UNIVERSITAT ROVIRA I VIRGILI
Department of Computer Engineering and Mathematics
July 27, 2021

CN-MAI Project

Complex Networks
Spring 2021, 5 Credits

Structural Descriptors In Language Classification Models

Name Bartosz Paulewicz
Betty Törnkvist

E-mail bartosz.paulewicz@student.put.poznan.pl
betty.tornkvist@gmail.com

Teacher
Alejandro Arenas Moreno

Contents

1 Introduction

There are different ways to build classifiers that accurately predicts and reflects the incoming data. In this task, we explore the possibility of combining machine-learning methods with complex networks measurements. More specifically, the posed hypothesis is to find out weather or not the structural descriptors retrieved from a network can be used as input to train an accurate language classifier.

2 Method

This section covers the following standard steps necessary to build and evaluate the language classifier. The program is implemented in Python, utilising the library igraph for network-related calls. A simplified overview of the process can be seen in Figure 1.

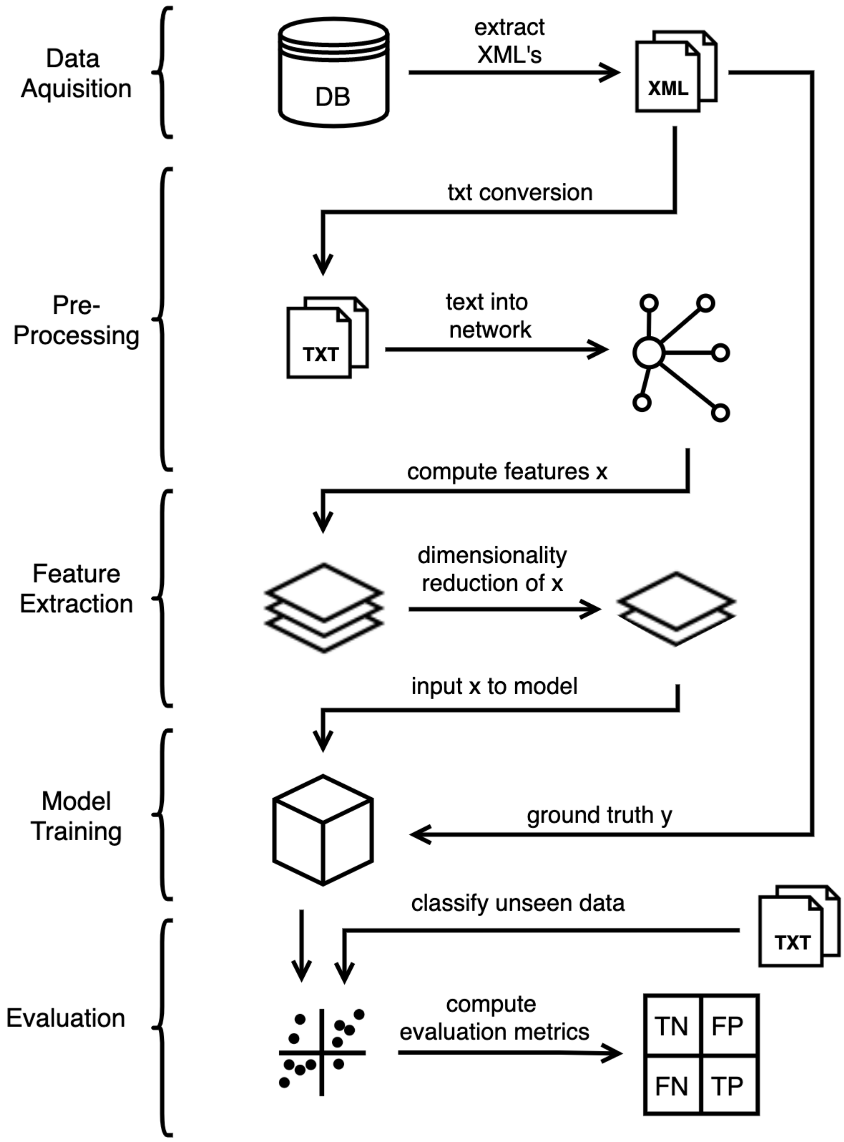


Figure 1: High-level architecture overview.

