Glossary

Training data - data from **train.txt**
Public test set - data from **test-public.txt**
True Edges - edges obtained from training data.
Fake Edges - edges that were randomly sampled to be negative examples.
Unknown Edges - edges from public test set that we had to predict to be true or fake.
Internal Edges - edges that connect two nodes both of which have descendants
Marginal Edges - edges for which sink node has no descendants (is a leaf node).

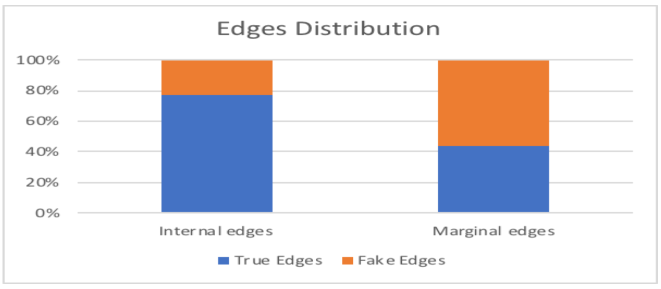
Analysis of Training and Test Data

Training data contained:

- 19570 unique source nodes that had descendant nodes and roughly 5 million leaf nodes
- Approximately 24 million edges, 2 million of which were internal edges and the rest 22 million marginal edges

Public test set contained:

- 369 internal edges, which according to our estimations had 77% true edges
- 1631 marginal edges, which according to our estimations had 44% true edges.



Training Set Generation and Fitting Models

The process comprised of the following stages:

- Sample 400k edges from training data (sample 200k true edges from training data; generate 200k fake edges)
- Generate edge-related features.
- Join precalculated nodes features for source and sink nodes.
- Fit models on prepared train set

For data manipulating and feature generation pandas and networkx packages were used.

Features Engineering

For all 5 million source and sink nodes the following **Basic features** were calculated:

- **children_counts** - the number of sink nodes that any source node has, 0 for leaf nodes
- **leaf_descendants_counts** - number of descendants that are leaf nodes
- **non_leaf_descendants_counts** - number of descendants that are non leaf nodes
- **parents_counts -** number of source nodes for any sink node

- **parents_leaf_descendants_count** - given any sink node, for all their parents we summed up all their leaf descendants
- **parents_non_leaf_descendants_count** - given any sink node, for all their parents we summed up all their non leaf descendants

For edges the following features were calculated

- Similarity Heuristics:
    - Adamic Adar Index
    - Jaccard's Coefficient
- Topology Related Features:
    - Triads - Triangular relationships between source & sink pairs - so as to provide a more substantial parameter for the classifier to learn on.
    - Common Neighbors - Between sink and source pair.
- Two principal components obtained from PCA

Sampling Fake and True Edges

1. **1st Sample Approach**
   Random sampling of true and fake edges.
    a. Sample size: 95000 fake edges + 95000 true edges
    b. Constraints: Source node should be non leaf node. Sink node should be leaf node.
    c. Description: Random sampling of true edges from the 24 million training dataset and random generation of source-sink pairs from the 4.7 million nodes and ensuring that they don't exist in the training data.
    d. Issues: All models yielded accuracy up to 98% on cross validation on training set and gave only 60% on Kaggle leaderboard. We were overfitting on this training set and regularisation didn't help.
2. **2nd Approach Sample**
   This was built keeping in consideration the need for diversity in our training dataset as the previous approach resulted in overfitting of models.
    a. Sample size: 190000 fake edges + 190000 true edges
    b. Aim: Build a more diverse model by sampling edges for all known source nodes
    c. Description: For each of 19570 source nodes sample 10 true and 10 fake sink nodes. The true edges were again sampled from the training dataset ensuring uniqueness.
    d. Issues: We faced the same issue with overfitting (accuracy was up to 95%). Some of our models gave 70-80% true predictions on public test set.
3. **3rd Approach Sample**
   Failing to increase accuracy of the predictions from our earlier model resulted us to increase features and include modely topology in feature selection.
    a. Sample Size: 60000 fake edges + 51000 true edges
    b. Constraints: Removing highly connected edges & corresponding nodes from the dataset from Approach 3, with the intent of fine tuning our model's distinguishing capability.
    c. Description: We selected 4 triads and included ratios of previously considered features and added another link-prediction heuristic to compile a list of 19 features segregated into 3 classes.

