COMP0123 Complex Networks and Web
Network Analysis of the Evolution of Programming Languages

Student Number: 20000962
Student UPI: SJGOH80
Candidate Number: HYHL0

January 2024

Abstract

Programming languages are the driving skeleton to cutting-edge inventions and applications in the current era. Modern programming language has evolved drastically since its birth over a century ago. However, it is unknown as to how all programming languages are linked and influencing each other. In this work, the evolutionary relationship between all programming languages is studied. With network science algorithms and techniques, the key drivers in the programming language evolution network will be investigated. Its network community structure is also analysed for realising the underlying impact on programming language categories. The report begins with an introduction to the topic and the formulated questions for this research. Next, results and analysis will be presented in detail, then concluding on the key findings, limitations and future work.

1 Background

Tracing back to as early as 1880s, the first ever programming language (PL) was invented by Ada Lovelace [1]. Information technology has grown rapidly ever since then, with compiled PLs such as C and Javascript, and modern PLs such as Python and Java being the common and popular languages in the current era. With these, PL evolution has marked significant advancement in both social and technological fields [2] such that hardware and applications are developed for simplifying all kinds of daily tasks.

Therefore, it is crucial to understand the key PLs in the network for determining the future direction and required improvements for further advancement. The community structure among PLs is important for observing the well-developed vs. new categories for pushing the limits in PLs. To understand these, the directed PL evolution network (PLEN) will be studied for answering the research questions introduced in the next section.

2 Research Questions

The research questions below will be investigated.

1. What is/are the key contributing PL(s) driving evolution in PLEN?
   Commonly used PLs are merely a small subset of the numerous invented PLs, whether they remain in use or are abandoned. For understanding the trend in evolution such as the key PL(s) that drove the evolution as well as the core inspirations of the popular modern PLs, network science algorithms are applied. In the research, results from betweenness centrality, PageRank and HITS (Hubs and Authority) will be analysed and compared for obtaining insights of the important PL nodes.

2. Do the community structure in PLEN infer PL categorisation?
   Due to PL evolution being uneven most of the time, its categorisation is vague. The growth of a PL can occur such that functionalities from different PL group is chosen. However, it is important to understand the categorisation to better select the correct tool/framework for project implementation. Therefore, the network science concepts of modularity and community structure are investigated and analysed to study the research problem.

3 Literature Survey

The directed, temporal network of PL is investigated in [2] for inspecting its evolution through punctuated equilibrium. It is shown through trees and probabilities that the evolution is highly uneven where new features selected are related to novel technology and social niches [2]. Minimal centrality is also studied such that core PLs include C, Java, Pascal and Lisp.

The most used packages in the R programming language is investigated in [3] through the network display of all R package dependencies in the past. From the study, the commonly used packages merely takes up about 50% of all available packages in R, showing a clear exponential decay [3]. The study merely analyses links and node degrees for understanding the most-utilised packages in R, thus future work on other metrics such as centrality, clustering and modularity remained as open questions.

The required skillset in the field of Computer Science (CS) is also studied through network science methods in [4]. With dataset extracted from the LinkedIn social network, it is discovered that the core communities of skills include generalists, infrastructure and security, software development, and embedded systems [4]. However, the study investigates both technical and soft skills in general, without focus on the evolution network or PL-related only.

In [5], network methods are used for investigating the open-source PL ecosystem through temporal networks. The Top 10 PL ecosystem such as Java, C#, Go, Ruby, Scala and more are respectively constructed. While there are suggestions on the possible network science metrics such as betweenness centrality and clustering coefficient, and structure which could be experimented on, actual results are missing and suggested as future work [5].

Understanding the previous work described above, [2] did not analyse the network structure and community within the PL network although minimal centrality is investigated. In [3] and [5], the authors focused on datasets and network construction without deep analysis into the possible metrics. Finally, the skills required in CS are studied in general without a focus in PL [4]. Thus, with reference to these past studies, the research questions of investigating key drivers of PLEN as well as the underlying modularity and community structure is proposed in this research.

4 Methodology

In the section, the dataset for constructing PLEN, algorithms for network analysis and coding tools for obtaining results will be introduced.

4.1 Dataset

The PLEN used in the analysis is a directed network, constructed based on the dataset from [6]. The dataset is converted from a PDF file into a text file with the format of every entries unchanged. There are two types of entry format such that:

- `Year "bash_(unix_shell)" "1989"`
  This defines the node in the network. `bash_(unix_shell)`, which is a programming language, is the node itself with `1989` as an attribute describing its year of release.