2.1 Data Acquisition

Data from different language groups were needed to generate networks. The site christos-c.com freely provides a multilingual parallel corpus created from translations of the Bible. This website effort to create a parallel corpus containing as many languages as possible that could be used for a number of NLP tasks. Using the Book, Chapter and Verse indices the corpus is aligned at a sentence level.

The dataset contains the Bible in 100 different languages, along with information about each language. Each Bible is available in XML-format, where each verse is in a separate tag.

To narrow it down and make the classification a bit more challenging, only European languages where collected from the data. For example, comparing European languages to Asian languages would be too easy due to their differences.

The New Testament was the only book available in all European languages and therefore it was chosen as the main data. Even if each language could be represented by different books, the choice of using the same source of data is made for comparative reasons.

2.2 Pre-Processing

The pre-processing covers the following steps (in order):

1. **Parsing and format conversion**
   Each XML-file is parsed and the verses are extracted and appended into a plain text file, one verse per row.
2. **Removing characters**
   Characters such as numbers and punctuation marks are removed to clean the text from unnecessary information.
3. **Lower-casing**
   Lower-casing of all text.
4. **Words into nodes**
   Mapping words to nodes, represented as integer identifiers.
5. **Relation to edges**
   For each word represented as a node, an edge is created between the word and its consecutive word.
6. **Language family and sub-family** The genus (language family: Germanic, Italic and Slavic) and sub-genus (sub-family: Italic-Romance, Germanic-West, Slavic-South, Germanic-North, Slavic-West, Slavic-East) are extracted and represented as integer identifiers. These two families will be the prediction goals of the classifications.

2.3 Feature Extraction

The following scalar metrics (single value) were extracted as features:

- Vertices count - the total number of nodes in the network.
- Edges count - the total number of edges in the network.
- Density - the amount of actual connections divided by the potential connections. Potential connections are calculated by $\frac{n*(n-1)}{2}$, where n represents the total number of nodes.
- Transitivity - The number of triangles in the network divided by the number of connected triples in the network.[1] A triangle is made when node x is connected to node y and z, and

y and z are also connected. A transitivity score of 1 implies that the network has all edges possible.

- Assortativity degree - Measures the similarities between nodes in terms of their degree, if high degree nodes connect to other high degree nodes, and vice versa for low degree nodes.
- Local transitivity average - The transitivity average per node.

Additionally, the following average metrics where calculated:

- Degree - Average number of edges per node.
- Page rank - Average of subsets of edges that contains at minimum one edge of every cycle in the network.
- Coreness - Average coreness over all nodes. The k-core is the maximal sub-graph where the degree of the nodes is $\geq$ k. If the node belongs to k and not k+1, the nodes coreness is k.[2]
- Hub score - The hub score is calculated by turning the network into an adjacency matrix A, and retrieving the principal eigenvector of A*t(A). [3]
- Constraint - Average of Burt's Constraint that measures how connected a nodes neighbour are towards the nodes other neighbours. [4]
- Feedback arc set - The minimal set of nodes that makes the network acyclic when removed. The sum of nodes is then averaged.

Following the feature extraction, supervised dimensionality reduction was applied. Dimensionality reduction is the transformation of data from high into low dimension space trying to preserve original data properties. Working with high dimensional data can be computationally difficult. Dimensionality reduction is popular in all the domains dealing with large numbers of observations and/or large numbers of variables and can be used for noise reduction, data visualisation, cluster analysis, or as an intermediate step to facilitate other analyses as it was used in our case.

The first method of dimensionality reduction we used is Linear Discriminant Analysis (LDA), which tries to identify attributes that account for the most variance between classes. In particular, LDA, in contrast to most well known methods like PCA, is a supervised method, using known class labels. It can be used as a classifier, or as in this case to reduce the dimensions of the input features. Eigenvalue decomposition is used as a solver in combination with automatic shrinkage, which is using the Ledoit-Wolf lemma. [5]

Another method we used is Neighbourhood Components Analysis (NCA), which tries to find a feature space such that a stochastic nearest neighbour algorithm will give the best accuracy. Like LDA, it is a supervised method.

