Machine Learning Course Project

Kristóf Balázs
Facultad de Informática de Barcelona
BDMA

Stefanos Kypritidis
Facultad de Informática de Barcelona
BDMA

Introduction

In this project, we address the task of root word prediction in syntactic dependency trees using multilingual data, framing it as a binary classification problem where each node (word) in a sentence graph is classified as root or non-root. While the available training dataset consists of 10,500 labeled training samples, covering 21 languages with 500 sentences each, the test dataset consists of 10,395 unlabeled samples, covering the 21 languages with 495 sentences each. Each sentence is modeled as a graph, where nodes represent words and edges capture syntactic dependencies. Preprocessing included computing 13 centrality measures using NetworkX and language&sentect-specific normalization. Additional features, such as graph metrics and Node2Vec embeddings, were tested but offered limited improvement. Final models—PCA & logistic regression and XGBoost, trained separately per language—were tuned via cross-validation and evaluated using a custom root prediction metric. This report outlines the dataset, methodology, and comparative results.

Data Exploration

It is important to start by distinguishing the existence of two datasets, the train and the test. The train set contains the labels (root node) for each sentence, while the test does not contain any labels. In the data exploration process, we focused on the train set because this set will be used to build machine learning models, and valuable insights can be obtained from some of its attributes.

The train set contains five attributes and each row describes a sentence in a particular language. In detail, the attributes are *language, sentence number, number of words, edgelist, and the id of the root word.* Next, we checked the types and the number of null values for each attribute. There are no null values. Moreover, the *language* attribute is of type string, *edgelist* is a list, and all other attributes are integers.

Analyzing the basic statistics of the integer attributes, we discovered that on average each sentence has 18.8 words and the root node can be from the 1st till the 68th word (depending on the size of the sentence). The training set includes 21 different languages, with 500 sentences per language, resulting in a total of 10,500 rows. While there are only 500 unique sentences overall, each one is translated into all 21 languages.

Next we focused on the sentence length attribute. Visualizing the distribution of sentence length, it is positively skewed with a mode around 15-20 words (Figure 1 of Appendix). However, we noticed that this distribution is not the same for every language by visualizing the sentence length distribution for each language (Figure 2). As a result, we created a barplot with the min, mean, median and max number of words for each language as seen in Figure 3.

The Figure 3 reveals that Japanese has the highest average word count per sentence at 25.8, while Finnish has the lowest with 13.5, likely due to Finnish condensing meaning into fewer words and Japanese relying on more separate words for the same ideas. Polish, Czech, and Russian show similar sentence statistics, reflecting their shared origin in the Slavic language family, while German and English exhibit nearly identical sentence statistics, as both are Germanic languages. Similarly, Portuguese, Spanish, Italian, and Galician have comparable statistics, which can be attributed to their common roots in Romance languages, all deriving from Latin. Those insights were valuable and played a key role in shaping our modeling approach later on.

The next step was to transform the dataset from a multi-class problem to a binary classification problem. For every row of the train data, a graph was created using the attribute of *edgelist* and the library networkX. Afterwards, the centrality measures of *degree, closeness, harmonic, betweenness, load, pagerank, eigenvector, katz, information, current flow betweenness, percolation, second order, and laplacian* were calculated for each node (word) of the graph (sentence) using the corresponding functions of networkX. The new dataset is an expanded version of the previous one, with each node (word) now also having a boolean label indicating whether it is the root.

Another important step was normalizing the centrality measures of the expanded data, but only within sentence groups of the same language. This is done because different languages produce different syntactic trees, and also sentences within the same language produce structurally different trees.

Feature Experiments

This section outlines the various measures and analyses we conducted to identify graph patterns that could enhance feature engineering, with the goal of improving the performance of our classifiers. However, the features generated from these patterns did not lead to improved model results and were therefore excluded from the final classifiers and modeling approaches.