- `Cite "perl" "smalltalk"`
  This defines the edge in the network. Based on the source [6], this means that

`perl` is influenced by `smalltalk`. Thus, the directed edge is connected such that `smalltalk -> perl` as illustrated in Figure 1.

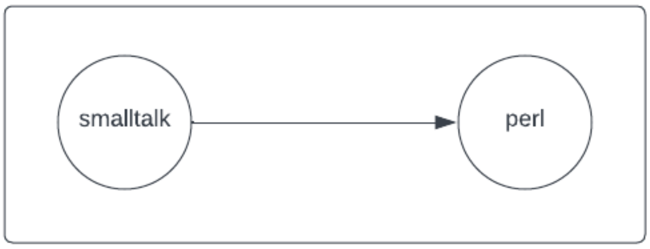

Figure 1: An illustration of an edge connecting two nodes (i.e., programming languages). In the example shown in the figure, an edge is directed from `smalltalk` to `perl` such that `perl` evolved from `smalltalk` and/or `perl` referenced `smalltalk`.

In Table 1, the statistical properties of the constructed PLEN is shown. Both in and out degree distributions are plotted in Figure 2. The trend of the line plot suggests that PLEN follows a power-law distribution. Moreover, the average clustering coefficient of 0.1011, which is close to 0 suggested that the nodes in PLEN have very minimal neighbourhood mixing. Assuming we have three nodes that are connected such that $A -> B -> C$, it is logical such that a language C which is inspired by B will highly unlikely reference A because the language B is equipped with advantageous features from A. On the other hand, the average assortative coefficient in PLEN shows neutral mixing due to its value (-0.0283) being close to 0.

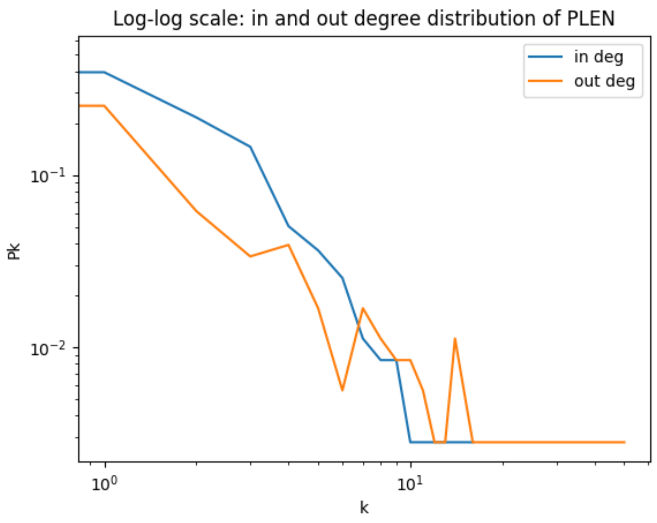

Figure 2: Line plot of the in and out degree distributions of PLEN on log-log scale, where the trends exhibit power-law distribution.

Table 1: Statistical properties of PLEN. The average clustering coefficient of 0.1011 shows minimal neighbourhood mixing whereas the average assortative coefficient of -0.0283 close to zero shows neutral mixing.

Property	**Value**
Total Nodes	357
Total Links	758
Average Clustering Coefficient	0.1011
Average Assortative Coefficient	-0.0283

4.2 Methods and Algorithms

To determine the PLs that have significant contribution to developments of new PLs, higher importance is given for the analysis on outward links. Thus, the betweenness centrality, hub value from the HITS algorithm, and PageRank will be used in the network analysis. Next, modularity will be used in evaluating community structures within PLEN.

4.2.1 Betweenness Centrality

Betweenness centrality of a node u, $C_B(u)$ measures the fraction of shortest path that passes through u among all available shortest paths. It is denoted by the equation:

$$C_B(u) = \sum_{a,b} \frac{s(a,b \mid u)}{s(a,b)} \tag{1}$$

where $s(a,b)$ is the number of shortest path connecting a and b whereas $s(a,b \mid u)$ is the number of shortest path connecting a and b given that it is passing u.

The metric is important as it helps in determining the set of nodes which are gathering information and connecting PL pairs to ensure that advancement is feasible.