2.4 Training

The data is divided by reduced feature input x, (one per language, the length of x is 22) and ground truth labels y, where y is set as either genus or sub-genus. We therefore train two models, one for genus and the other for sub-genus.

Leave One Out Cross Validation (LOOCV) is used to split the data into train and test during training. It is equivalent to K-fold Cross Validation with number of iterations k, is set to n, the total number of samples. For each iteration, one sample is used as test and the rest as train, until all samples has been used as test. Since our sample size is small (22), LOOCV is a good choice. For larger sample sets, this method can take much longer to run.

For the genus prediction, there are 3 classes: Germanic, Italic and Slavic. The dataset is slightly imbalanced, where Slavic has 9 samples, Germanic has 8 samples and Italic has 5 samples. We apply a Synthetic Minority Oversampling Technique (SMOTE), which synthesises samples of the minority classes. This results in a more balanced dataset, where each class has 9 samples and the sample size is increased to 27. Instead of using the LDA as a dimension reduction, it is applied as a classifier. Three other classifiers where also tried, Support-Vector Machine with a linear kernel (SVM), Logistic Regression (LR) and a supervised neural network: Multi-layer Perceptron (MLP).

For the sub-genus prediction, there are 6 classes: Italic-Romance, Germanic-West, Slavic-South, Germanic-North, Slavic-West and Slavic-East. Evening out the classes with SMOTE does not work well when a class contains few classes. In our case, the class Slavic-East only has 2 samples, which makes it unsuitable for SMOTE. Instead, the results are passed from the LDA dimensionality reduction to four kinds of classifiers: ExtraTreesClassifier(ETC), SVM, LR and MLP.

2.5 Evaluation

Evaluation is done by calculating the average accuracy over folds. Since the task is multi-class, the F1 score i calculated with a macro-average, meaning that each labels metric is calculated and the unweighted mean is returned. A confusion matrix is generated for each model, which reveals the true positives, true negatives, false positives and false negatives. In a optimal confusion matrix the descending diagonal is filled and the rest of is zero. This means that the model correctly classified all samples, true positives and true negatives.

Additionally, all models where run with grid search (in combination with LOOCV), to find the optimal hyper-parameters that maximises performance.

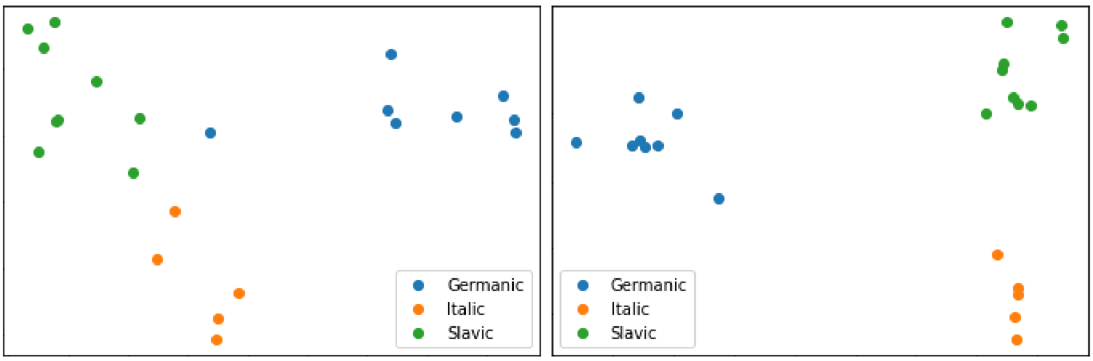
3 Result

3.1 Feature Extraction

The extracted structural descriptors can be seen in Appendix A.

3.2 Dimensionality Reduction

The results of the dimensionality reduction can be observed in Figure 2. We can see that both methods enforce a clustering of the data that is visually meaningful despite the large reduction in dimension.



