Força dos Relacionamentos em Redes Heterogêneas do GitHub

Gabriel P. Oliveira, Michele A. Brandão, Mirella M. Moro

Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

{gabrielpoliveira,micheleabrandao,mirella}@dcc.ufmg.br

Abstract. *Social networks represent interactions between individuals in different contexts. In this paper, we model GitHub developers in a heterogeneous network by considering social and technical features. As relationships are not equally important in a network, we propose novel metrics for the strength of relationships. We also apply such metrics in a real ranking application and in a link analysis. The results show the new metrics are not correlated, bringing new information about the relationships. Also, the results of applying such metrics in the ranking task reveal that such study is a step forward to more complex analyses, such as to determine users' influence and popularity, as well as to study team formation and community detection. Finally, in the link analysis, the filter of networks by using social features reduces considerably their size without losing their main characteristics, which reveals a promising contribution to the area of graph sampling.*

Resumo. *Redes sociais representam interações entre indivíduos em diferentes contextos. Neste artigo, modelamos os desenvolvedores do GitHub em uma rede heterogênea, considerando fatores sociais e técnicos. Como os relacionamentos não são igualmente importantes em uma rede, propomos novas métricas para a força dos relacionamentos. Também aplicamos essas métricas em uma aplicação real de ranking e em uma análise temporal dos pares. Os resultados mostram que as novas métricas não estão correlacionadas, trazendo novas informações sobre os relacionamentos. Além disso, os resultados da aplicação de tais métricas no ranking revelam que tal estudo é um primeiro passo em direção a análises mais complexas, como determinar a influência e popularidade dos usuários, bem como para estudar a formação de equipes e a detecção da comunidade. Finalmente, na análise temporal, o filtro das redes utilizando fatores sociais reduz consideravelmente seu tamanho sem perder suas principais características, o que revela uma contribuição promissora para a área de amostragem de grafos.*

1. Introdução

Redes sociais online não são apenas para diversão e conexões pessoais, mas também para trabalho e conexões profissionais. Por exemplo, com dados da plataforma online de desenvolvimento de software GitHub, é possível construir uma rede social onde participantes são desenvolvedores que podem criar/contribuir/compartilhar repositórios e projetos de software [Alves et al. 2016; Batista et al. 2017c]. A partir dos dados, das características topológicas de rede e de métricas diversas é possível extrair informações úteis e novos

conhecimentos sobre tais redes, incluindo pessoas mais influentes, relacionamentos promissores e comunidades. Também é possível melhor compreender o desenvolvimento colaborativo de software bem como aplicar tal conhecimento para aplicações complexas como ranqueamento, recomendação de especialistas e times, entre várias outras [Dabbish et al. 2012; Goyal et al. 2018; Tomasello et al. 2017; Tsay et al. 2014].

Para tais análises, é comum modelar a rede como um grafo, onde nós representam indivíduos e arestas seus relacionamentos. Desse modo, as análises são definidas a partir de cálculos (métricas) sobre os nós e arestas da rede. Para arestas, a *força dos relacionamentos* é uma propriedade central, com ampla aplicabilidade em tarefas de recomendação, detecção de comunidades e ranqueamento [Batista et al. 2017c; Easley and Kleinberg 2010]. Em estudos anteriores, nós medimos a força dos relacionamentos de desenvolvimento de software considerando redes homogêneas (apenas um tipo de nó e de aresta) e características topológicas e semânticas, incluindo tempo [Alves et al. 2016; Batista et al. 2017c; Batista et al. 2017b; Rocha et al. 2016]. Neste artigo, propomos uma modelagem de rede *heterogênea* mais complexa e completa, que explora diferentes características semânticas bem como novas métricas para analisá-las.

Além disso, como já mencionado, uma aplicação de medir a força dos relacionamentos é o ranking. Portanto, também estudamos como as diferentes métricas para força dos relacionamentos classificam pares de desenvolvedores. Tal estudo permite identificar métricas que melhor representem a intensidade real das relações para uma aplicação real. Além disso, identificamos diferentes classes de relacionamento usando um algoritmo existente que considera o aspecto temporal e compara os resultados considerando a rede completa e a filtrada. Por fim, embora nosso foco esteja no GitHub, a maior parte do nosso trabalho (especialmente as métricas) pode ser aplicada ou facilmente mapeada para diferentes redes de colaboração, nas quais as equipes de usuários interagem para atingir objetivos comuns.

Nossas contribuições são assim resumidas. Primeiro, expandimos consideravelmente a coleta dos dados do GitHub para considerar vários tipos de relacionamento e várias linguagens de programação (Seção 3.1). Segundo, propomos uma modelagem de rede *heterogênea* de colaboração do GitHub a qual considera cinco novos relacionamentos além do relacionamento base de colaboração em repositórios (Seção 3.2). Terceiro, propomos um novo conjunto de métricas semânticas para tais relacionamentos (Seção 4). Finalmente, realizamos uma análise comparativa das métricas, e as utilizamos em uma aplicação real de *ranking* de pares de desenvolvedores e de análise temporal dos relacionamentos (Seção 5). De modo geral, a nova modelagem e respectivas métricas definem novidades em relação ao entendimento do processo colaborativo de software, conforme descrito a seguir.

2. Trabalhos Relacionados

Uma rede social é um sistema complexo que representa indivíduos e seus relacionamentos do tipo familiar, profissional e/ou de amizade [Barabási 2016]. Formalmente, tais redes podem ser representadas por um grafo, onde os nós são indivíduos e as arestas representam seus relacionamentos. Diferentes estudos analisam redes sociais, incluindo a qualidade e a formação de comunidades (*clusters*) nas redes [Said et al. 2018], bem como fatores temporais para medir a força dos relacionamentos [Batista et al. 2017b].

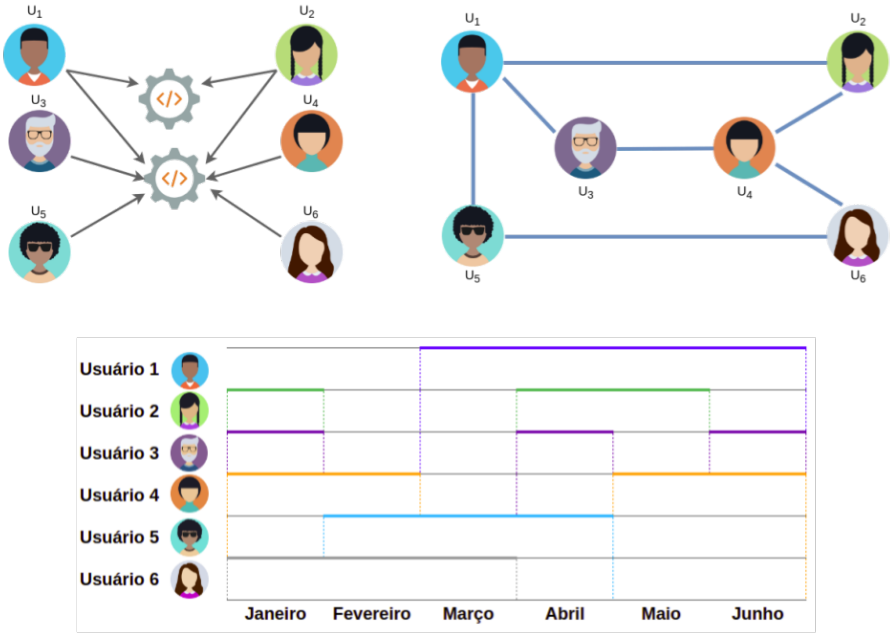


Figura 1. Exemplo da construção da rede homogênea de colaboração no GitHub.

