sensitivity and F1 Score

|  |  |  |
| --- | --- | --- |
|  | training | testing |
| accuracy | 0.9375 | 0.7 |
| recall | 0.8649 | 0.8333 |
| precision | 1 | 0.7143 |
| F1 score | 0.9275 | 0.7692 |

Table 5 and 6 show the confusion matrices of training and testing sets, respectively. Table 7 lists the accuracies, recalls, precisions and F1 scores of training and testing sets. We can read from these three tables that the performance of training set is very good while the testing result is still acceptable but not as good as our training performance.

* 1. **Clustering Analysis**

Here, we would like to perform Clustering Analysis to check if we can group these 100 papers into several clusters successfully. If yes, we want to take a deeper look at these clusters and try to discover if there exist some connections among papers within the same clusters. If no, we may discuss the possible explanation of this failure. The first algorithm we are about to use here is K-Means Clustering. The dimension that we are about to use is dimension Whole-paper only since it contains the whole textual information of one paper.

At the beginning, the number of clusters is an important decision that we need to make. Here, we would like to introduce Silhouette Coefficient which is calculated based on both the mean intra-cluster distance and the mean nearest-cluster distance for each sample. The range of this coefficient is from -1 to 1. The negative coefficient represents different clusters being more similar. Values near 0 indicate overlapping clusters. Positive coefficients tend to be more preferred. We fit 6 models with different number of clusters ranging from 2 to 7 (including 7 here), the Silhouette Coefficient of which are listed in the table below.

Table 8: Silhouette Coefficients of Different No. of Clusters (K-Means)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. of Cluster | 2 | 3 | 4 | 5 | 6 | 7 |
| Silhouette Coefficient | 0.1595 | 0.135 | 0.1334 | 0.0126 | 0.0046 | 0.0021 |

We can read from the above table that the best number of clusters should be set 2 while 3 or 4 clusters are still in the acceptable range since their coefficients are only slightly less than the one of 2. The number of clusters being greater than 4 is not allowed since the Silhouette Coefficient drops towards 0 immediately, which stands for the existence of overlapping clusters.

However, the clustering results still gave us a big surprise. We found out that most of the points were assigned into 1 cluster. For example, when cluster number is 2, only paper “kdd15-p309” was assigned to another cluster. Similar stories happen we set number of clusters to be 3 or 4. This phenomenon makes us think that whether most of these 100 papers have some similarities, which may be true since they are all from KDD conference. Maybe, it is more reasonable if we treat those papers which were assigned into minority groups as outliers instead of distinct clusters.

In addition to K-Means algorithm that we just tried, we would like to additionally use DBSCAN algorithm to see if this one can give us a more reasonable clustering results. In this algorithm, we have two parameters “Eps” and “MinPts”. Here, “MinPts” is set to be fixed 5 while “Eps” is considered as a tuning parameter ranging from 0.85 to 1.05 by 0.05. We would like to check how many clusters and outliers DBSCAN algorithm will give us.

Table 9: Clustering Analysis Results (DBSCAN)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Eps | 0.85 | 0.9 | 0.95 | 1 | 1.05 |
| No. of Cluster | 1 | 1 | 1 | 1 | 1 |
| No. of Outliers | 41 | 23 | 12 | 7 | 5 |

As the “Eps” increases, the number of outliers goes towards 0. When “Eps” is equal to or greater 1.1, there will be no points being considered as outliers among the 100 papers that we processed. Unfortunately, DBSCAN algorithm still can’t help us group our papers into more than one cluster. The results of both K-Means and DBSCAN algorithms tell us that most of the 100 papers share some similarities with each other while few of them are different from the rest.

* 1. **Outlier Detection**

Like we mentioned previously, we will not use any new algorithm but borrow the results from DBSCAN from part 4.3. This is also a good example telling us that the algorithms may be multi-functional. For example, even though we did not successfully group our papers into several clusters by using Clustering Analysis, this algorithm helps us detect the outliers among these 100 papers.

Table 9 tells us that when “Eps” is set to be greater than 0.95, most of the papers tend to be grouped into the same cluster and less than around 10 papers are considered as outliers here. We can think about taking a deeper look at those papers and trying to discover if they have significance difference from the rest majority papers.

Here, if we compare the results of K-Means and DBSCAN algorithms, we will find some similarities between these two. From K-Means, even though there is no technically defined outlier, we remember that paper “kdd15-p309” alone is put into a different cluster other than the rest 99 papers. The same paper is also defined as one of the outlier papers from DBSCAN algorithm without doubts.

1. **Summary and Conclusion**

For Frequent Pattern Mining, we successfully found that several scholars do research together. We can think that scholars usually do have a fixed research group, which can explain why scholars tend to publish papers with a certain group of other scholars. On the other side, there are always new members from the name list. It is always good to embrace new ideas and blood from the outside since creativity is extremely essential to the research.

For Classification, we successfully built a classifier. We can think that it is feasible to classify different research papers into database or non-database category. Actually, several KDD papers do not contain the “Category” part at the beginning, which may be ignored by the authors. This classifier can help us categorize those papers with missing “Category” into the class that they belong to.

For Clustering Analysis, we found that the papers tend to be similar with each other. Since all of them are from KDD conference in 2015, that they share some ideas or topics also makes sense.

For Outlier Detection, we can still find interesting results even though our Clustering Analysis does not work as expected. We successfully located one outlier paper “kdd15-p309” from 100 papers that we processed.

1. **Further Thoughts**

In the future, the following parts can be taken into consideration to improve our performance.

* 1. **Larger dataset**

More records in our datasets may give us a better and more satisfactory results. For example, Professor A may only have one publication among the 100 papers that we processed, but he may have 20 publications in KDD since 1989. If we perform Frequent Pattern Mining based on that data cube, we may find more interesting phenomenon.

* 1. **Different Algorithms**

There is no perfect algorithm that fits every dataset perfectly. Naïve Bayes has a “naïve” conditional-independence assumption while K-Means cannot help us find the non-convex clusters. There are many possible algorithms that can be attempted. For example, if we would like to build another classifier, we can try decision tree, support vector machine (SVM), ensemble methods, et cetera. At the same time, we should pay attention to the potential over-fitting issue.

* 1. **Further Outlier Analysis**

In the real world, just detecting the outliers from a dataset is not enough. Some of them were resulted from mistake like data entry errors. In that case, we can try to correct them by looking for the original resource. If that is not possible, deleting or down-weighting can be considered. If the outliers are resulted from real but special events, we should take a deeper look at them and make different decisions based on different situations.

**Reference:**

1. Jiawei Han, Micheline Kamber and Jian Pei, Data Mining: Concepts and Techniques (3rd ed), Morgan Kaufmann, 2011.
2. T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd ed., Springer, 2009.
3. Scikit-Learn. Retrieved 16:05, July 20, 2017, from <http://scikit-learn.org/stable/>.
4. TF-IDF. In Wikipedia, the Free Encyclopedia. Retrieved 17:00, July 20, 2017, from <https://en.wikipedia.org/wiki/Tf%E2%80%93idf>.
5. Singular value decomposition. In Wikipedia, the Free Encyclopedia. Retrieved 17:25, July 20, 2017 from <https://en.wikipedia.org/wiki/Singular_value_decomposition>.
6. K-means clustering. In Wikipedia, the Free Encyclopedia. Retrieved 20:05, July 25, 2017, from <https://en.wikipedia.org/wiki/K-means_clustering>.
7. DBSCAN. In Wikipedia, the Free Encyclopedia. Retrieved 21:32, July 25, 2017, from <https://en.wikipedia.org/wiki/DBSCAN>.