Complex and Social Networks

Giulia Pinciroli (`giulia.pinciroli@estudiantat.upc.edu`) and Gabriele Villa (`gabriele.villa@estudiantat.upc.edu`)

Laboratory 3 - Significance of Metrics

Contents

1 Introduction

Real-world networks often show patterns that are not simply due to chance. To understand if a network property is truly meaningful or just a random artifact, we needed to compare our observations with appropriate randomized models.

In this work, we investigated the statistical significance of the mean closeness centrality $\mathcal{C}$ in global syntactic dependency networks constructed from multiple languages in the Parallel Universal Dependencies (PUD) collection. These networks represented words as vertices, with edges connecting word pairs that formed syntactic dependencies in a treebank corpus. Closeness centrality measures how close a node is to all other nodes in the network by computing the average of the reciprocal geodesic distances.

To assess whether the observed closeness centrality values were significantly large, we used Monte Carlo simulations under two null hypotheses: the Erdős-Rényi binomial graph model, which preserved only the number of vertices and edges, and the switching model, which additionally preserved the degree sequence of the original network. By estimating p-values through repeated randomization, we quantified how much the network organization deviated from these baseline expectations.

2 Results

2.1 Basic Network Properties

As a first step in our project, we computed basic network properties of our graphs and obtained the results summarized in Table 1.
These values describe the global structure of the dependency networks and provide a quantitative overview of their size and connectivity before further analysis.

Table 1: Basic network properties for PUD syntactic dependency networks.

Language	N	E	$\langle k \rangle$	δ
Arabic	4785	16251	6.792	0.00142
Chinese	5446	16919	6.213	0.00114
Czech	5318	14984	5.635	0.00106
English	4666	16908	7.247	0.00155
Finnish	4255	11569	5.438	0.00128
French	4630	18186	7.856	0.00170
Galician	4476	17438	7.792	0.00174
German	5385	17316	6.431	0.00119
Hindi	4420	15319	6.932	0.00157
Icelandic	4837	14786	6.114	0.00126
Indonesian	3726	14961	8.031	0.00216
Italian	4814	18072	7.508	0.00156
Japanese	4905	19899	8.114	0.00165
Korean	8258	14188	3.436	0.00042
Polish	5033	14840	5.897	0.00117
Portuguese	6237	19007	6.095	0.00098
Russian	5155	15432	5.987	0.00116
Spanish	4528	17396	7.684	0.00170
Swedish	5006	15676	6.263	0.00125
Thai	4040	15988	7.915	0.00196
Turkish	4597	13437	5.846	0.00127

The properties include:

- N: number of vertices
- E: number of edges
- $\langle k \rangle = 2E/N$: mean degree
- $\delta = 2E/[N(N-1)]$: edge density

2.2 Parameter tuning and models performances

The early-stopping approach was applied as an optimization technique to reduce computational cost during the estimation of mean closeness centrality.
For each language, a value M was obtained representing the number of vertices required for the stopping condition to be met.
These values are reported in Table 2.

Table 2: Language-specific M values used for early-stopping bounds.

Language	M
Arabic	4306
Chinese	4901
Czech	4786
English	4199
Finnish	4042
French	4167
Galician	4028
German	4846
Hindi	3978
Icelandic	4353
Indonesian	3540
Italian	4333
Japanese	4414
Korean	7845
Polish	4530
Portuguese	5613
Russian	4640
Spanish	4075
Swedish	4505
Thai	3838
Turkish	4137

The performance of the different implementations of closeness centrality was then evaluated on two random-graph baselines: an Erdős–Rényi (ER) model with the same N and E as the Indonesian network (network with smallest N), and a degree-preserving switching-model graph.
Table 3 summarized the execution times for each method.

Table 3: Runtime (in seconds) for closeness-centrality implementations on two random-graph baselines matched to Indonesian: an Erdős–Rényi (ER) graph with N, E, and a degree-preserving switching-model graph.

Model	ER graph (s)	Switching graph (s)
Built-in R function (igraph)	0.382766962051392	0.359624862670898
Sampled approximation	0.709634065628052	0.601946115493774
Degree-1 optimization	5.19723606109619	5.33880400657654
Early stopping with bounds	5.19278597831726	6.03214001655579
Combined optimization	4.19898700714111	5.45836305618286

2.3 Statistical significance

After identifying the most efficient implementation, we proceeded to apply it to the full dataset to evaluate the statistical significance of the observed mean closeness centrality across all languages.
The analysis aimed to determine whether the observed values of the metric were significantly larger than those expected under random graph models.
The test was assessed for all 21 PUD languages using two null models: an Erdős–Rényi (binomial) random graph and a degree-preserving switching model.
For each language, 500 random graphs ($T = 500$) were generated under each hypothesis, and the proportion of simulated mean closeness values greater than the observed one was used to estimate the p-value.
The resulting observed mean close centrality values and corresponding p-values are summarized in Table 4.