4.2.2 PageRank

PageRank is a kind of eigenvector centrality measures. It is motivated from the idea of "how likely a random surfer will visit a website", given a sufficiently long web surfing duration. Thus, PageRank denotes the reachability and popularity of a node in the form of probability. Similar to HITS, the algorithm has to be computed in several iterations until convergence.

$$pagerank = (1-\beta) \sum_{i \to j} \frac{pagerank_i}{d_i} + \beta \frac{1}{n} \tag{2}$$

where β is the probability that a node is followed at random, $pagerank_i$ refers to the PageRank score for node i, d_i is the number of out-degrees for node i, and n is the total number of nodes in the network.

4.2.3 Hub from the HITS Algorithm

The HITS algorithm consists of two parts: hub and authority. Hubs are nodes that have valuable outward links whereas authority is a node which has valuable inward links. In this work, the hub value will be analysed for understanding the set of nodes that contributed to PL developments.

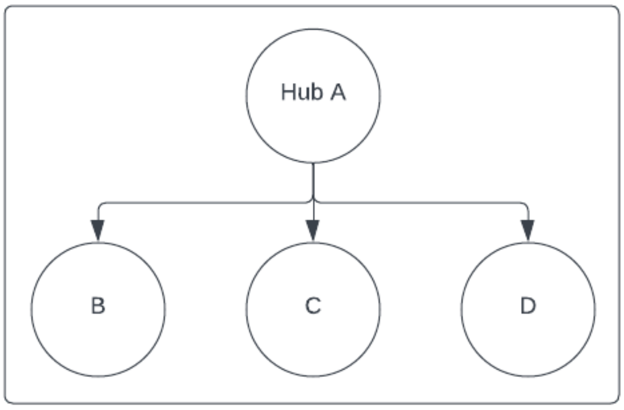

Figure 3: An example of a hub node. Node A is a hub since it links to many nodes such as B, C and D.

The hub value of a node u is calculated in several iterations by summing up the authority scores of the nodes that u is linking to in each iteration. The hub value then converges after sufficient iterations or when a threshold is met. The equation is shown below [7].

$$hub(u) = \sum_{v \in u_{out}} authority(v) \tag{3}$$

where v are nodes from u_{out} which is the set of outward connecting nodes of node u.

4.2.4 Modularity

Modularity measures the quality of a community partition within a network. With values within the range of $[-1, 1]$, positive values means that the number of edges within a group exceeds expectation, whereas a value larger than 0.3 refers to significant community structure. The equation is denoted by:

$$Q(G, S) = \frac{1}{2m} \sum_{s \in S} \sum_{i \in s} \sum_{j \in s} (A_{ij} - \frac{k_i k_j}{2m}) \tag{4}$$

where Q is modularity score, G is a network, S is partitions of communities, m is the total number of edges, and A_{ij} is an adjacency matrix such that $A_{ij} = 1$ is there exists an edge between nodes i and j.

4.3 Tools

The following frameworks and libraries are used in the research:

1. **Programming Language:** Python [8]
2. **Libraries:**
    - **Graph/Network:** NetworkX [9]
    - **Plotting:** Matplotlib [10]
    - **Mathematical Operations:** NumPy [11]
3. **GitHub:** juliagsy/network-research

5 Results

5.1 What is/are the key contributing PL(s) driving evolution in PLEN?

The results of betweenness centrality, hub score and PageRank are respectively shown in Tables 2, 3 and 4 below. It is observed that the rankings of betweenness centrality and hub score are more similar to each other than with that of PageRank. This could be due to the difference in nature of each metric, where hub score weighs outward connections more whereas PageRank better accounts for inward connections of a node. Furthermore, betweenness centrality measures the fraction of shortest path passing through a node, thus both inward and outward connections of a node are important.

From the results, it can be seen that Python is the only PL that outperformed other PLs in all measures. Python came in 2nd in both betweenness centrality and PageRank, and 7th in hub score. Apart from that, Haskell came in 1st in PageRank and 4th in betweenness centrality, whereas C, Java, C++ and Lisp are some of the top-ranked PLs in both betweenness centrality and hub score.

It can be concluded that Python is one of the top contributing PL in PLEN as it has valuable inward connections based on its PageRank score of 0.0125 as well as important outward connections based on its hub score of 0.0385. This made technological advancement possible due to references from other PLs in Python as well as its evolution/contribution in later PLs, which is also due to its good betweenness centrality of 0.0105.