Redes de contextos diferentes apresentam complexidades também distintas. Por exemplo, redes sociais como o GitHub (colaboração entre desenvolvedores de software) permitem diversas interações entre os usuários [Dabbish et al. 2012], disponibilizando uma variedade rica de conhecimento sobre o processo de desenvolvimento em si. Por exemplo, Tsay et al. [2014] investigam que, além de quesitos puramente técnicos como a qualidade do código, fatores *sociais* podem influenciar a aceitação de contribuições de usuários em um repositório. Além disso, Casalnuovo et al. [2015] observam que o relacionamento entre os desenvolvedores é influenciado por experiências *anteriores* com a linguagem de programação e com outros usuários. Outros estudos exploram a rede do GitHub com as mais diversas finalidades, como identificar desenvolvedores e repositórios influentes [Thung et al. 2013] e compreender o comportamento dos desenvolvedores diante de contribuições incomuns [Goyal et al. 2018].

Com foco na força dos relacionamentos, Alves et al. [2016] e Batista et al. [2017b] propõem novas propriedades semânticas para mensurar tal força, contrastando com propriedades já existentes. Especificamente, Alves et al. [2016] definem a rede homogênea de colaboração do GitHub como um grafo $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, onde $\mathcal{V}$ é o conjunto de nós que representam os desenvolvedores e $\mathcal{E}$ é o conjunto de arestas não-direcionadas que lhes conectam quando ambos colaboram em um mesmo repositório. Além disso, Batista et al. [2017c] demonstram que é importante considerar o aspecto temporal em tal modelagem. Portanto, os relacionamentos (arestas) entre desenvolvedores existem apenas se ambos trabalham juntos durante um mesmo intervalo de tempo. A Figura 1 apresenta um exemplo de como a colaboração entre seis desenvolvedores no GitHub é modelada em uma rede homogênea. O peso dessas arestas é calculado por métricas topológicas descritas na Tabela 1, ou semânticas na Tabela 2.

A modelagem homogênea também permite considerar os aspectos temporais dos relacionamentos, definindo o momento em que um relacionamento começa e termina. Tal

Tabela 1. Métricas obtidas através de propriedades topológicas.

Considere para um nó X da rede, $\mathcal{N}(X)$ como o conjunto de vizinhos de X, $w(X)$ como a soma dos pesos das arestas conectadas a X e $w(X,Y)$ como o peso da aresta entre X e Y.

Propriedades topológicas	
Neighborhood Overlap (NO)	Segundo Ealsey e Kleinberg [2010], é uma maneira de medir a força das ligações entre nós através da similaridade de seus vizinhos, sendo portanto $NO_{(X,Y)} = \frac{\lvert\mathcal{N}(X)\cap\mathcal{N}(Y)\rvert}{\lvert\mathcal{N}(X)\cup\mathcal{N}(Y)-\{X,Y\}\rvert}$.
Preferencial Attachment (PA)	Há uma relação linear entre o número de vizinhos de um nó e a sua probabilidade de conectar-se a outro nó [Barabási and Albert 1999]: $PA_{(X,Y)} = \lvert\mathcal{N}(X)\rvert\lvert\mathcal{N}(Y)\rvert$.
Adamic-Adar Coefficient (AA)	Definido por Adamic e Adar [2003], dá maior peso aos vizinhos que não se relacionam com muitos outros: $AA_{(X,Y)} = \sum_{\forall Z\in\mathcal{N}(X)\cap\mathcal{N}(Y)} \frac{1}{log\lvert\mathcal{N}(Z)\rvert}$.
Propriedades topológicas ponderadas	
Tieness (T)*	Brandão e Moro [2017] a definem para medir a força das relações de coautoria. No contexto da rede de colaboração do GitHub: $T_{(X,Y)} = \frac{\frac{\lvert\mathcal{N}(X)\cap\mathcal{N}(Y)\rvert+1}{1+\lvert\mathcal{N}(X)\cup\mathcal{N}(Y)-\{X,Y\}\rvert}\lvert\lvert w(X,Y)\rvert\rvert}{2}$, onde o valor do peso $w(X,Y)$ é normalizado.

** Utilizada em conjunto com cada uma das métricas semânticas da Tabela 2, cujo valor é o peso $w(X,Y)$.*

Tabela 2. Métricas obtidas através de propriedades semânticas.

Propostas por Alves et al. [2016] e Batista et al. [2017b], as métricas semânticas foram divididas aqui em fatores técnicos e sociais. Seja $\mathcal{R}$ o conjunto de todos os repositórios onde dois usuários X e Y colaboram.

Propriedades semânticas para força dos relacionamentos no GitHub	
Number of Shared Repositories (SR)	Definida como o número de repositórios compartilhados entre dois desenvolvedores: $SR_{(X,Y)} = \lvert\mathcal{R}\rvert$.
Jointly Developers Contribution to Shared Repositories (JCSR)	Seja $JCSR_{(X,Y,r_i)}$ a razão entre a quantidade de desenvolvedores envolvidos no relacionamento e o total de desenvolvedores em r_i. Define-se então $JCSR_{(X,Y)} = \frac{\sum_{\forall r_i\in\mathcal{R}} JCSR_{(X,Y,r_i)}}{\lvert\mathcal{R}\rvert}$.
Jointly Developers Commits to Shared Repositories (JCOSR)	Sejam $NC_{(X,r_i)}$ o número de *commits* feitos por um usuário X em um repositório r_i e $NC_{(r_i)}$ o número total de *commits* no repositório r_i: $JCOSR_{(X,Y)} = \sum_{\forall r_i\in\mathcal{R}} \frac{NC_{(X,r_i)}+NC_{(Y,r_i)}}{NC_{(r_i)}}$.
Previous Collaboration (PC)	Seja $ND_{(r_i,t)}$ o número de desenvolvedores no repositório r_i no tempo t, $PC_{(X,Y,t)}$ é definida como a quantidade de colaboração oferecida por X e Y no tempo t: $PC_{(X,Y,t)} = \frac{\sum_{\forall r_i\in\mathcal{R}} \frac{1}{ND_{(r_i,t)}}}{\lvert\mathcal{R}\rvert}$.
Global Potential Contribution (GPC)	Seja $T_{(X,Y,r_i)}$ o intervalo de tempo em que os desenvolvedores X e Y contribuem no repositório r_i e $\mathcal{D}$ o conjunto de todos os desenvolvedores na rede: $GPC_{(X,Y)} = \frac{\sum_{\forall r_i\in\mathcal{R}} T_{(X,Y,r_i)}}{max_{\forall(D_i,D_j)\in\mathcal{D},r_i\in\mathcal{R}} T_{(D_i,D_j,r_i)}}$.