Table 4: Mean closeness centrality and estimated p-values under two null hypotheses: the Erdős–Rényi (binomial) model and the degree-preserving switching model, with $T = 500$ Monte Carlo iterations.

Language	Closeness Centrality	p-value (binomial)	p-value (switching)
Arabic	0.3023	0.0000	0.9880
Chinese	0.2955	0.0000	0.8220
Czech	0.2821	0.0000	1.0000
English	0.3117	0.0000	1.0000
Finnish	0.2767	0.0000	1.0000
French	0.3444	0.0000	1.0000
Galician	0.3463	0.0000	1.0000
German	0.3229	0.0000	1.0000
Hindi	0.3321	0.0000	1.0000
Icelandic	0.2950	0.0000	1.0000
Indonesian	0.3039	0.0000	1.0000
Italian	0.3399	0.0000	1.0000
Japanese	0.3382	0.0000	0.9860
Korean	0.2127	0.0000	0.0000
Polish	0.2849	0.0000	1.0000
Portuguese	0.3133	0.0000	0.9860
Russian	0.2866	0.0000	0.9940
Spanish	0.3418	0.0000	1.0000
Swedish	0.2982	0.0000	0.9980
Thai	0.3066	0.0000	0.7620
Turkish	0.2882	0.0000	0.7400

3 Discussion

3.1 Basic Network Properties

The basic properties reported in Table 1 show that the number of vertices (N) varies significantly across languages, reflecting differences in vocabulary size and annotation practices in the treebanks.

The mean degree values $\langle k \rangle$ range from about 3.4 (Korean) to 8.1 (Japanese and Indonesian), showing moderate variation in connectivity. Languages with higher $\langle k \rangle$ values tend to have more interconnected lexical networks, while those with lower values, such as Korean, exhibit sparser structures.

The general low edge densities ($\delta < 0.0022$) indicate that all PUD syntactic dependency networks are sparse. This value is expected for linguistic networks, as not all words co-occur syntactically.

3.2 Parameters and model selection

The M values in Table 2 represent the number of vertices required for the early-stopping bounds on mean closeness centrality to become decisive.
When comparing the M values to the total number of vertices N from table 1, we notice that M typically represents $90 - 95\%$ of N for most languages.
For example, English required $M = 4199$ out of $N = 4666$ vertices ($\approx$ 90%, while Finnish required $M = 4042$ out of $N = 4255$ ($\approx$ 95%).
These ratios confirm the theoretical expectations that M grows roughly with N but remains smaller, demonstrating that early stopping provides computational savings without loss of precision.

The practical impact of these M values becomes evident when examining the runtime of the different implementations of mean closeness centrality.
Table 3 compares the different runtime performances evaluated on two random-graph baselines: an Erdős–Rényi graph with the same number of vertices and edges as the Indonesian network (smallest N), and a degree-preserving switching-model graph.
The results confirm the theoretical trade-off between computational precision and efficiency discussed in the theory.

The `igraph` implementation of harmonic centrality, which relies on optimized internal algorithms, achieved the fastest execution times, completing in under 0.4 seconds for both graph models.
Manual implementations based on breadth-first search (BFS) that considered only 10% of the network's nodes required almost twice the runtime of the `igraph` function, while producing approximated results.
Full manual implementations, computing distanced for all vertices, were considerably slower due to explicit BFS calculations, surpassing the 5 seconds mark.
The optimized approaches introduces in this project, particularly the one combining caching of BFS results with the early stopping threshold M, achieved notable performance gains, reducing computation time by approximately $20 - 25\%$ compared to the full exact manual method while maintaining identical results.
These findings demonstrate that caching for repeated local structures and early termination based on bounds of $\mathcal{C}$ are effective in practice and scale well across different null-model networks.

For the scope of this project, which focuses on assessing the statistical significance of the mean closeness centrality metric, we proceeded with the built-in `igraph` implementation, as it provides the fastest and most efficient computation for large-scale iterative analysis.

3.3 Statistical significance

Having established the optimal computational approach, the next step involved applying it to the complete dataset to evaluate the statistical significance of the observed closeness centrality across all languages under two null hypotheses.

The results in Table 4 show a clear contrast between the two null hypotheses considered. Using a Monte Carlo approach with $T = 500$ iterations for each language, we estimated the probability that a random graph would produce a mean closeness centrality greater than the observed value.

Under the Erdős–Rényi (binomial) null model, all languages exhibit p-values equal to 0, indicating that the observed mean closeness centrality in each real syntactic network is significantly larger than would be expected by chance in a random graph with the same number of vertices and edges.
This confirms the theoretical expectation that linguistic dependency networks are not random in their global organization: real networks tend to have shorter average path lengths and higher overall connectedness than purely random structures.