Comparison of Models

Sampling	Features	Train Set Cross Validation Score				Kaggle score				Advantages & Disadvantages
		Logistic Reg	RF	KNN	NN	Logistic Reg	RF	KNN	NN	
Approach 1	Basic	96%	93%	98%	88%	56%	58%	71%	--	Intrinsic features with low prediction power.
Approach 2	Basic + Heuristic	98%	96%	94%	93%	83%	80%	76%	80%	Heuristic value as a feature boosted the prediction accuracy
Approach 3	Basic + Heuristic + Topology	98%	97%	98%	91%	68%	56%	68%	56%	This approach was assumed to be the best performing, but the test data-set had source nodes with comparatively low connectivity making the deep topological features just noise.

Table 1: Model-specific performance metrics

Model	Configuration	Pros	Cons
Logistic Regression	- L2 penalty with stochastic gradient descent - Penalty factor C=11 - Tolerance = 0.000001	- Model learns faster in comparison with all other methods - Simple model to implement without throwing away too many features. - With same features even with lesser number of data points gave better performance than other models.	- Features like childred_count need scaling - Model was performing too well on the training set even after much tuning.
KNN	- Tried for both scaled feature set and non-scaled data. - On cross validation, loss stabilized at K=4,5 for Approaches 1+2, 3 respectively.	- Model implementation was simple and straightforward. - The model takes comparatively less time to train.	- Need to select number of neighbour hyperparameter - Dependent on training set contain similar objects to new objects. - Overfit on the training data generated by us.
Neural Network	- Multi Layer Perceptron model. - Supervised classifier. - 3 Hidden layers with 6 nodes each. - Two output layers. - Relu activation function - Stochastic gradient descent with adam solver	- This model was not overfitting on the training data set as compared to other models. - early stopping was enabled to stop training if the cross validation score was not improving.	- Test data set accuracy was low compared to all other models. - Tuning the model and topology was difficult. - Model is very sensitive to small changes in the data set. - Fits slowly in comparison to other methods

Random Forest (RF)	- 22 Trees - max_depth up to 5 - ‘gini’ impurity - minimum impurity decrease of *0.0001*	- Implemented grid search to iteratively find the best parameters. - Tuning the model and finding the hypermaters was easy.	- Number of true edge predictions were always high. - Test data accuracy was low in all three approaches. - Even after much pruning it overfits the train data set.

Table 2: Comparison of different models implemented

Conclusion

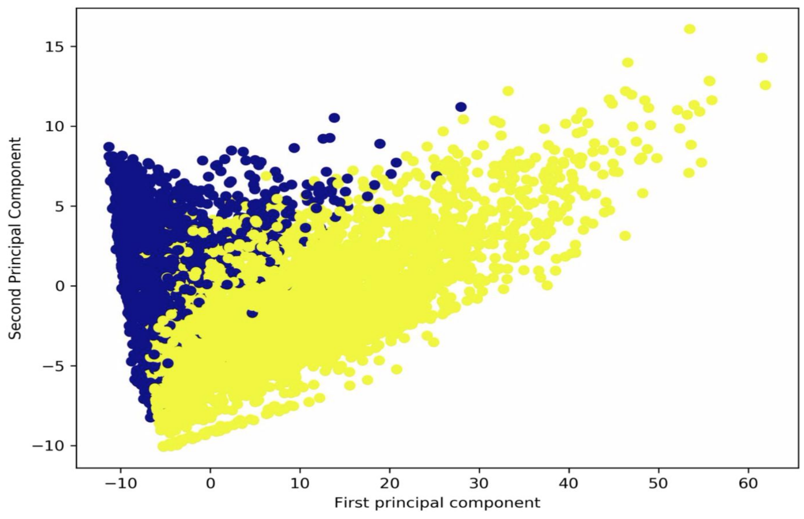


Figure-2 Scatter plot of train data set with dimension reduced from 9 features to 2 principal components.

- The figure-2 above is the scatter plot of a random sample of 10000 data points from our training data set with dimensions reduced to 2 principle components. The blue points are the true examples and the yellow points are the fake examples.
- In our analysis, we found out that the true and fake edges in our training data could be classified into different classes, that is why we went ahead with the classification models listed in the above table.
- Finally, as listed in the table above which compares the differents models they all had pros and cons with similar kind of performance. The PCA scatter plot enabled us to come to a conclusion that Logistic Classification was the best model given the scatter and the fact that it does not look linearly separable. Furthermore, looking at the scatter and pair plot of our features we added polynomial terms to our Logistic model which improved the accuracy on the test data set.
- We could not overcome overfitting problem on our training dataset. Models could distinguish objects into separate classes on our training sets but we witnessed significant drop of performance on public test set. We should have spend more time on studying of distribution of unknown edges.