We investigated whether the root node being a leaf node—defined as having only one neighboring edge—had any impact on our results. While this pattern was notably observed in the Hindi and Japanese languages (Figure 4), it did not lead to any improvement in our prediction performance. Furthermore, we plotted kernel density estimates (estimations of the probability distribution) for five node-level features—eccentricity, distance to center, minimum leaf distance, subtree max depth, and height (longest shortest path from a node to any other node)—to compare root and non-root nodes across languages. This analysis could help reveal how root nodes differ from other nodes in terms of their graph-theoretic roles, but did not produce any significant insight (Figure 5).

A new metric, Normalized Position in Sentence (vertex number divided by sentence length), was introduced. However, analysis using probability densities, medians, boxplots, and correlations showed it is not a strong predictor of root status (Figures 6, 7, 8). Moreover, we introduced Subtree-Balance Metrics, computing max branch size and subtree entropy per node. Visualizations showed limited patterns (Figures 9, 10), and prediction accuracy was only slightly above random.

Additionally, we generated 32-dimensional Node2Vec embeddings for each node to predict the root node. A t-SNE visualization showed no clear separation between root and non-root nodes (Figure 11). A simple baseline, predicting the root as the node with the largest embedding norm, achieved accuracy barely above random selection—indicating Node2Vec is not effective for this task.

An exploration was made of whether root nodes differ from their neighbors in centrality by computing deltas for degree, PageRank, closeness, and betweenness. Pairwise scatterplots showed overlapping distributions for root and non-root nodes, with no clear separation (Figures 12, 13). A simple heuristic using PageRank delta for root prediction performed only slightly better than random, suggesting limited usefulness of these features.

Another experiment was the alternative attribute of: 1-hop neighbor aggregates of centrality measures. In this approach, we computed the mean, max, and sum of several centrality scores over each node's direct neighbors, and normalized these features within each language-sentence group. Although this preprocessing step offered a new perspective on node importance, it led to only marginal accuracy gains (1–2%) in the main models. Unfortunately, these slight improvements did not justify the added complexity and computational cost, making this approach suboptimal.

Modeling methods

Logistic regression

Our approaches leverage the observation that different languages exhibit distinct sentence length and centrality scores distributions and root characteristics and patterns. Accordingly, a separate machine learning pipeline is trained for each language, consisting of PCA and StandardScaler followed by logistic regression. To optimize the hyperparameters of each pipeline, we use *GridSearch* with 5-fold cross-validation. Due to the extensive grid of hyperparameters defined, the process of identifying the optimal configuration for each of the 21 language-

specific pipelines was computationally intensive. GridSearchCV, which performs an exhaustive search over all possible combinations within the defined grid, was employed to achieve the optimal combination of hyperparameters.

Using those pipelines that combine PCA with Logistic Regression is a practical and effective approach for this kind of binary classification problem. Additionally, PCA helps reduce the number of features by capturing the most important patterns in the mentioned centrality scores, which not only simplifies the model but also helps prevent overfitting (which is useful when dealing with variable-sized trees). By removing noise and correlations between features, PCA makes the data more suitable for linear classifiers like Logistic Regression. Logistic Regression, in turn, is a fast, interpretable method that performs well even with imbalanced datasets, especially when regularized properly. Moreover, Logistic regression models the probability that a given node belongs to a particular class (root class) which was necessary for the our custom scoring function described later.

The hyperparameter grid included PCA's *n_components* and logistic's regression *penalty, C, solver, l1_ratio, fit_intercept, warm_start, max_iter*. This setup allowed the gridsearch to find the best regularization approach to reduce overfitting, balance the bias-variance tradeoff through *C*, choose the most suitable optimization algorithm, and ensure solver convergence within a reasonable number of iterations.

The original training set is split into a training set (80%) and a validation set (20%) using *GroupShuffleSplit*. This method ensures that all occurrences of the same sentence remain in either the training or validation set, preventing data leakage. In addition, *StratifiedGroupKFold* is used for the 5-fold cross-validation which ensures that data of the same sentence can not be found in different folds. Similarly, this helps prevent any data from the training folds from leaking into the test fold.

