Practice of Classification models on TIP data
(treatise on invertebrate paleontology)

CHANG LIU FQ8596

11/19/2015

Outline

1. Introduction
   - Classification methods choosing
   - Original dataset
2. Preprocessing
   - Data extraction
   - Data clean
   - Feature selection
3. Classification
   - Parameter tuning
   - Results and comparison
4. Conclusion

Main Goal:

Using fossils description to identify unknown species. In another word, employ classification methods to classify new species.

1.1 Classification methods choosing

According to the requirement of project goal and the dataset. We should use multi-class classification. Thus, I choose Decision Tree and Random Forest. They are both inherently multiclass.

Advantages of Decision Tree :

- extremely fast at classifying unknown records
- able to handle both continuous and discrete attributes
- work well in the presence of redundant attributes
- robust to the effect of outliers
- robust in the presence of noise

Disadvantages:

- irrelevant attributes may affect badly the construction of a decision tree
- error-prone with too many classes

Advantages of Random Forest :

- produces a highly accurate classifier
- runs efficiently on large databases
- has methods for balancing error in class population unbalanced data sets
- can handle thousands of input variables without variable deletion
- can be extended to unlabeled data

Disadvantages:

- has been observed to overfit for some datasets with noisy classification tasks
- difficult for humans to interpret

[1] J.R. Quinlan. C4. 5: programs for machine learning. Morgan Kaufmann, 1993.

[2] L. Breiman, “Random Forests”, Machine Learning, 45(1), 5-32, 2001.

1.2 Original dataset

Two files : both have 4 ranks of taxonomy, but are different.

fs1_v2_379-429.pdf
(COELENTRATA supplement 1)

51 pages

fs1_v2_506-670.pdf
(systematic descriptions: TABULATA)