(a) LDA (b) NCA

Figure 2: Data after applying dimensionality reduction

3.3 Hyper-parameters

The hyper-parameter search for each genus model can be seen in Figure 3, and the same for sub-genus models in Figure 4.

```
***************
SVC() -MODEL
Best score:
0.7727272727272727
Best parameters set:
{'C': 1, 'gamma': 1, 'kernel': 'linear'}
***************
***************
LogisticRegression() -MODEL
Best score:
0.9090909090909091
Best parameters set:
{'C': 100, 'intercept_scaling': 3, 'max_iter': 100, 'penalty': 'l1', 'solver': 'liblinear', 'tol': 0.0002}
***************
***************
MLPClassifier(max_iter=100) -MODEL
Best score:
0.8636363636363636
Best parameters set:
{'activation': 'relu', 'hidden_layer_sizes': (10,), 'learning_rate_init': 0.05, 'random_state': 0, 'solver': 'adam'}
***************
***************
LinearDiscriminantAnalysis() -MODEL
Best score:
0.9090909090909091
Best parameters set:
{'shrinkage': 0, 'solver': 'lsqr', 'tol': 0.0001}
***************
```

Figure 3: Hyperparameter tuning for genus models. SMOTE excluded.

```
***************
ExtraTreesClassifier() -MODEL
Best score:
0.5
Best parameters set:
{'max_depth': 32, 'n_estimators': 360, 'random_state': 0}
***************
***************
SVC() -MODEL
Best score:
0.4090909090909091
Best parameters set:
{'C': 1, 'gamma': 1, 'kernel': 'linear'}
***************
***************
LogisticRegression() -MODEL
Best score:
0.6818181818181818
Best parameters set:
{'C': 1, 'penalty': 'l2'}
***************
***************
MLPClassifier(alpha=0.001, max_iter=100, random_state=1) -MODEL
Best score:
0.36363636363636365
Best parameters set:
{'activation': 'tanh', 'hidden_layer_sizes': (6,), 'learning_rate_init': 0.001, 'solver': 'adam'}
***************
```

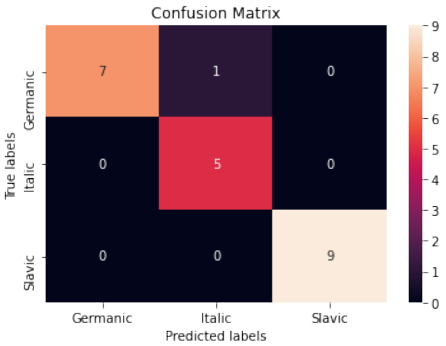
Figure 4: Hyperparameter tuning for sub-genus models. LDA dimensionality reduction excluded.

3.4 Model Evaluation Metrics

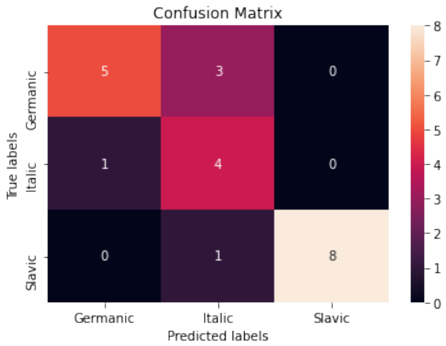
The results in terms of accuracy,f1 and confusion matrix for genus can be observed in Table 4 and Figure 5. The same for sub-genus can be seen in Table 3 and Figure 6.

Genus		
Model	**Accuracy**	**F1(macro**
SVM	0.77	0.75
LDA	0.95	0.95
LR	0.86	0.84
MLP	0.55	0.48

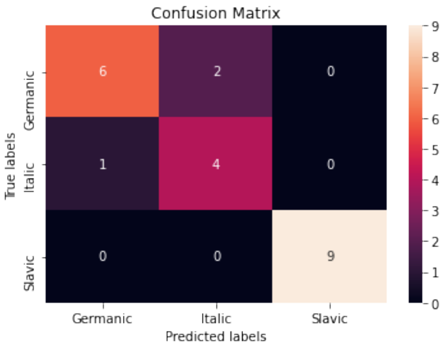
Table 1: Accuracy and F1 for the different genus prediction models.