C, Haskell, Java, C++ and Lisp are also considered key PLs in the evolution network. The high score of Haskell in PageRank (ranked 1st) allowed the inference that it has many “page visits” such that it is invented by referencing many valuable PL sources/ancestors. C, Java, C++ and Lisp on the other hand outperformed in hub score. This means that they were important in inspiring PLs developed later in time. This could be due to the fact that they are older PLs, thus making them sources of innovation.

Table 2: PLs with top 8 betweenness centrality.

Rank	PL	Betweenness Centrality
1	C	0.0113
2	Python	0.0105
3	Java	0.0091
4	Haskell	0.0082
5	PL/I	0.0051
6	Javascript	0.0049
7	C++	0.0043
8	Lisp	0.0040

Table 3: PLs with top 8 hub scores.

Rank	PL	Hub
1	C	0.1142
2	Java	0.0755
3	C++	0.0575
4	Lisp	0.0462
5	Smalltalk	0.0425
6	Perl	0.0389
7	Python	0.0385
8	Pascal	0.0359

Table 4: PLs with top 8 PageRank.

Rank	PL	PageRank
1	Haskell	0.0179
2	Python	0.0125
3	Windows Powershell	0.0098
4	Dylan	0.0092
5	Candle	0.0084
6	Eulisp	0.0082
7	Go	0.0078
8	Rust	0.0075

5.2 Do the community structure in PLEN infer PL categorisation?

With the approach of maximising modularity greedily, the resulting community structure is shown in Figure 4. The structure shown achieved a modularity score of $0.5703 > 0.3$, indicating a significant community structure. Due to the amount of nodes and possible overlapping between them, some colours and structures may be hidden away, thus making some details unobvious visually. To account this, Table 5 lists out the top 5 communities along with the results of the key PLs from the first research question.

From the results in Table 5, it can be observed that each community has a majority of PL members from the same category. For instance, C is in the 1st community that represents compiled language along with PL/I, Fortran, and more. The 2nd community can be said to be the group of object-oriented PLs due to members of Java, C++ and Ada 2005, whereas the 3rd community consists of functional PLs. With Lisp in the 4th community, it is a list-based PL community whereas the 5th community being interpreted languages inferring from PLs such as Python, Perl, Pascal and so on.

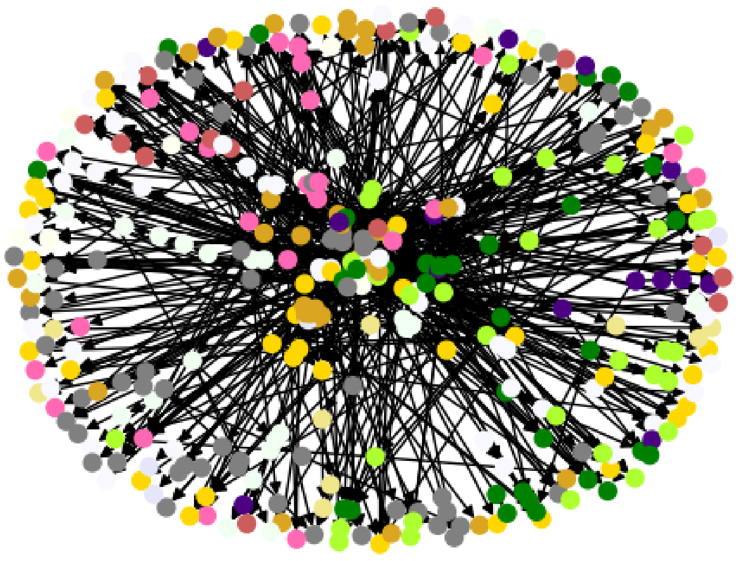

Figure 4: Community structure of PLEN. The structure has a modularity score of 0.5703, showing significant community structure.

Table 5: Top 5 PL communities along with their sizes, key PL and example of other PLs within the community. The results have shown that each community has a key PL representing their category. For example, the 1st community with C is mainly compiled language, the 2nd with Java and C++ represents object-oriented language, the 3rd community with Haskell is functional language, the 4th with Lisp as the key PL are majority list-based PLs, and the 5th community with Python consists of mostly interpreted languages.