164 pages

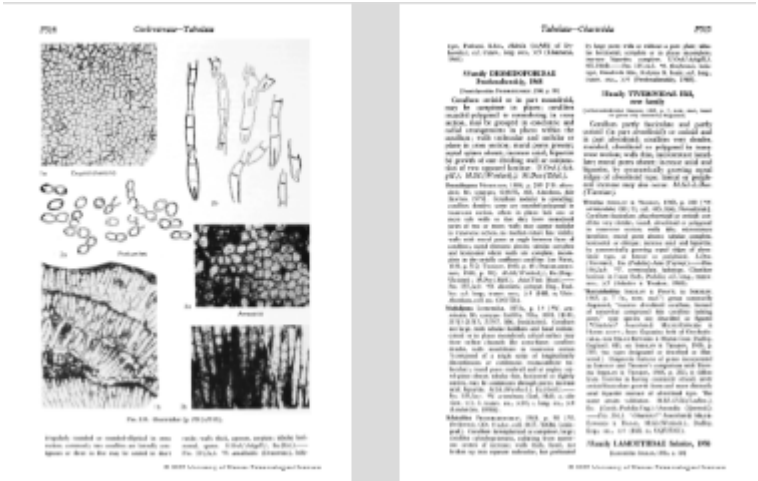
OCR

fs1_v2_379-429_p.txt

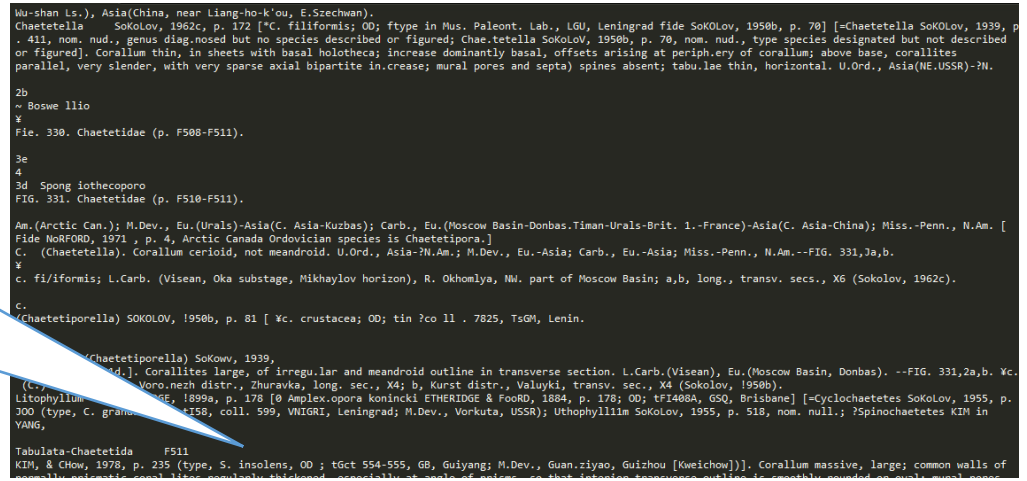
811 lines

fs1_v2_506-670_p.txt

2107 lines



Too many
noises!
errors!
Format loss!



Can not extract information we want directly by programming.

2.1 Preprocessing - Data extraction (1/2)

I. Manual processes:

For both files:

1. Change file encoding format to utf-8;
2. Delete noises generated from figures;
3. Add necessary line breaks which were missed by OCR software; (if not, some species will not be detected because its name was concatenated with previous species and treated as previous one's description)
4. Delete redundant line breaks to make sure each paragraph content is in the same line according to original document;
5. Modify critical information, such as species name, which were wrongly recognized by OCR software;
6. Add marks to form recognizable patterns as follows:

   Because we require 4 Ranks for both files, I add 4 different prefixed before each rank name like this:

Rank 1	Rank 2	Rank 3	Rank 4
"###"	"@@@"	"$$$"	"==="

After all the manual processes, the text file will be like this:

demo

###Suborder LITHOSTROTIONINA Spasskiy & Kachanov, 1971
[Lithostrotionina SPASSKIY & KACHANOV, 1971, p. 48)
Compound Stauriida; commonly with ax.ially somewhat thickened lathlike columella continuous in early stages with cardinal and counter septa but later commonly with coun.ter septum only; tabular floors conical; tabu.lae complete or incomplete; diphymorphs in which columella fails and tabulae flatten are common, some develop an aulos; dis.sepimentarium commonly normal, concen.tric with subglobose dissepiments, and minor septa commonly continuous longi.tudinally; cardinal fossula not distinct. Carb.-Perm.
@@@Family LITHOSTROTIONIDAE d'Orbigny, 1852
[Lithostrotionidae D'ORBIGNY, 1852, p. 184] [=NematophylÃ¡ linae McCov, 1851b, p. 33; Stylaxidae DE FROMENTEL, 1861, p. 74, 313 (nom. correct. HILL, herein, ex Stylaxinidae GERTH, 1921, p. 69, et Stylaxiniens DE FROMENTEL, 1861, p.74, 313); Diphyphyllidae DYBOWSKI, 1873c, nom. transl. GRABAU, 1936, p. 43, ex Diphyphyllinae DYBOWSKI, 1873c, p.332; Lithostrotiontidae SMITH, 1917, p. 294; LithostroÃ¡ tionicae, nom. transl. IVANOVSKIY, 1965a, p. 53]
Fasciculate or massive; commonly with axially somewhat thickened lathlike col.umella continuous in early stages with both cardinal and counter septa but in later stages with counter septum only; tabular floors conical, tabulae complete or incom.plete; dissepimentarium commonly of nor.mal concentric small plates with minor septa commonly longitudinally continuous; cardinal fossula not distinct. L.Carb.-L. Perm.
$$$Subfamily LITHOSTROTIONINAE d'Orbigny, 1852
[nom. transl. HILL, herein, ex Lithostrotionidae o'ORBIGNY, 1852, p. 184] [=Nematophyllinae McCOY, 185lb, p. 33; Stylaxidae DE FROMENTEL, 1861, p. 74, 313 (nom. co"ect. HILL, herein, ex Stylaxinidae GERTH, 1921, p. 69, et Stylaxiniens DE FÂ¥oMENTEL, 1861, p. 74, 313) J
Fasciculate or massive Lithostrotionidae with columella lenticular in section and commonly continuous with counter septum in late stages, with conical, complete or in.complete tabulae and with normal concen.tric dissepimentarium; diphymorphic coral.lites may occur; cardinal fossula indistinct. L.Carb.-? L.Perm.
===Lithostrotion FLEMING, 1828
, p. 508 ["L. striatum; SD ICZN, Opin. 117; t1870.14.370, missing from FLEMING Coll., RSM, Edinburgh; lectotype by ?THOMSON, 1887, p. 377 and KATO, 1971, p. 2; L.Carb., Brit. I.J [=Lithostrotium AGASSIZ, 1846, p. 214, nom. van., ICZN 1957, Dir. 76; Nema.phyllum McCoY, 1849, p. 15 (type, N. arach.noide11m, SD MILNE-EDWARDS & HAIME, 1850, p.lxxi; iÃ¡A2400, SM, Cambridge, lectotype by HUDSON, 1930, p. 97; L.Carb., Derbyshire); Sty./axis McCoY, 1849, p. 119 (type, S. fiemingi, SD LANG, SMITH, & THOMAS, 1940, p. 127; tA2051, SM, Cambridge, lectotype here chosen; L.Carb., Derbyshire); Lasmocyathtts D'0RBIGNY, 1849, p. 12 (type, Astraea aranea McCov, 1844, p. 187, M; t81.1925 and slides 50.1926