A custom scoring function was created in order to evaluated the performance of each hyper-parameter combination. This function selects a single root node for each sentence of a distinct language. In fact, it chooses the node with the highest probability of being labeled as the root (class 1). Since the classifier is based on logistic regression, obtaining probability estimates is straightforward. The final score is calculated as the percentage of sentences in which the correct root was selected which corresponds to the 0-1 loss.

For each language, the optimal hyperparameters are identified, and the corresponding score is printed. A new pipeline is then trained using both the training and validation data to utilize the entire dataset. This trained pipeline is stored in a dictionary. The test data undergoes the same preprocessing steps as the training data. For each test sample (identified by its unique ID), the relevant language model is used to make predictions (binary classification at the node level). The final prediction for the root node of the sentence (identified by the ID) is the node with the highest probability of belonging to class 1.

XGBoost classifier

The main non-linear approach uses a separate XGBoost classifier per language to predict the root node. The rest of the non-linear approaches are discussed in the Modeling Experiments section. It was chosen as it is very effective in capturing complex, non-linear relationships between features, which is extremely important for us given the structure of syntactic dependency trees. It also handles imbalance naturally with the hyperparameter *scale_pos_weight*, which penalizes misclassification of the minority class. Moreover, it has regularization methods (L1 and L2 penalties), which mitigates the most frequent problem we ran into with more complex models on the limited training set, which is overfitting. Moreover, it also has GPU acceleration support through the *hist* algorithm (parallel computations), which reduced training and hyperparameter tuning times (with CUDA), so we could explore larger hyperparameter spaces.

The training pipeline follows a similar structure to logistic regression. We again split the data into training (80%) and validation sets (20%) using *GroupShuffleSplit*. To handle class imbalance, we adjusted the model parameter *scale_pos_weight* inversely proportional to the class distribution. We used the same scoring function as with logistic regression.

Hyperparameter tuning was done through a grid search covering learning rate (*eta*), tree depth (*max_depth*), regularization parameters (*gamma*, *reg_alpha*, *reg_lambda*), and sampling parameters (*subsample*, *colsample_bytree*, *min_child_weight*). Early stopping with a maximum of 200 boosting

rounds was added, automatically terminating training after no improvement for 20 rounds, which also helped with overfitting.

After the initial tuning identified the best number of trees (*ntree_limit*), we performed an additional fine-tuning step by evaluating nearby iteration counts (*+/- 20 rounds*). We selected the configuration that maximized our custom root-score metric.

The best hyperparameters, scalers, and models per language were stored. The final predictions on the test set were obtained by selecting the node with the highest predicted probability, as in previous methods.

Modeling Experiments

Random Forest was our initial choice for the non-linear model, mainly because it is easy to interpret. Following the methodology from our practical labs, we performed hyperparameter tuning using out-of-bag (OOB) validation instead of cross-validation. However, after evaluating a hyperparameter grid similar to the one used for XGBoost, we found that Random Forest consistently underperformed compared to XGBoost for the same data and tuning effort. Consequently, we stopped further exploration with Random Forest and focused our non-linear modeling efforts exclusively on XGBoost.

Additionally, we conducted a feature importance analysis to evaluate the contribution of both standard centrality measures and our engineered subtree features. The results indicated that, while some features like *laplacian* and *current_flow_betweenness* consistently showed relatively high importance across multiple languages, our engineered subtree-based features provided minimal predictive value. The feature importance patterns varied across languages, and no universal set of features consistently distinguished root nodes across all cases (Figures 15, 16). Thus, using one classifier per language family or even adding a universal classifier trained across all languages did have optimal results; indeed, the outcomes of our experiments confirmed these.

We also did a small literature review on papers where a related problem was presented. Inspired by similar graph-based root prediction problems, such as rumor-source detection with graph convolutional networks Dong et al. (2019), we investigated the applicability of **Graph Source Localization (GraphSL)** methods Wang and Zhao (2024). Graph source localization attempts to solve an inverse diffusion problem: given a set of "infected" nodes, the goal is to identify the origin nodes of the diffusion. This fits our task very well, where dependency trees can be conceptualized as diffusion processes originating from a single root node.