aspecto é considerado por vários estudos (por exemplo, [Batista et al. 2017c; Brandão et al. 2017; Falkowski et al. 2006]) que enfatizam a importância de considerar o tempo para analisar as relações. Dessa forma, alguns algoritmos que classificam esses laços incluem RECAST [Vaz de Melo et al. 2015] e fast-RECAST [Brandão et al. 2017], que considera persistência do relacionamento e amigos em comum (*neighborhood overlap*) para classificar os relacionamentos em quatro classes. Para melhorar esses algoritmos e considerar a intensidade da interação (ou seja, o peso da aresta) na rede, Brandão et al. [2017] propõem o STACY como um algoritmo que classifica a força do empate ao

Tabela 3. Classes de Relacionamento do STACY.

Classe	Frequência da interação	Sobreposição da vizinhança	Intensidade da interação
strong	alta	alta	alta
bridge+	alta	baixa	alta
transient	baixa	alta	alta
periodic	alta	alta	baixa
bursty	baixa	baixa	alta
bridge	alta	baixa	baixa
weak	baixa	alta	baixa
random	baixa	baixa	baixa

longo do tempo considerando: (*i*) persistência de aresta para medir como as relações persistem com o tempo; (*ii*) amigos em comum dado pela sobreposição de vizinhança; e (*ii*) intensidade de interação, isto é, peso de borda. Assim, dependendo da força, os relacionamentos são atribuídos a uma das oito classes apresentadas na Tabela 3.

Entretanto, nenhum desses estudos diferencia fatores técnicos e sociais para medir a força dos relacionamentos em uma rede *heterogênea* de colaboração do GitHub. Os estudos realizados consideram apenas um fator (e.g., a intensidade de contribuições em um repositório) para determinar tal força em uma rede homogênea. A modelagem *heterogênea* proposta aqui permite uma definição mais precisa da força dos relacionamentos, e a análise conjunta dos novos fatores possibilita a descoberta de novos pares relevantes, podendo evidenciar de maneira mais forte usuários influentes na rede.

3. Rede Social de Colaboração

Esta seção descreve a base de dados utilizada para criar a rede social (Seção 3.1) e a modelagem da rede heterogênea de desenvolvimento colaborativo de software (Seção 3.2).

3.1. Base de Dados

A base de dados utilizada é originada do GitSED (*GitHub Socially Enhanced Dataset*) [Batista et al. 2017a], um conjunto de dados do GitHub curado, expandido e enriquecido a partir do GHTorrent [Gousios 2013]. Inicialmente, com dados até setembro de 2015, a base de dados apresenta informações sobre usuários e repositórios, além das métricas topológicas e semânticas existentes para a rede homogênea do GitHub. A versão original do GitSED considera repositórios desenvolvidos em apenas três linguagens de programação.

Dessa forma, expandimos a base de dados para considerar seis linguagens subdivididas em dois grupos, de acordo com seu nível de colaboração definido por Rocha et al. [2016]: linguagens mais colaborativas (JavaScript, Ruby e Python) e linguagens menos colaborativas (Assembly, Pascal e Visual Basic). Assim, pode-se comparar o comportamento das redes dos dois grupos de linguagens diante das novas métricas semânticas propostas na Seção 4. É verificado também se o perfil de colaboração de cada linguagem se mantém. Além disso, utilizando a mesma metodologia de coleta e curadoria, atualizou-se o GitSED com dados obtidos do GHTorrent até maio de 2017. Portanto, tem-se uma versão ampliada do conjunto de dados, disponibilizada publicamente[1], com dados mais recentes e incrementada com as novas métricas semânticas.

[1]Projeto Apoena: `http://www.dcc.ufmg.br/~mirella/projs/apoena`

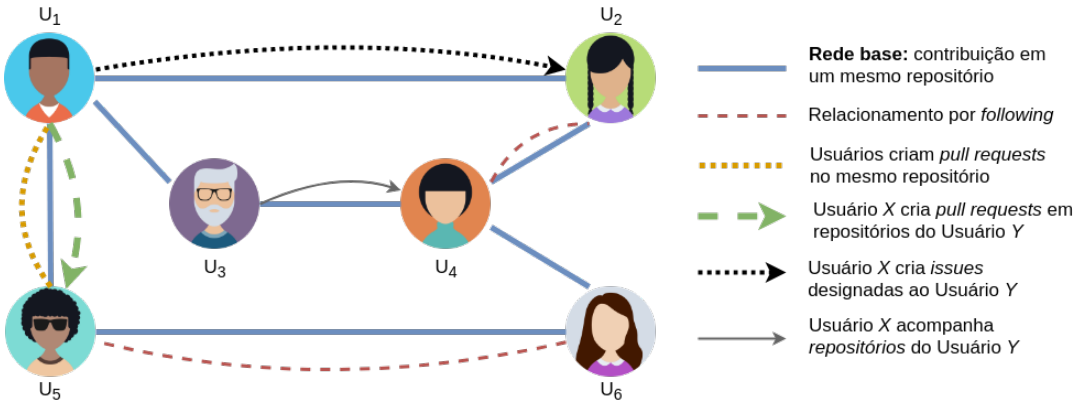


Figura 2. Exemplo da rede heterogênea de colaboração do GitHub com novos tipos de relacionamento considerados.

3.2. Modelagem da Rede

Para modelar a rede heterogênea de colaboração entre desenvolvedores de software do GitHub, utiliza-se como base a rede homogênea $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ da Seção 2. Na nova modelagem, a partir da rede base criada com os relacionamentos de colaboração no mesmo repositório, são adicionados novos tipos de arestas. Assim, a rede se torna *heterogênea* representada por um multigrafo $\mathcal{G}' = (\mathcal{V}, \mathcal{E}_1 \cup \mathcal{E}_2 \cup \cdots \cup \mathcal{E}_6)$, onde o conjunto $\mathcal{V}$ continua contendo vértices que representam os desenvolvedores da rede. No entanto, cada conjunto $\mathcal{E}_k$, para $k \in \{1, \cdots, 6\}$, representa um dos seis tipos de arestas existentes na rede heterogênea: (i) colaboração no mesmo repositório (obrigatório, pois são desenvolvedores-colaboradores); (ii) seguidores; (iii) criação de *pull requests* em um mesmo repositório; (iv) criação de *pull requests* em repositório de outro usuário; (v) criação de *issues* designadas a outro usuário; e (vi) acompanhamento de repositórios favoritos.