2.1 Preprocessing - Data extraction (2/2)

II. Programming processes:

For both files: Write a python program to extract the data. (see parse.py)

the result table contains 20 columns:

count	rank-1 label	rank-1 name	rank-1 author	rank-1 date	rank-1 description	rank-2 label	rank-2 name	rank-2 author	rank-2 date	rank-2 description	rank-3 label	rank-3 name	rank-3 author	rank-3 date	rank-3 description	rank-4 name (Species)	rank-4 author	rank-4 date	rank-4 description

III. Counts:

For File 1:

1 Rank label	no. of 1 Rank	no. of Family	no. of Subfamily	no. of Species
?Order	1	1	0	6
Suborder	2	8	10	122
Order and Family Uncertain	1	0	0	20
Subclass Uncertain	1	0	0	3
Doubtful Genera	1	0	0	3
Nomina Nuda	1	0	0	37
Unavailable Genus-Group	1	0	0	56
Total	8	9	10	247

Didn't miss any species!

For File 2:

1 Rank label	no. of 1Rank	2 Rank label	no. of 2Rank	no. of Subfamily	no. of Species
?Order	1	Family	3	3	24
		?Family	4	0	7
Order	2	Family	5	5	35
Suborder	3	Superfamily	1	0	7
		Family	40	12	257
		?Familly	5	0	11
Order Superfamily and Family Uncertain	1	-	0	0	2
Unrecognizable Genera	1	-	0	0	17
Nomina Nuda	1	-	0	0	10
Taxa Probably Neither Rugosa nor Tabulata	1	-	0	0	19
Total	10		58	20	389

2.2 Preprocessing - Data clean

1. Remove all species whose rank-1 label are

For file 1, "Subclass Uncertain" / "Doubtful Genera" / "Nomina Nuda" / "Unavailable Genus-Group Names“

For file 2, "Unrecognizable_Genera" / "Taxa_Probably_nor_Tabulata" / "Nomina Nuda"

Because their species description are nearly nothing or meaningless.

The total number of species left is file 1: 148 file 2: 343. (original training dataset)

2. Collect testing data

Step1: randomly choose 67 species name from file 1; and choose 145 species name from file 2.