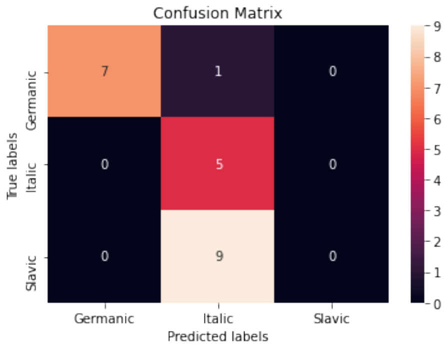
(a) LDA.



(b) SVM.

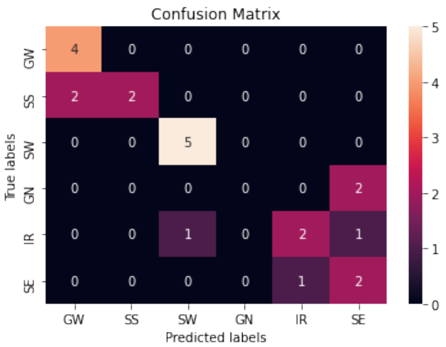


(c) LR.

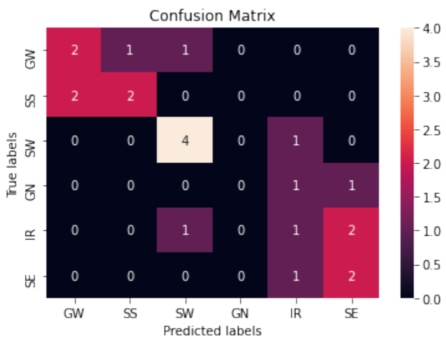


(d) MLP.

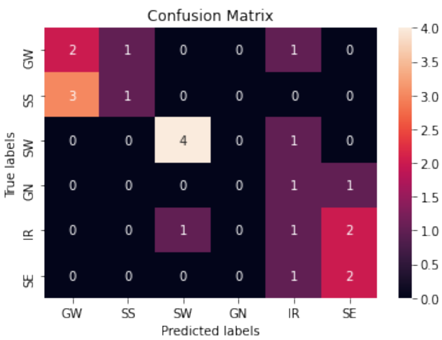
Figure 5: Confusion matrices for the different genus classification models (grid search not applied).



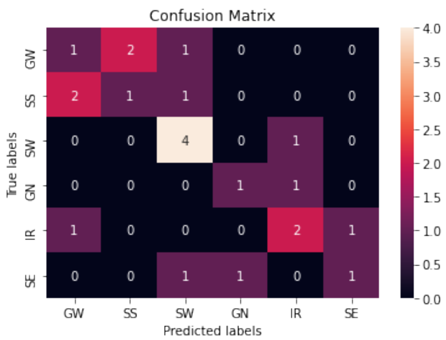
(a) ETC.



(b) SVM.



(c) LR.



(d) MLP.

Figure 6: Confusion matrices for the different sub-genus classification models (grid search not applied).

Genus - Grid Search	
Model	**Accuracy**
SVM	0.77
LDA	0.90
LR	0.90
MLP	0.77

Table 2: Grid search accuracy for the different genus prediction models. SMOTE was not applicable.

Sub-Genus		
Model	**Accuracy**	**F1(macro**
SVM	0.50	0.42
ETC	0.68	0.57
LR	0.45	0.38
MLP	0.45	0.43

Table 3: Accuracy and F1 for the different sub-genus prediction models. LDA dimensionality reduction applied.

Sub-Genus Grid Search	
Model	**Accuracy**
SVM	0.41
ETC	0.50
LR	0.68
MLP	0.36

Table 4: Grid search accuracy for the different sub-genus prediction models. LDA dimensionality reduction not applied.