Specifically, we used the GraphSL library by Wang and Zhao (2024), which provides implementations of both heuristic (LPSI, NetSleuth, OJC) and graph neural network-based methods for source localization. Due to limited computational resources (NVIDIA GeForce RTX 3050 6 GB GPU), we experimented on a small scale, evaluating the prescribed methods on a subset of Polish sentences from the dataset. Among the tested methods, LPSI had the highest validation AUC, outperforming NetSleuth and OJC.

The initial results had very promising potential; however, full-scale tuning and more experimentation are quite computationally prohibitive within the project's scope. With enough computational resources and large hyperparameter tuning, this methodology, particularly when combined with graph neural network-based approaches (e.g., GCNSI, SLVAE), is extremely promising and aligns closely with recent advancements in inverse diffusion and graph-based prediction problems Dong et al. (2019); Wang and Zhao (2024).

Lastly, we were aware that the dataset included leakage, specifically directionality in the edge lists. Indeed, one simple heuristic based on treating edges as directed arcs and selecting the node with zero in-degree achieved a perfect score of 100% accuracy. We **independently** identified this and experimented with it in a separate notebook, just to verify it. However, we did not submit these results to Kaggle, nor did we incorporate any directed-graph-based features in our main modeling pipelines. Throughout the project, we strictly treated the graphs as undirected.

Results Comparison

In this section, we compare the performance of the different modeling approaches. For logistic regression, and XGBoost, we used the same validation set to estimate generalization error. To speed up the evaluation for the random forest model, we relied on the out-of-bag (OOB) score instead. The results are summarized in Table I and the best parameters in the appendix-tables.

Modeling Approach	Score
Logistic Regression	0.297
Random Forest	0.411
XGBoost	0.549

TABLE I: Modeling approaches and 0-1 loss score

Final model

Our final submission uses the per-language XGBoost classifiers. Across all 21 languages, these models achieved the highest root-accuracy on both our hold-out splits and the full test set. This pipeline has efficiency (GPU-accelerated training, early stopping), robustness to imbalance (built-in weighting), and simplicity of deployment (single classifier per language, no ensemble stacking). Future work could integrate XGBoost with alternative graph-based source localization outputs (e.g. LPSI scores) or stacked ensembles selected by validation performance, but given our resource constraints, the current XGBoost solution represents the best balance between accuracy, runtime, and good methodology.

Conclusions

In conclusion, thorough data preparation and attribute tests were crucial to identify the most useful features in predicting sentence root nodes. Key insights came from analyzing the labeled training set, including sentence statistics that reflect linguistic similarities within language families—Slavic (Polish, Czech, Russian), Germanic (German, English), and Romance (Portuguese, Spanish, Italian, Galician). Converting the task to a node-level binary classification and applying centrality measures, and normalization within sentence groups of the same language helped tailor the model per language. While various engineered features and embedding approaches were tested, most did not significantly improve performance, highlighting the importance of tested, data-driven feature selection.

Computational limits ($\approx$500 labelled samples per language and a single RTX 3050 GPU) constrained our GraphSL experiments to a small Polish subset, but full GNN-based tuning was infeasible. With more data and compute, a hybrid approach that feeds GraphSL scores (e.g., LPSI outputs) into the per-language XGBoost models or a simple ensemble could further improve accuracy.

We used a sentence-wise grouping in both hold-out splits and cross-validation to eliminate leakage and used a custom "root-score" metric matching the Kaggle 0-1 loss. Finally, we verified that the provided "randomized" train/test split reproduces identical training sets and validation errors. This is also reflected in the fact that the validation error is very similar to the generalization error of the approximate 10% kaggle test set.

References

Ming Dong, Bolong Zheng, Nguyen Quoc Viet Hung, Han Su, and Guohui Li. Multiple rumor source detection with graph convolutional networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 569–578, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369763. doi: 10.1145/3357384.3357994. URL https://doi.org/10.1145/3357384.3357994.

Junxiang Wang and Liang Zhao. Graphsl: An open-source library for graph source localization approaches and benchmark datasets. *Journal of Open Source Software*, 9(99):6796, 2024. doi: 10.21105/joss.06796. URL https://doi.org/10.21105/joss.06796.