Step2: search descriptions of these species from google or website: http://fossilworks.org, and save them in tables separately

For descriptions in both training data files and testing data files, (write python program)

3. Remove all punctuation.
4. Remove all numbers.
5. Remove all words whose length is less than 3.

2.3 Preprocessing - Feature selection

Write program feature3.py

1. Concatenate training data and testing data, and add a flag column to mark which sample is training data and which is testing data.
2. Apply TfidfVectorizer to get features:

   Code segment as following:

```
vectorizer = TfidfVectorizer(max_df=0.5, min_df=2, stop_words='english')
tfidf = vectorizer.fit_transform(datas)
tfidf_matrix=tfidf.toarray()
feature_names =vectorizer.get_feature_names()
```

   Feature selection Results: for file 1, I got 1329 features; for file 2, I got 2217 features;

3. Convert tfidf result sparse matrix to original matrix format, and save as following format:

flag	Species name	Feature 1	Featue 2	Feature 3	.....	Feature n

Save in files as input of classification.

3.1 Classification - Parameter tuning (1/2)

used Metrics are defined as follows:

```
precision = precision_score(Test_y, Pred_y, average='micro')
recall = recall_score(Test_y, Pred_y, average='micro')
F1 = f1_score(Test_y, Pred_y, average='micro')
```

Set average ='micro' means: Calculate metrics globally by counting the total true positives, false negatives and false positives. Because in our case, we only have 1 train data for each class (species), and at most 1 test data for each class. So using globally counting will make precision always equals to recall, thus f1 is also the same.

<DecisionTree>: write program DecisionTree.py

Parameters to be considered in sklearn.tree.DecisionTreeClassifier()

Criterion = 'gini‘ or 'entropy‘

Splitter = 'best‘ or ‘random’

Max_features = 5 or 10 or 100 or None or ‘auto’

Evaluation of classifier with different parameters combination (record best one of 10 times prediction)

Max_features=5	Criterion='gini'	Criterion='entropy'
Splitter='best'	File 1 Precision: 0.04477 recall: 0.04477 f1: 0.04477 File 2 Precision: 0.03448 recall: 0.03448 f 1: 0.03448	File 1 Precision: 0.02985 recall: 0.02985 f1: 0.02985 File 2 Precision: 0.01379 recall: 0.01379 f1: 0.01379
Splitter='random'	File 1 Precision: 0.07463 recall: 0.07463 f1: 0.07463 File 2 Precision: 0.04827 recall: 0.04827 f1: 0.04827	File 1 Precision: 0.05970 recall: 0.05970 f1: 0.05970 File 2 Precision: 0.02069 recall: 0.02069 f1: 0.02069

Max_features=10	Criterion='gini'	Criterion='entropy'
Splitter='best'	File 1 Precision: 0.05970 recall: 0.05970 f1: 0.05970 File 2 Precision: 0.02069 recall: 0.02069 f1: 0.02069	File 1 Precision: 0.01493 recall: 0.01493 f1: 0.01493 File 2 Precision: 0.00690 recall: 0.00690 f1: 0.00690
Splitter='random'	File 1 Precision: 0.05970 recall: 0.05970 f1: 0.05970 File 2 Precision: 0.02759 recall: 0.02759 f1: 0.02759	File 1 Precision: 0.04478 recall: 0.04478 f1: 0.04478 File 2 Precision: 0.02069 recall: 0.02069 f1: 0.02069

Max_features=100	Criterion='gini'	Criterion='entropy'
Splitter='best'	File 1 Precision: 0.02985 recall: 0.02985 f1: 0.02985 File 2 Precision: 0.02069 recall: 0.02069 f1: 0.02069	File 1 Precision: 0.01493 recall: 0.01493 f1: 0.01493 File 2 Precision: 0.00690 recall: 0.00690 f1: 0.00690
Splitter='random'	File 1 Precision: 0.07463 recall: 0.07463 f1: 0.07463 File 2 Precision: 0.02759 recall: 0.02759 f1: 0.02759	File 1 Precision: 0.01493 recall: 0.01493 f1: 0.01493 File 2 Precision: 0.00690 recall: 0.00690 f1: 0.00690

From above results, Splitter='random‘ and Criterion='gini‘ is always better.