3.5 Explainability

Tests where applied to the best-performing sub-genus predictor (ETC) to provide some level of explainability for the feature importance. The library SHAP (SHapley Additive exPlanations) was used to show feature importance, averaged over folds. The result can be seen in Figure 7.

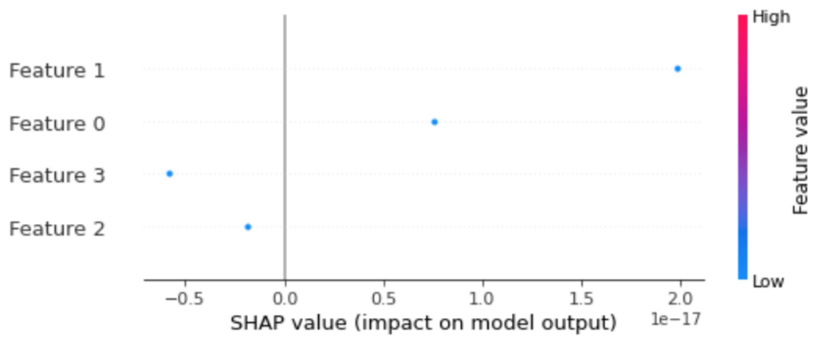


Figure 7: Feature impact for sub-genus classification.

4 Conclusion

A binary task with a classifier that has an accuracy of 0.5 implies that the classification is by chance, and gets it right half of the time. In this case where the task is multi-classification, the same accuracy from classification by chance (with an even sample frequency) would be 1/number_of_families. For genus that is evened out by SMOTE and therefore has an even sample frequency, that is $1/3 \approx 0.33$. For sub-genus, that is $1/6 \approx 0.16$.

However, since SMOTE was not applicable for the sub-genus classification, the classification by chance would instead need to take the sample frequency in regards. The random accuracy could instead be calculated from:

$$\frac{\sum_{n=1}^{len(C)} c_n^2}{(\sum_{n=1}^{len(C)} c_n)^2}$$

where C is the list of classes and c is the sample size per class. For sub-genus, this amounts to:

$$\frac{5^2 + 4^2 + 4^2 + 4^2 + 3^2 + 2^2}{(5 + 4 + 4 + 4 + 3 + 2)^2} \approx 0.33$$

Taking that into consideration, the results are fairly good for all models, especially the genus LDA with and accuracy of 95%. Observing the corresponding confusion matrix we can see that only one sample was miss-classified (Icelandic language predicted as Italic when actually it's Germanic). when comparing to the grid search results, we can observe an unexpected difference in accuracy for the LDA model, which dropped the accuracy by circa 4.5%. The grid search passes the same input as the parameters for the model used without grid searh, which implies that the discarding of the SMOTE-method was the reason for the accuracy drop. The other genus models benefited from the grid search, especially the MLP which increased its accuracy by approximately 22%.

For the sub-genus classification, the ETC ranks highest with 0.68%. Considering the increase in classes and that the random guess would be 18%, this is a good result. The grid search decreases the results (except for LR, which increases by 23%), which at first glance can be read as that the hyper-parameter space should be further expanded. However, doing the grid search in combination with LOOCV and dimensionality reduction proved to be a bit tricky, so the LDA was discarded in this step. Since the space of hyper-parameters covers most cases of the initial model setup, the accuracy drop is therefore probably due to the discarding of LDA.

In contrast to neural networks, the advantage of the used models is that they offer higher level of explainability. In general, it can often be a trade-off between accuracy and explainability, but in our case the less complex algorithms are as good as, or outperforms the multi-perceptron and still acquires a high accuracy. The feature importance shown in Figure 7 shows that feature 1 is the one with the highest impact, and it stands for a combination of *all* features. However, since the LDA feature reducation combines the original 22 features by weight, features 0, 2 and 3 would need further dissection to reveal which combination and weight they consist of.