O peso de cada conjunto $\mathcal{E}_k$ de arestas corresponde aos valores das métricas propostas por Alves et al. [2016], Batista et al. [2017b] e as novas métricas descritas neste trabalho. A rede possui somente pares de desenvolvedores que contribuíram em um mesmo intervalo de tempo, mantendo apenas relacionamentos que realmente coexistiram [Batista et al. 2017b]. A Figura 2 apresenta a construção da rede heterogênea a partir da rede homogênea apresentada na Seção 2.

4. Novas Propriedades Semânticas

Esta seção apresenta as novas propriedades semânticas para medir a força dos relacionamentos na rede heterogênea considerando fatores específicos provenientes da semântica de conexões que o GitHub oferece para seus usuários. As propriedades existentes consideram apenas a colaboração de usuários em um mesmo repositório e seus aspectos temporais [Alves et al. 2016; Batista et al. 2017c]. Agora, estende-se o conjunto de propriedades considerando os novos tipos de relacionamentos descritos na seção anterior. As novas propriedades estão separadas conforme o tipo de interação que consideram, seja ele técnico (*i.e.* relacionada a questões do código em si) ou social (entre dois usuários).

4.1. Propriedades com Fatores Técnicos

Esta seção apresenta as métricas semânticas que conectam desenvolvedores através de fatores técnicos de desenvolvimento de software presentes no GitHub.

UAI – *Unidirectional Assigned Issues*. Sejam $NI_{(X,Y)}$ o número de *issues*[2] criadas pelo usuário X que são designadas ao usuário Y e $NTI_{(Y)}$ o número total de *issues* designadas ao usuário Y. A nova métrica para o número unidirecional de *issues* é:

$$UAI_{(X,Y)} = \frac{NI_{(X,Y)}}{NTI_{(Y)}}$$

Por exemplo, na Figura 2, se das 10 *issues* designadas a U_2, três são criadas por U_1, então $UAI_{(U_1,U_2)} = \frac{3}{10} = 0,3$.

UPR – *Unidirectional Pull Requests*. Sejam $PR_{(X,Y)}$ o número de *pull requests*[3] que o usuário X cria em repositórios do usuário Y e $TPR_{(Y)}$ o número total de *pull requests* que os repositórios do usuário Y possuem. A métrica unidirecional para *pull requests* é:

$$UPR_{(X,Y)} = \frac{PR_{(X,Y)}}{TPR_{(Y)}}$$

Por exemplo, se U_1 cria seis *pull requests* em repositórios pertencentes a U_5 e o total de *pull requests* nos repositórios de U_5 é 10, então $UPR_{(U_1,U_5)} = \frac{6}{10} = 0,6$.

BPR – *Bidirectional Pull Requests*. Sejam $NPR_{(X,r)}$ o número de *pull requests* que o usuário X cria em um repositório r, $NTPR_{(r)}$ o número total de *pull requests* em r e $\mathcal{U}$ o conjunto universo dos repositórios existentes na base. Para cada par (X, Y) em um repositório $r \in \mathcal{U}$, se $NPR_{(X,r)} \neq 0$ e $NPR_{(Y,r)} \neq 0$, a métrica BPR é definida por:

$$BPR_{(X,Y)} = \sum_{\forall r_i \in \mathcal{U}} \frac{NPR_{(X,r_i)} + NPR_{(Y,r_i)}}{NTPR_{(r_i)}}$$

Por exemplo, na Figura 2, considere que no repositório r_1, dos 10 *pull requests* existentes, dois foram criados por U_1 e um por U_5. Além disso, no repositório r_2 existam 20 *pull requests*, dos quais cinco foram criados por U_1 e 10 por U_5. Assim, $BPR_{(U_1,U_5)} = \frac{2+1}{10} + \frac{5+10}{20} = 0,3 + 0,75 = 1,05$.

4.2. Propriedades com Fatores Sociais

Esta seção introduz as métricas que consideram fatores sociais da relação *direta* entre dois usuários através de funcionalidades oferecidas pelo GitHub.

BIF – *Bidirectional Intensity of Followers*. A intensidade de seguidores é definida com base na relação onde um usuário X *segue* um usuário Y no GitHub. Propõe-se os seguintes valores para medir tal intensidade:

$$BIF_{(X,Y)} = \begin{cases} 1 & \text{se } X \text{ segue } Y \textbf{ \textit{AND} } Y \text{ segue } X \\ 0,5 & \text{se } X \text{ segue } Y \textbf{ \textit{XOR} } Y \text{ segue } X \\ 0 & \text{caso contrário} \end{cases}$$

Na Figura 2, se U_5 segue U_6 mas U_6 não segue U_5, o valor de $BIF_{(U_5,U_6)}$ é $0,5$. Se U_2 segue e é seguida por U_4, $BIF_{(U_5,U_6)} = 1$.

[2] *Issues:* é uma funcionalidade do GitHub que permite compartilhar comentários sobre melhorias ou problemas (*bugs*) encontrados nos projetos de software.

[3] *Pull requests*: é uma funcionalidade do GitHub através da qual usuários pedem para incorporar trechos de código em um repositório após serem avaliados.

Tabela 4. Estatísticas das redes das linguagens mais colaborativas.

	JavaScript	Python	Ruby
Número de repositórios	6.767.297	3.074.827	2.536.133
Número de nós (desenvolvedores)	854.255	519.771	279.281
Número de arestas	2.571.154	3.699.096	33.979.590
Densidade (10^{-3})	0,007	0,027	0,871
Grau médio	6,02	14,23	243,34
Coeficiente de Clusterização Médio	0,358	0,384	0,429
Número de nós no GC*	379.637 (44,4%)	259.355 (49,9%)	180.175 (64,5%)
Número de arestas no GC*	2.105.747 (81,9%)	3.282.140 (88,7%)	33.873.748 (99,7%)

** Giant Component (GC): maior componente conectado de um grafo*

UIM – *Unidirectional Intensity of starMarks*. Sejam $NS_{(X,Y)}$ o número de repositórios de um usuário Y nos quais X tem interesse (clicando no botão *star*) e $NRS_{(X)}$ o número total de repositórios nos quais X está interessado. A intensidade unidirecional de estrelas é definida como:

$$UIM_{(X,Y)} = \frac{NS_{(X,Y)}}{NRS_{(X)}}$$

Na Figura 2, considere que U_3 acompanha um total de 50 repositórios, dos quais 10 pertencem a U_4. Então, $UIM_{(U_3,U_4)} = \frac{10}{50} = 0,2$.