When comparing the results to the degree-preserving switching model, the pattern reverses.
For nearly all languages, p-values are close to or equal to 1 (ranging from 0.74 for Turkish to 1 for most languages), indicating that the observed mean closeness centrality is not significantly larger than in random graphs that preserve the same degree sequence. The only exception is Korean, which shows $p = 0.0000$ under this model, indicating that its observed closeness is significantly smaller than expected, which could be linked to the nature of the structure of the language, which can be observed in Table 1. In fact, we notice that Korean has a much bigger N than the other languages, while presenting the lowest amount of edges E.
These outcomes align with the theoretical insights from the lecture slides, which emphasize that the switching model provides a much stricter baseline: by maintaining the local degree distribution, it accounts for the heterogeneity of node connectivity that largely determines the path lengths in a network.
Thus, when the sequence is controlled for, the residual structure of the network no longer contributes significantly of increasing mean closeness, implying that the metric is largely explained by the degree distribution itself.

Across languages, the absolute values of mean closeness centrality vary moderately (roughly between 0.21 and 0.35), but overall significance pattern is highly consistent.
Languages with higher closeness values (e.g. French, Galician, and Spanish) and those with lower values (e.g. Finnish or Turkish) all follow the same significance trend: strongly significant under the binomial null, non-significant under the switching null.
This results suggest that, despite linguistic and typological diversity, the global organization of syntactic dependency networks share similar topological properties one degree effects are controlled for.

3.4 Analysis of Valid and Invalid Switchings

Given two edges $u \sim v$ and $s \sim t$ in a graph, we analyzed which switchings preserve the degree sequence and which produce inadmissible edges.

3.4.1 Switchings Preserving the Degree Sequence

A switching preserves the degree sequence if and only if all four vertices u, v, s, t are distinct. In this case, the original edges $\{u, v\}$ and $\{s, t\}$ can be replaced by either:

- Configuration 1: $\{u, s\}$ and $\{v, t\}$, or
- Configuration 2: $\{u, t\}$ and $\{v, s\}$

Both configurations maintain the degree of each vertex unchanged, as each vertex loses one connection and gains exactly one new connection.

3.4.2 Switchings that Preserve Degree Sequence but Produce Inadmissible Edges

Even when the degree sequence is preserved, a switching may produce edges that violate graph constraints:

Case 1: Coincident vertices (self-loops). If two vertices coincide (e.g., $u = s$), the switching would create edges such as $\{u, u\}$ and $\{v, t\}$. The self-loop $\{u, u\}$ is inadmissible in simple graphs, making this switching invalid despite preserving degrees.

Case 2: Multi-edges. When all four vertices are distinct but one of the new edge configurations already exists in the graph, the switching would create a multi-edge. For example, if $\{u, s\}$ already exists before the switching, replacing $\{u, v\}$ and $\{s, t\}$ with $\{u, s\}$ and $\{v, t\}$ would duplicate the edge $\{u, s\}$, which is not allowed in simple graphs.

3.5 Conclusion

In conclusion, the findings in this report support the theoretical claim that mean closeness centrality is predominantly a degree-driven metric: its large values in linguistic networks arise mainly from the degree distributions rather than from additional higher-order structural organization.

The Monte Carlo implementation with $T = 500$ iterations provided stable empirical estimates of p-values while ensuring computational feasibility across all 21 languages, especially when combined with parallelized computation.
The comparison between the two null hypotheses further highlights the difference in interpretability: the Erdős–Rényi (binomial) model, which ignores degree constraints, consistently overestimates the statistical significance of the observed metric, while the degree-preserving switching model reveals that most networks do not differ substantially from random graphs with the same local connectivity patterns.

Overall, these findings confirm that once degree effects are accounted for, syntactic dependency networks across languages exhibit similar structural organization, suggesting that much of their global connectivity can be explained by local attachment properties rather than by language-specific hierarchical or long-range dependencies.

4 Methods

4.1 Implementation and Performance Optimization

We implemented five different methods to compute the mean closeness centrality $\mathcal{C}$ for graphs generated under the null hypotheses. Since the computational logic was consistent across methods, we tested their performance on both an Erdős-Rényi graph and a switching model with the same size as the Indonesian syntactic network, which had the smallest number of nodes N among all languages. This approach allowed us to efficiently identify the fastest implementation without wasting computational resources on larger networks.

The five methods we tested were:

1. **Built-in R function**: We used the `harmonic_centrality` function from the `igraph` package.

2. **Sampled approximation**: This method computed closeness centrality on a random sample of nodes, controlled by a `sample_fraction` parameter. We set this value to 0.1, which provided excellent results despite the small sample size.

3. **Degree-1 optimization**: This method exploited nodes with degree equal to 1 to avoid redundant BFS computations.