4.1 Improvements

Possible improvement and future work could be:

- **Different dataset**
  Our bible-dataset did
- **Further preprocessing**
  nltk has different language packages for preprocessing which could be applied. However, in our case the language expressed in the bibles can be out-dated and not well-processed by the nltk packages.
- **Expand Features**
  Adding additional network descriptors as features, or combining with non-network descriptors as features. Possible additional descriptors are for example: eigenvector centrality, similarity jaccard, similarity dice, etc.
- **Feature combinations**
  Additional exploration of the feature-spece, like using different setups for comparison or finding best feature combination.

- **Additional fine-tuning of hyper-parameter**
  The grid-search search-space could be expanded to include further hyper-parameters and/or increased iterations. As noted before, some models performed worse during grid search. The SMOTE and LDA could be included to make sure that they do not negatively impact the results.

- **Additional explainability**
  There are more explainability-modules available[6], which could be applied to achieve a better understanding of the resulting models.

References

[1] M. Insight. Definition of the transitivity of a graph. [Online]. Available: https://mathinsight.org/definition/transitivity_graph#:~:text=The%20transitivity%20T%20of%20a,of%20nodes%20in%20the%20network.&text=With%20this%20definition%2C%200%E2%89%A4,network%20contains%20all%20possible%20edges.

[2] G. Csardi. K-core decomposition of graphs. [Online]. Available: https://search.r-project.org/CRAN/refmans/igraph/html/coreness.html

[3] igraph. Kleinberg's hub centrality scores. [Online]. Available: https://igraph.org/r/doc/hub_score.html

[4] CentiServer. Burt's constraint. [Online]. Available: https://www.centiserver.org/centrality/Burts_constraint/

[5] S. Learn. Linear discriminant analysis. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html

[6] J. T. R. bc1faced. Explainability. [Online]. Available: https://python-data-science.readthedocs.io/en/latest/explainability.html

A Feature Extraction Tables

	genus	subgenus	vcount	ecount	assortativity_degree	density	transitivity	transitivity_avglocal
Afrikaans	Germanic	West	3666	23932	-0.2755773763628665	0.0035623989181215	0.069498576112974	0.6015039959296217
Bulgarian	Slavic	South	7154	32323	-0.2089696684699594	0.0012632941848797	0.0271093617649419	0.4334237563156602
Croatian	Slavic	South	8101	35729	-0.1765397700425452	0.0010889983099175	0.023219106222208	0.3763857021342764
Czech	Slavic	West	8271	36751	-0.1334792800952921	0.0010745722624335	0.0241423831957024	0.335850596211073
Danish	Germanic	North	4594	27546	-0.2134088437868611	0.0026109653149949	0.055719234479833	0.5108654932169968
Dutch	Germanic	West	4829	34136	-0.2765899897861457	0.0029283174716136	0.0856609591123344	0.5178419183754693
English	Germanic	West	3482	25302	-0.2527401388525258	0.0041749574823267	0.0690112950854975	0.5906526015354577
French	Italic	Romance	5927	30901	-0.2742560079638902	0.0017595675954168	0.0365673018897095	0.4433567539367139
German	Germanic	West	5010	31101	-0.1918779666246614	0.0024786521984977	0.056983551304207	0.5254476745013541
Icelandic	Germanic	North	6505	31846	-0.196230232431854	0.0015054178212804	0.03737963144926	0.4416838174414337
Italian	Italic	Romance	6688	33788	-0.2284955649704571	0.0015110014932923	0.0350790868975916	0.3783445366081213
Norwegian	Germanic	North	4085	25959	-0.2335429806612185	0.0031120040951523	0.0628242406401763	0.5525480963924013
Polish	Slavic	West	8080	35233	-0.1732164326956338	0.0010794701824434	0.0248489477190727	0.3543619970776914
Portuguese	Italic	Romance	6573	32153	-0.2187052615159291	0.0014886421415038	0.0305757642855109	0.4400644942932387
Romanian	Italic	Romance	5450	31317	-0.2193157922645498	0.0021090983784584	0.0423107104995289	0.3803186351252302
Russian	Slavic	East	8510	33252	-0.134777726149058	0.0009184165131576	0.0205215731506161	0.3404550647311334
Serbian	Slavic	South	6838	32031	-0.1555526843061344	0.0013702689497723	0.0242096481783243	0.4192029944765094
Slovak	Slavic	West	8087	34300	-0.1223090488457407	0.0010490662988797	0.0199655057488455	0.3712452608168997
Slovene	Slavic	South	7527	34300	-0.1447517844032474	0.0012109828304877	0.0253658294418369	0.37482055894186
Spanish	Italic	Romance	5920	29896	-0.2698851749351176	0.0017063693191417	0.0333392880651882	0.4723279483703249
Swedish	Germanic	North	5011	30240	-0.2205201050406074	0.0024090713006236	0.0598758026849948	0.4987419077784785
Ukranian	Slavic	East	8558	35746	-0.2125553532817458	0.0009762558123421	0.0278700939183599	0.3439895926469485