5. Resultados

Neste trabalho, cada linguagem de programação considerada possui sua própria rede heterogênea de colaboração. Desse modo, é possível analisar o comportamento das novas métricas nas diferentes linguagens individualmente bem como compará-las entre as mais e menos colaborativas. Portanto, esta seção apresenta os resultados obtidos na caracterização das redes de colaboração de cada linguagem (Seção 5.1), no estudo da correlação das diferentes métricas para a força dos relacionamentos (Seções 5.2 e 5.3), no ranqueamento dos pares de desenvolvedores (Seção 5.4) e na classificação dos pares (Seção 5.5).

5.1. Caracterização das Redes Heterogêneas de Colaboração

A análise de diferentes propriedades das redes de colaboração permite melhor entender o comportamento das novas métricas nas redes de cada linguagem de programação. Assim, analisamos as seis linguagens de programação para verificar se o nível de colaboração de cada uma se mantém em relação à classificação feita por Rocha et al. [2016]. As Tabelas 4 e 5 mostram os resultados dessa caracterização.

Observando as linguagens mais colaborativas na Tabela 4, percebe-se que as três possuem um alto índice de nós e arestas em seus maiores componentes conectados (*Giant Component* – GC). Em todas elas, mais de 80% das arestas da rede estão presentes no GC. Portanto, verifica-se que essas redes são bem conectadas, confirmando a classificação realizada por Rocha et al. [2016]. Ademais, ao analisar a rede de JavaScript, observa-se que apesar de possuir o maior número de nós e de repositórios, é a rede que apresenta menor número de arestas, menor densidade e menor grau médio dos nós, nesta tabela. Tal resultado indica que os desenvolvedores dessa linguagem não colaboram tanto entre si quanto os das outras linguagens. Por outro lado, Ruby é a linguagem com o menor

Tabela 5. Estatísticas das redes das linguagens menos colaborativas.

	Assembly	Pascal	Visual Basic
Número de repositórios	35.073	20.330	33.275
Número de nós (desenvolvedores)	7.516	3.520	5.602
Número de arestas	14.906	9.377	7.205
Densidade (10^{-3})	0,528	1,514	0,459
Grau médio	3,97	5,3	2,57
Coeficiente de Clusterização Médio	0,354	0,374	0,311
Número de nós no GC*	483 (6,4%)	577 (16,4%)	95 (1,7%)
Número de arestas no GC*	3.335 (22,3%)	5.140 (54,8%)	1.368 (19%)

** Giant Component (GC): maior componente conectado de um grafo*

número de nós e repositórios na rede, mas possui uma quantidade de arestas mais de dez vezes maior que JavaScript. Tal comportamento gera um alto grau médio dos nós e a maior densidade das redes estudadas. Assim, conclui-se que os desenvolvedores de Ruby colaboram muito entre si. Além disso, quase todas as arestas da rede de Ruby estão no GC, de forma que essa é a linguagem mais colaborativa estudada neste trabalho.

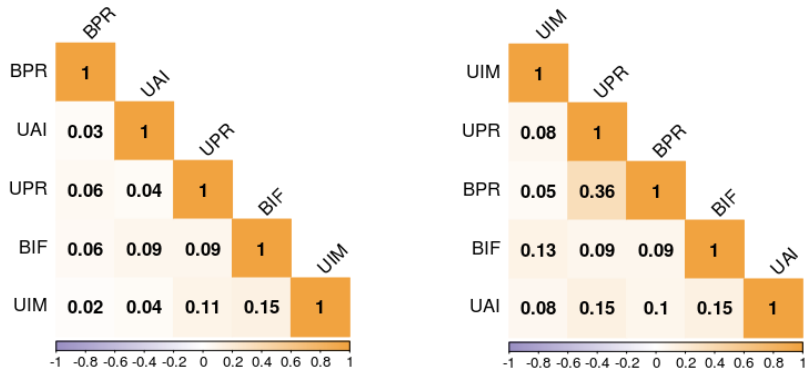
Analisando as linguagens menos colaborativas, apresentadas na Tabela 5, observa-se que Assembly, Visual Basic e Pascal possuem comportamentos semelhantes em suas redes. Todas elas apresentam um baixo grau médio e um baixo índice de arestas no GC. Portanto, infere-se que os repositórios dessas linguagens são em sua maioria compostos por poucos desenvolvedores, que colaboram pouco entre si. Dessa forma, nas análises a seguir, são consideradas apenas Ruby e Visual Basic como representantes das classes de linguagens de programação mais e menos colaborativas.

5.2. Análise das Novas Propriedades Semânticas

Neste trabalho, propomos cinco novas métricas que consideram diferentes propriedades semânticas para medir a força dos relacionamentos na rede heterogênea de colaboração do GitHub. Para verificar a independência entre as novas métricas, a correlação entre elas é analisada pelo uso dos coeficientes de Pearson e Spearman (verificam relações lineares e monotônicas entre as métricas, respectivamente). Por limitações de espaço, são apresentadas apenas as correlações obtidas por Spearman, que são similares às geradas pelo coeficiente de Pearson. A Figura 3 apresenta a matriz de correlação entre as cinco novas métricas para as redes de Ruby e Visual Basic.

Em todas as linguagens, observa-se que a correlação entre as novas métricas é baixa ou insubstancial, com valores próximos a zero. Tal resultado pode ser explicado pelo fato de que cada métrica considera fatores diferentes para calcular a interação entre os pares. A exceção está nas métricas UPR e BPR, onde ambas analisam a interação através da criação de *pull requests*. No entanto, as duas métricas representam características diferentes dos relacionamentos. UPR considera apenas uma fração do conjunto total de *pull requests*, aqueles criados em repositórios de um determinado usuário. Enquanto que BPR considera todos os repositórios da base e analisa a quantidade de *pull requests* feitos pelo par de desenvolvedores.

A baixa correlação entre as métricas é um forte indicador de que as novas propriedades semânticas adicionam novas informações à rede de colaboração do GitHub. Portanto, todas elas podem ser consideradas ao medir a força desses relacionamentos.



(a) Ruby (b) Visual Basic

Figura 3. Coeficiente de correlação de Spearman entre as novas propriedades semânticas: BIF, UAI, UPR, BPR e UIM.

Tabela 6. Participação na rede a partir de métricas selecionadas.