	Max_features=None (n_features)	Max_features=auto (sqrt(n_features))
Splitter='random' Criterion='gini'	File 1 Precision: **0.07463** recall: 0.07463 f1: 0.07463 File 2 Precision: **0.07586** recall: 0.07586 f1: 0.07586	File 1 Precision: 0.04478 recall: 0.04478 f1: 0.04478 File 2 Precision: 0.02759 recall: 0.02759 f1: 0.02759

For both files, the best combination of parameter setting is as following:

Criterion = 'gini': use Gini impurity function to measure the quality of a split.

Splitter = 'random': choose the best random split at each node.

Max_features = None: consider the number of all features at each split when looking for the best split.

3.1 Classification - Parameter tuning (2/2)

<RandomForest>: write program RandomForest.py

Parameters to be considered in sklearn.ensemble.RandomForestClassifier()

Criterion = 'gini‘ or 'entropy‘

N_estimators = 5 or 10 or 100

Max_features = 5 or 10 or 100 or None or ‘auto’

Evaluation of classifier with different parameters combination (record best one of 10 times prediction)

Max_features=5	Criterion='gini'	Criterion='entropy'
N_estimators=5	File 1 Precision: 0.05970 recall: 0.05970 f1: 0.05970 File 2 Precision: 0.02069 recall: 0.02069 f1: 0.02069	File 1 Precision: 0.04478 recall: 0.04478 f1: 0.04478 File 2 Precision: 0.02069 recall: 0.02069 f1: 0.02069
N_estimators=10	File 1 Precision: 0.07463 recall: 0.07463 f1: 0.07463 File 2 Precision: 0.02758 recall: 0.02758 f1: 0.02758	File 1 Precision: 0.07463 recall: 0.07463 f1: 0.07463 File 2 Precision: 0.00690 recall: 0.00690 f1: 0.00690
N_estimators=100	File 1 Precision: 0.05970 recall: 0.05970 f1: 0.05970 File 2 Precision: 0.0 recall: 0.0 f1: 0.0	File 1 Precision: 0.04478 recall: 0.04478 f1: 0.04478 File 2 Precision: 0.0 recall: 0.0 f1: 0.0

Max_features=10	Criterion='gini'	Criterion='entropy'
N_estimators=5	File 1 Precision: 0.05970 recall: 0.05970 f1: 0.05970 File 2 Precision: 0.02069 recall: 0.02069 f1: 0.02069	File 1 Precision: 0.04478 recall: 0.04478 f1: 0.04478 File 2 Precision: 0.01379 recall: 0.01379 f1: 0.01379
N_estimators=10	File 1 Precision: 0.08955 recall: 0.08955 f1: 0.08955 File 2 Precision: 0.02069 recall: 0.02069 f1: 0.02069	File 1 Precision: 0.04478 recall: 0.04478 f1: 0.04478 File 2 Precision: 0.00690 recall: 0.00690 f1: 0.00690
N_estimators=100	File 1 Precision: 0.05970 recall: 0.05970 f1: 0.05970 File 2 Precision: 0.0 recall: 0.0 f1: 0.0	File 1 Precision: 0.02985 recall: 0.02985 f1: 0.02985 File 2 Precision: 0.0 recall: 0.0 f1: 0.0

Max_features=100	Criterion='gini'	Criterion='entropy'
N_estimators=5	File 1 Precision: 0.07463 recall: 0.07463 f1: 0.07463 File 2 Precision: 0.04138 recall: 0.04138 f1: 0.04138	File 1 Precision: 0.01493 recall: 0.01493 f1: 0.01493 File 2 Precision: 0.00690 recall: 0.00690 f1: 0.00690
N_estimators=10	File 1 Precision: 0.10448 recall: 0.10448 f1: 0.10448 File 2 Precision: 0.02759 recall: 0.02759 f1: 0.02759	File 1 Precision: 0.02985 recall: 0.02985 f1: 0.02985 File 2 Precision: 0.00690 recall: 0.00690 f1: 0.00690
N_estimators=100	File 1 Precision: 0.10448 recall: 0.10448 f1: 0.10448 File 2 Precision: 0.01379 recall: 0.01379 f1: 0.01379	File 1 Precision: 0.01493 recall: 0.01493 f1: 0.01493 File 2 Precision: 0.0 recall: 0.0 f1: 0.0

From above results, N_estimators=10 and Criterion='gini‘ is always better.