	degree	feedback_arc_set	hub_score	constraint	coreness	pagerank
Afrikaans	13.05619203491544	11960.508906103518	0.0345706620433027	0.3197100965299003	6.81505728314239	0.0002727768685215
Bulgarian	9.036343304445063	16146.117361938816	0.016667717354056	0.359154319105905	4.677103718199609	0.0001397819401733
Croatian	8.820886310332058	17882.19812515835	0.0155871504016302	0.3541611174380698	4.562029379089001	0.0001234415504258
Czech	8.886712610325233	18407.484182437416	0.0138620346734091	0.3558195954907297	4.590013299480111	0.0001209043646475
Danish	11.992163691771877	13790.436369973424	0.0261593653366819	0.3311637511000891	6.227252938615585	0.0002176752285589
Dutch	14.137916752950922	17071.28770301624	0.0344912008642289	0.3113907138661281	7.405052805963968	0.000207082211638
English	14.533026995979322	12660.134182668073	0.0350256157938089	0.2965852811809319	7.573520964962665	0.0002871912693854
French	10.427197570440358	15431.06994994995	0.0236955513228386	0.3440643703624197	5.38214948540577	0.0001687194196051
German	12.41556886227545	15516.61869538556	0.0249169695809535	0.3219160531568938	6.42814371257485	0.0001996007984031
Icelandic	9.791237509607994	15955.649790861022	0.0195219512492926	0.3506897913963428	5.044427363566487	0.0001537279016141
Italian	10.104066985645932	16875.161359359434	0.020154061400802	0.3481428076746228	5.215909090909091	0.0001495215311004
Norwegian	12.709424724602204	12978.263908571427	0.0297219589188134	0.3176442922278677	6.595348837209302	0.0002447980416156
Polish	8.721039603960396	17602.186123591368	0.0161622238850082	0.3600149266677613	4.509282178217822	0.0001237623762376
Portuguese	9.783356153963185	16068.726163949808	0.0190497002266904	0.3561221860798726	5.0470104974897305	0.0001521375323292
Romanian	11.492477064220184	15669.464744085357	0.0222739543742396	0.3173020759264707	5.951376146788991	0.0001834862385321
Russian	7.814806110458284	16646.334397607403	0.0124135155969773	0.3749523215172781	4.059459459459459	0.0001175088131609
Serbian	9.368528809593448	16049.23334921013	0.0158611409310615	0.3544151608373316	4.841620356829482	0.0001462415911085
Slovak	8.482750092741437	17122.562218661784	0.0125964336084099	0.3641910268759775	4.396191418325708	0.0001236552491653
Slovene	9.113856782250563	17173.10640920296	0.0144611021568132	0.3554451617949316	4.717815862893583	0.0001328550551348
Spanish	10.1	14936.946448679984	0.0224066990958055	0.3494738228578048	5.212162162162162	0.0001689189189189
Swedish	12.069447216124528	15142.688585017837	0.0258174648852146	0.3278838108451708	6.26781081620435	0.000199560965875
Ukranian	8.353820986211732	17846.28651292802	0.0165344551574882	0.3655847327903933	4.324842252862818	0.0001168497312456