Métrica	# de desenvolvedores		# de arestas	
	Ruby	Visual Basic	Ruby	Visual Basic
Rede completa	279.281 (100%)	5.602 (100%)	33.979.590 (100%)	7.205 (100%)
UAI - *Unidirecional Assigned Issues*	9.982 (3,57%)	124 (2,21%)	8.517 (0,02%)	82 (1,14%)
UPR - *Unidirecional Pull Requests*	14.811 (5,3%)	166 (2,96%)	10.927 (0,03%)	104 (1,44%)
BPR - *Bidirectional Pull Requests*	57.604 (20,63%)	453 (8,09%)	1.697.256 (5%)	391 (5,43%)
BIF - *Bidirecional Intensity of Followers*	58.715 (21,02%)	923 (16,48%)	102.736 (0,3%)	616 (8,55%)
UIM - *Unidirecional Intensity of starMarks*	13.248 (4,74%)	107 (1,91%)	15.577 (0,04%)	56 (0,78%)

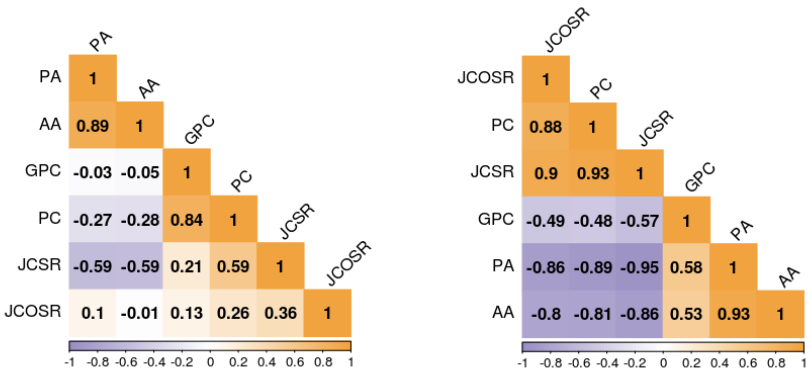
Observa-se também que as métricas técnicas e sociais apresentam baixa correlação entre si. Logo, apesar de terem sido agrupadas pelo tipo e por suas definições, as novas métricas devem ser avaliadas individualmente para medir a interação entre desenvolvedores.

Ademais, para entender o quanto cada tipo de relacionamento representa da rede, a Tabela 6 mostra a proporção de desenvolvedores e arestas presentes quando são retirados os pares que não contêm o tipo de relacionamento representado por cada métrica. Os resultados mostram que poucos nós e arestas permanecem na rede para métricas com fatores sociais (BIF e UIM). Ou seja, desenvolvedores do GitHub não consideram tais funcionalidades quando colaboram em um repositório. Em relação às métricas com fatores técnicos (UAI, UPR e BPR), o baixo número de nós e arestas é explicado pelo fato de que grande parte das *issues* e *pull requests* nos repositórios provém de usuários externos, i.e. não colaboradores. Tal comportamento pode ocorrer por ser mais provável que usuários da aplicação ou desenvolvedores de repositórios que fizeram *fork*[4] reportem problemas e façam *pull requests* no repositório base.

5.3. Correlação com as Métricas Existentes

Esta seção apresenta uma análise da correlação entre as novas métricas com as propriedades já estudadas no contexto do GitHub. Dessa forma, também utilizamos aqui o coeficiente de Pearson e Spearman para analisar a correlação entre as métricas existentes

[4]*Fork*: é a cópia de um repositório de modo que testes podem ser feitos no código sem alterar a versão original (repositório base).



(a) Ruby (b) Visual Basic

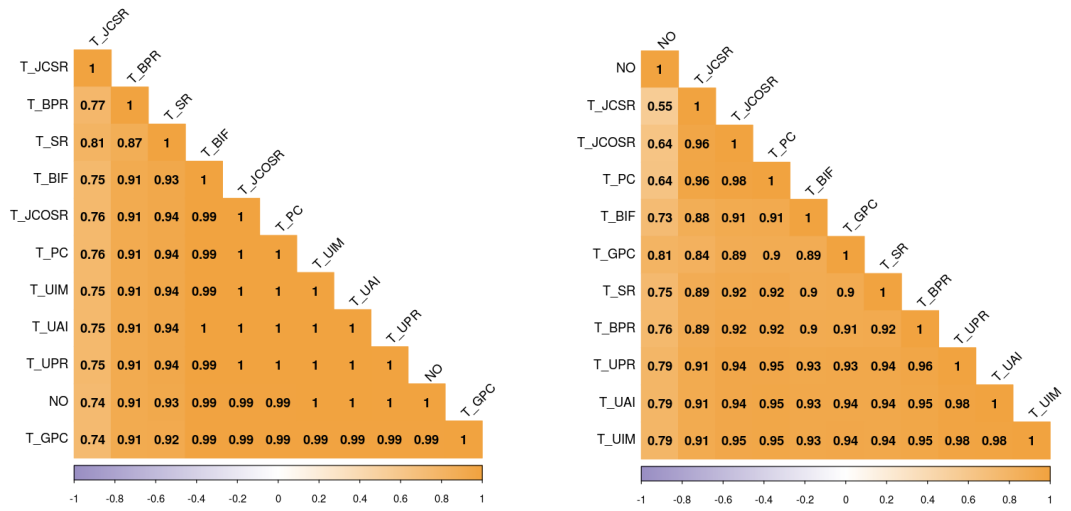
Figura 4. Coeficiente de correlação de Spearman para propriedades inversamente correlacionadas.

(NO, PA, AA, T, PC, GPC, SR, JCSR, JCOSR) e as propostas (BPR, UAI, UPR, BIF e UIM). Os resultados são similares para os dois coeficientes de correlação e para todas as linguagens de programação, tanto as menos quanto as mais colaborativas, com algumas exceções.

Não há correlação significativa linear ou monotônica entre a maioria das propriedades para a força dos relacionamentos. Alguma das exceções são AA e PA com correlação acima de 0,87 considerando todas as linguagens, conforme mostrado na Figura 5. Portanto, se PA e AA estão correlacionadas, então é esperado que um métrica correlacionada com AA também esteja com PA e vice-versa. Outra exceção é JCSR (representa a contribuição conjunta em repositórios) que apresenta correlação negativa no intervalo [-0,95;-0,57] com PA e AA para todas as linguagens. De modo geral, PA e AA representam a tendência de desenvolvedores a se conectarem com outros baseando-se na quantidade de vizinhos em comum. Assim, tal correlação negativa pode indicar que pares de desenvolvedores com intensa contribuição conjunta em repositórios podem não se conectar com muitos outros desenvolvedores.

Uma diferença significativa entre as linguagens mais e menos colaborativas é a existência de correlação negativa no intervalo [-0,9;-0,4] entre JCOSR com GPC, PA, AA e NO para Assembly, Pascal e Visual Basic. As métricas PA, AA e NO, de maneira geral, representam o quanto dois desenvolvedores estão conectados em relação a seus vizinhos em comuns. Já a métrica JCOSR representa a contribuição conjunta de um par de desenvolvedores considerando a quantidade de *commits*. Assim, a forte correlação negativa entre essas métricas indica que nas linguagens menos colaborativas, os relacionamentos de um par de desenvolvedores com seus vizinhos influenciam negativamente à contribuição conjunta por *commits*. Por outro lado, a correlação negativa entre JCOSR e GPC indica que pares de desenvolvedores tendem a não contribuir por muito tempo.