Community	**Size**	**Key PL(s)**	**Other Nodes**
1	43	**c**	pl/i, c_shell, bcpl, fortran, harbour_(software), ...
2	40	**java, c++**	optimj, ada_2005, object_pascal, ada_95, eiffel, ...
3	36	**haskell**	hope, ml, erlang, miranda, mercury, ...
4	35	**lisp**	joy, clisp, reduce, factor, nemerle, ...
5	34	**python**	tcl, php, perl, pascal, xc, ...

6 Discussion

The results of this work has successfully shown that the set of top contributing nodes in PL evolution includes Python, C, Haskell, Java, C++ and Lisp due to their high betweenness centrality, hub score (HITS) and/or PageRank. From the comparisons in Table 6, it is also evident that Python further stood out in the set by outperforming the other PL members, making it to the top 8 lists across all metrics.

Table 6: Comparisons of betweenness centrality, hub score and PageRank for the following PLs: Python, C, Haskell, Java, C++ and Lisp. The numbers in bracket indicate their respective rank for the particular metric. Python is the only PL entering all top 8 lists, whereas the other PLs only made it to at most 2 lists.

PL	**Betweenness Centrality (Rank)**	**Hub Score (Rank)**	**PageRank (Rank)**
Python	0.0105 (1)	0.0385 (7)	0.0125 (2)
Haskell	0.0082 (4)	-	0.0179 (1)
C	0.0113 (1)	0.1142 (1)	-
C++	0.0043 (7)	0.0575 (3)	-
Java	0.0091 (3)	0.0755 (2)	-
Lisp	0.0040 (8)	0.0462 (4)	-

Apart from that, the second key finding is PL categorisation through applications of modularity and community structure. The work helped in realising that the grouping of PLs is also strongly associated with its evolution network - PLEN, on top of externally observed functional features and syntax style.

In terms of limitations, the modularity and network community approach to the second research question restricts the PL categorisation to a single group. However, modern PLs are often multi-paradigm, which means that they span across multiple PL categories. Although the result successfully groups PLs into communities of a single type, overlapping communities exist which is not accounted for in this work.

To conclude, future work includes addressing the limitation from possibilities of overlapping communities stated above. Moreover, the network used in the study is considered small such that it consists of less than 1000 nodes and links. Furthermore, the dataset is updated up to the year of 2015 and there may be newly developed PLs. Thus, it is crucial to also include PLs which are absent in the dataset, and attempt experiments on the larger dataset where possible.

Given that polyglot programming and evolutionary thinking [2] are the new trends such that developers are inclined to apply existing knowledge instead of building a work fully from scratch, other network analysis techniques such as KNN could also be investigated to understand the relationship between neighbouring PLs and possibly insights on their cross-compatibility.

References

[1] Wikipedia, *Analytical engine*, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Analytical_engine.

[2] S. Valverde and R. V. Solé, "Punctuated equilibrium in the large-scale evolution of programming languages," *Journal of The Royal Society Interface*, vol. 12, no. 107, p. 20 150 249, 2015.

[3] C. Gillespie, *Graphical display of r package dependencies*, 2011. [Online]. Available: https://csgillespie.wordpress.com/2011/03/23/graphical-display-of-r-package-dependencies/.

[4] M. Maghsoudi, "Uncovering the skillsets required in computer science jobs using social network analysis," *Education and Information Technologies*, pp. 1–22, 2023.

[5] A. Agroskin, E. Lyulina, S. Titov, and V. Kovalenko, "Constructing temporal networks of oss programming language ecosystems," in *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, IEEE, 2023, pp. 663–667.

[6] S. Valverde and R. V. Solé, *Supplementary material 2 (sm2) for "punctuated equilibrium in the large scale evolution of programming languages"*, 2012. [Online]. Available: https://royalsocietypublishing.org/action/downloadSupplement?doi=10.1098%2Frsif.2015.0249&file=rsif20150249supp1.pdf.

[7] Wikipedia, *Hits algorithm*, 2023. [Online]. Available: https://en.wikipedia.org/wiki/HITS_algorithm.

[8] Python, *Python*, 2023. [Online]. Available: https://www.python.org.

[9] A. Hagberg and D. Conway, "Networkx: Network analysis with python," *URL: https://networkx. github. io*, 2020.

[10] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science & engineering*, vol. 9, no. 03, pp. 90–95, 2007.

[11] NumPy, *Numpy*, 2023. [Online]. Available: https://github.com/numpy/numpy.