	Max_features=None (n_features)	Max_features=auto (sqrt(n_features))
N_estimators=10 Criterion='gini'	File 1 Precision: 0.10448 recall: 0.10448 f1: 0.10448 File 2 Precision: 0.02759 recall: 0.02759 f1: 0.02759	File 1 Precision: **0.14925** recall: 0.14925 f1: 0.14925 File 2 Precision: **0.02759** recall: 0.02759 f1: 0.02759

For both files, the best combination of parameter setting is as following:

Criterion = 'gini': use Gini impurity function to measure the quality of a split.

N-estimators = 10: The number of trees in the forest is 10.

Max_features = auto: consider square root of all features at each split when looking for the best split.

3.2 Classification - Results and comparison (1/2)

In brief, the best performance of 2 models by using the best combination of parameter setting is

Using cleaned description, file 1: 1329 features; for file 2 : 2217 features;

Better!

	DecisionTree		RandomForest
criterion='gini' splitter='random' max_features=None	File 1 Precision: **0.07463** File 2 Precision: **0.07586**	criterion='gini' n-estimators= 10 max_features='auto'	File 1 Precision: **0.14925** File 2 Precision: **0.02759**

➢ Compare with using Original description (without any preprocessing——non-cleaned)

Modify the programs of cleaning data and feature selection. Then

For file 1, I got 1651 features; for file 2, I got 2855 features;

	DecisionTree		RandomForest
criterion='gini' splitter='random' max_features=None	File 1 Precision: 0.05970 File 2 Precision: 0.03448	criterion='gini' n-estimators= 10 max_features='auto'	File 1 Precision: 0.07463 File 2 Precision: 0.03448

3.2 Classification - Results and comparison (2/2)

In brief, the best performance of 2 models by using the best combination of parameter setting is

Using cleaned description, file 1: 1329 features; for file 2 : 2217 features;

	DecisionTree		RandomForest
criterion='gini' splitter='random' max_features=None	File 1 Precision: **0.07463** File 2 Precision: **0.07586**	criterion='gini' n-estimators= 10 max_features='auto'	File 1 Precision: **0.14925** File 2 Precision: **0.02759**

Better!

➢ Compare with Using over-cleaned description (remove words whose length >=3)

Modify the programs of cleaning data and feature selection. Then

For file 1, I got 1064 features; for file 2, I got 1746 features;

	DecisionTree		RandomForest
criterion='gini' splitter='random' max_features=None	File 1 Precision: 0.04478 File 2 Precision: 0.04138	criterion='gini' n-estimators= 10 max_features='auto'	File 1 Precision: 0.08955 File 2 Precision: 0.02759

4. Conclusion

Why is the accuracy so low?

- The classifiers are trained on original words. Besides removing stop-words, we didn't use other useful NLP methods. If we used stemming, it may improve the results.
- The cleaned data still include noise that needs to be removed, like what were recognized wrongly and there are too many meaningless citation content in the description etc. They all treats as part of the actual description by the classifiers.
- The testing dataset was not from the same distribution that the training dataset extracted.
- Too many classes and too little samples for each class.

4. Conclusion

Why is the accuracy so low?

- The classifiers are trained on original words. Besides removing stop-words, we didn't use other useful NLP methods. If we used stem this would improve the results.
- The cleaned data still include noise that needs to be removed, like what were recognized wrongly and there are too many meaningless citation content in the description etc. They all treats as part of the actual description by the classifiers.
- The testing dataset was not from the same distribution that the training dataset extracted.
- Too many classes and too little samples for each class.

Thank you !