A Figura 5 mostra uma forte correlação entre as métricas *Tieness* ponderadas com todas as propriedades semânticas estudadas. Note que tal métrica permite a combinação entre a propriedade topológica da rede com uma propriedade semântica, ao considerá-la como peso. A alta correlação indica que a *Tieness* com diferentes pesos traz as mesmas



(a) Ruby (b) Visual Basic

Figura 5. Coeficiente de correlação de Spearman para as propriedades mais bem correlacionadas.

Tabela 7. *Ranking* de pares de desenvolvedores ordenados pela métrica BPR.

#	Ruby							Visual Basic						
	D1	D2	BPR	T_BPR	NO	SR	GPC	D1	D2	BPR	T_BPR	NO	SR	GPC
1	01	02	139,816	0,078	0,144	45	0,051	20	21	5,745	0,234	0,424	8	0,372
2	03	02	138,762	0,091	0,173	149	0,204	22	23	5	0	0	5	0,144
3	04	02	99,879	0,088	0,193	26	0,04	24	25	4,07	0,427	0,6	8	0,104
4	05	02	77,652	0,061	0,148	17	0,028	26	27	4	0,318	0,556	9	0,193
5	06	07	76,332	0,19	0,486	204	0,992	20	28	3,828	0,221	0,485	6	0,368
6	08	09	64,942	0,053	0,144	36	0,128	24	29	3,186	0,291	0,4	9	0,251
7	10	02	62,149	0,064	0,167	22	0,039	20	30	2,75	0,071	0,156	3	0,121
8	11	07	61,774	0,208	0,58	196	0,838	31	32	2,685	0,122	0	2	0,196
9	12	02	50,568	0,002	0,005	19	0,037	33	34	2	0	0	1	0,132
10	03	04	49,306	0,081	0,234	33	0,055	35	36	2	0,337	0,334	3	0,007

informações à análise da força dos relacionamentos, e assim, pode-se escolher apenas uma delas para medir tal força. Nesse caso, recomenda-se o uso de métricas com baixo custo computacional, como T_BIF ou T_SR (combinação de T com BIF e T com SR, respectivamente). Há também uma alta correlação com NO, indicando que esta não precisa ser considerada quando *Tieness* é utilizada. Esses resultados reforçam que a forma como um par de desenvolvedores está na rede de colaboração em relação a seus vizinhos influencia fortemente os relacionamentos.

5.4. Ranqueamento de Pares de Desenvolvedores

Como um exemplo de aplicação das novas métricas, podemos ordenar os pares de desenvolvedores através de uma métrica selecionada. A Tabela 7 mostra o *ranking* de pares[5] ordenados através da métrica BPR para as redes de Ruby e Visual Basic. Tal métrica foi escolhida por possuir o maior número de pares com valor relativamente alto. Para efeito de comparação, adicionamos também as métricas T_BPR, NO, SR e GPC.

O *ranking* mostra que BPR captura alguns relacionamentos muito fortes na rede.

[5]Os usuários foram anonimizados por questões de privacidade.

Por exemplo, na rede de Ruby, o par $\{06,07\}$ possui o maior valor da métrica SR em toda a rede, compartilhando 204 repositórios. Além disso, esse mesmo par possui um dos maiores tempos de colaboração da rede, com o valor da métrica GPC próximo de 1. Da mesma forma, na rede de Visual Basic, os pares do *ranking* possuem um valor considerável para as métricas SR e GPC. Tal resultado revela que *pull requests* são uma boa forma de classificar os relacionamentos entre os desenvolvedores, e a BPR pode ser utilizada em um modelo para a força dos relacionamentos.

Dentre as métricas apresentadas, BPR é a única que não possui valores no intervalo $[0, 1]$. Tais valores não são normalizados pois esse processo causa a perda de informações sobre a força dos relacionamentos. Por exemplo, comparando o primeiro par dos *rankings* de Ruby e Visual Basic, percebe-se que o par $\{01,02\}$ possui um valor de BPR quase 28 vezes maior do que o par $\{20,21\}$. Portanto, conclui-se que o comportamento da métrica é o mesmo, mas os valores máximos são muito diferentes. Assim, BPR pode ser usada para indicar o nível de colaboração local (de pares de desenvolvedores) e global (de linguagens de programação).

Finalmente, é importante notar que alguns usuários estão presentes em vários pares nos dois *rankings*. Por exemplo, em Ruby, o Usuário 2 está presente em seis dos dez pares da Tabela 7. Da mesma forma, os usuários 20 e 24 estão presente em dois dos dez pares do *ranking* de Visual Basic. Tal informação pode indicar que tais usuários são influentes na rede de suas respectivas linguagens, e assim, as novas métricas podem ser uma importante ferramenta para definir a popularidade e a influência de usuários, bem como identificar possíveis perfis de colaboração.

5.5. Análise Temporal e Classificação dos Pares

O aspecto temporal dos relacionamentos no GitHub é muito importante, pois está presente na definição da rede de colaboração por si só. Em geral, alguns relacionamentos persistem mais que outros, o que pode influenciar na dinâmica da rede. De fato, as relações são classificadas aqui de acordo com sua força, considerando também a noção de temporalidade. Este estudo permite entender melhor o comportamento dos desenvolvedores na plataforma.

Os relacionamentos são classificados pelo algoritmo STACY (*Strength of Ties Automatic-Classifier over the Years*), apresentado na Seção 2. Em relação à sua entrada, as redes de cada linguagem de programação têm colaborações desde outubro de 2007 (primeiro commit no GitHub), com intervalos de tempo de 15 dias. Então, o número de repositórios compartilhados (ou seja, a métrica SR) representa a frequência de interação. Tal métrica foi escolhida porque seu valor é sempre um valor inteiro e absoluto, o que é apropriado para a entrada do STACY. O algoritmo atribui cada par de desenvolvedores na rede a uma classe de relacionamento, variando de relacionamentos fortes a aleatórios. A figura 6 apresenta os resultados gerados pelo STACY para as redes de Python e Assembly, representando as linguagens mais e menos colaborativas, respectivamente. Os resultados são semelhantes para as outras linguagens consideradas neste estudo.

Observe que a maioria dos relacionamentos é fraco nas linguagens de programação mais e menos colaborativas, o que significa que esses relacionamentos não persistem com o tempo e não têm uma alta frequência de interação, apesar de terem uma grande sobreposição de vizinhança. Portanto, a alta presença de relacionamentos fracos

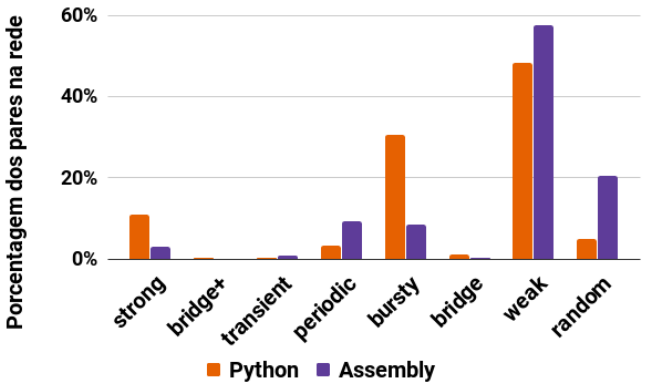


Figura 6. Pares de desenvolvedores classificados pela força dos seus relacionamentos através do algoritmo STACY.

não depende da linguagem de programação, mas reflete as características do modelo de rede e do próprio GitHub. Além disso, muitos desenvolvedores contribuem brevemente para os repositórios, o que revela interesses específicos em seu comportamento de desenvolvimento.