4. **Early stopping with bounds**: This method used lower and upper bounds ($\mathcal{C}_{min}$ and $\mathcal{C}_{max}$) to determine early whether $\mathcal{C}_{NH} \geq \mathcal{C}$ or $\mathcal{C}_{NH} < \mathcal{C}$, stopping computation once the condition was satisfied.

5. **Combined optimization**: This method integrated both the degree-1 optimization and early stopping strategies.

4.2 Implementation Details

4.2.1 Distance Computation

For all methods except the built-in R function, we computed shortest path distances using breadth-first search (BFS) with the `bfs` function from the `igraph` package. This ensured efficient $O(N + E)$ time complexity for distance calculations.

4.2.2 Caching Strategy for Degree-1 Nodes

In the degree-1 optimization method, we implemented a caching mechanism to avoid redundant BFS computations. When processing a node i with degree 1, we identified its single neighbor k. If BFS distances from k were already cached from a previous computation, we reused them and simply added 1 to account for the edge between i and k. Otherwise, we computed distances from i and stored them in the cache for potential future reuse. This significantly reduced computation time for networks with many degree-1 nodes.

4.2.3 Parameter Selection for Early Stopping

For the early stopping methods, we dynamically determined the optimal value of M (the number of nodes to process before checking bounds) for each graph. We sampled potential M values ranging from 80% to 100% of N, with a 5% step size. For each candidate M, we tested whether the optimization condition was satisfied, and selected the smallest M that triggered early stopping.

Additionally, we sorted nodes by degree before computing their closeness values, but the ordering strategy differed depending on the null hypothesis. For the Erdős-Rényi model, we sorted nodes in **ascending order** by degree. This strategy was motivated by the observation that $\mathcal{C}_{ER}$ was typically smaller than $\mathcal{C}$ for the real networks. Since low-degree nodes generally have lower closeness centrality values, processing them first increased the likelihood of satisfying the condition $\mathcal{C}_{max}^{NH} < \mathcal{C}$ earlier, thus reducing overall computation time.

Conversely, for the switching model, we sorted nodes in **descending order** by degree. This choice was based on the observation that $\mathcal{C}_{SW}$ was often greater than $\mathcal{C}$ for the real networks. Since high-degree nodes typically exhibit higher closeness centrality values, processing them first increased the likelihood of satisfying the condition $\mathcal{C}_{min}^{NH} \geq \mathcal{C}$ earlier, again reducing computation time.

4.3 Method Selection

We benchmarked all five methods using `Sys.time()` to measure execution time on the Indonesian network. The fastest method was then used to compute p-values for all languages in the PUD collection.

All the following computations, as well as some parts of the pre-processing stages, were parallelized across 15 cores using R's `parallel` library, substantially reducing total runtime and enabling efficient large-scale analysis across the 21 networks.

4.4 Number of Random Graphs for p-value Estimation

For the Monte Carlo estimation of p-values, we set the number of random graphs $T = 500$. Since our optimized implementation was computationally efficient, we chose a higher value of T to improve the accuracy of the p-value estimates.

4.5 Switching Parameter for the Degree-Preserving Model

For the switching model, we determined the number of edge switching attempts as $Q \cdot E$, where E is the number of edges and $Q = \max(10, 2\log E)$. According to the coupon collector's problem, the expected number of random draws needed to sample each edge at least once scales as $Q \sim \log E$. By using $Q = 2\log E$, we ensured sufficient randomization to thoroughly explore the space of graphs with the same degree sequence.

We set a lower bound of $Q = 10$ to guarantee a minimum level of randomization where $2\log E < 10$ in case one wanted to replicate the experiment with a smaller network. This prevented under-mixing in cases where the network had few edges, ensuring that even small graphs underwent adequate randomization.

4.6 Closeness Centrality Computation for Real Networks

For computing the mean closeness centrality $\mathcal{C}$ of the real syntactic dependency networks, we used the `harmonic_centrality` function from the `igraph` package. Based on the performance benchmarks described in Section 2, this method proved to be the fastest among all implementations tested, allowing us to compute exact closeness centrality values efficiently for all languages in the dataset.

4.7 Graph Storage Format

We stored each syntactic dependency network as a text file using an edge list format. Each line in the file represents a single edge by listing the two words forming that edge.

4.8 Implementation of the Switching Model

To generate randomized graphs under the switching model, we used the `rewire` function from the `igraph` package with the `keeping_degseq` method. This function performs $Q \cdot E$ edge rewiring attempts while preserving the degree sequence of the original network.

Importantly, the rewiring algorithm attempts each switch even if it cannot be completed, ensuring that the total number of iterations $Q \cdot E$ includes both successful and unsuccessful switching attempts. The function automatically rejects switches that would create loops (edges connecting a node to itself) or multiedges (multiple edges between the same pair of vertices), as required by the lab specifications.