Além disso, as linguagens colaborativas têm uma proporção maior de relacionamentos *strong* e *bursty*. Essas classes revelam uma alta frequência de interação entre pares de desenvolvedores. Esse resultado mostra que mais linguagens colaborativas tendem a ter mais desenvolvedores trabalhando juntos em repositórios, tendo mais pares com valores de SR mais altos. Por outro lado, as linguagens menos colaborativas têm mais desenvolvedores envolvidos em um relacionamento *periodic*, *weak* ou *random*. Essas classes compartilham uma baixa frequência de interação, o que revela que há poucos desenvolvedores compartilhando repositórios. Este fato reforça a classificação dessas linguagens como menos colaborativa e a caracterização das redes realizadas na seção 5.1.

Para verificar como os fatores sociais influenciam a classificação dos relacionamentos, as redes de colaboração são filtradas para conter apenas os pares de desenvolvedores que possuem pelo menos um tipo de interação *social*, representada pelas métricas BIF e UIM. A Figura 7 mostra uma comparação dos resultados STACY para as redes de linguagem Python completas e filtradas. Observe que os resultados são semelhantes às outras linguagens de programação.

A análise das redes completa e filtrada revela que a proporção de relacionamentos *strong* é a mesma para ambos. Além disso, há uma proporção maior de relacionamentos *bridge+* e *transient* na rede filtrada. O fato dessas três classes terem uma alta intensidade de colaboração revela que fatores sociais são importantes para a construção de relacionamentos mais intensos na rede. Além disso, esse resultado indica que o filtro das redes, considerando as métricas sociais, mantém os relacionamentos fortes nas redes, o que mostra uma direção para reduzir o número de relacionamentos e, posteriormente, permitir a aplicação de algoritmos com maior custo computacional. Por outro lado, a proporção de relacionamentos *random* é consideravelmente maior na rede filtrada, enquanto a proporção de relacionamentos *weak* diminui. Essas classes diferem apenas no nível de sobreposição de vizinhança, que é alto em relacionamentos *weak* e baixo

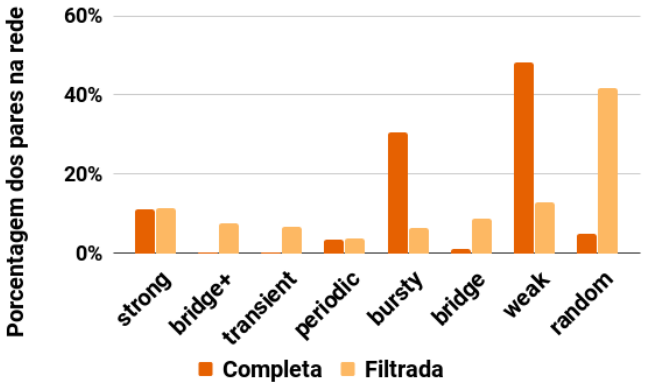


Figura 7. Resultados do STACY para a rede de colaboração de Python filtrada por interações sociais.

em relacionamentos *random*. Isso ocorre porque vários pares no mesmo repositório (que contribuem para uma sobreposição de vizinhança maior) foram removidos da rede filtrada porque não possuem interações sociais.

6. Conclusão e Trabalhos Futuros

Este trabalho apresentou uma modelagem heterogênea para a rede de desenvolvedores no GitHub, e cinco novas métricas semânticas para a força dos relacionamentos, que permitem avaliar redes construídas a partir de linguagens de programação mais e menos colaborativas. A análise dessas métricas revelou uma falta de correlação linear e monotônica entre elas, ou seja, todas elas representam novas informações sobre as relações. Além disso, pode-se dizer que não há muita interação social em colaborações em repositórios de software.

Além disso, as novas métricas foram utilizadas para classificar pares de desenvolvedores, e os resultados revelaram que os novos tipos de interação podem capturar desenvolvedores com uma alta capacidade de colaboração na rede. Além disso, a análise dos relacionamentos considerando o aspecto temporal mostrou que as linguagens mais colaborativas têm uma proporção maior de relacionamentos fortes, que tendem a persistir ao longo do tempo. Por fim, também foi mostrado que as interações sociais no GitHub podem contribuir para colaborações mais intensas.

Como trabalhos futuros, planeja-se estudar outros fatores sociais que geram a interação entre desenvolvedores e analisar como eles influenciam na previsão de links. Também pretendemos investigar a relação entre as propriedades semânticas e a influência de desenvolvedores no GitHub.

Agradecimentos. Trabalho parcialmente financiado por CAPES e CNPq.

Referências

Adamic, L. A. and Adar, E. (2003). Friends and neighbors on the Web. *Social Networks*, 25(3):211 – 230.

Alves, G. B. et al. (2016). The Strength of Social Coding Collaboration on GitHub. In *SBBD - Short Papers*, pages 247–252.

Barabási, A.-L. (2016). *Network science*. Cambridge University Press.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.

Batista, N. A., Alves, G. B., Gonzaga, A. L., and Brandão, M. A. (2017a). GitSED: Um Conjunto de Dados com Informações Sociais Baseado no GitHub. In *SBBD - Dataset Showcase Workshop*, pages 224–233.

Batista, N. A. et al. (2017b). Aspectos Temporais para Medir a Força da Colaboração no GitHub. In *SBBD - Short Papers*, pages 234–239.

Batista, N. A. et al. (2017c). Collaboration Strength Metrics and Analyses on GitHub. In *WI*, pages 170–178.

Brandão, M. A. et al. (2017). Tie Strength Dynamics over Temporal Co-authorship Social Networks. In *WI*, pages 306–313.

Brandão, M. A. and Moro, M. M. (2017). The strength of co-authorship ties through different topological properties. *JBCS*, 23(1).

Brandão, M. A. et al. (2017). STACY: Um Novo Algoritmo para Automaticamente Classificar a Força dos Relacionamentos ao Longo dos Anos. In *SBBD - Full Papers*, pages 136–147.

Casalnuovo, C. et al. (2015). Developer onboarding in GitHub: the role of prior social links and language experience. In *ESEC/FSE*, pages 817–828.

Dabbish, L. A. et al. (2012). Social coding in GitHub: transparency and collaboration in an open software repository. In *CSCW*, pages 1277–1286.

Easley, D. and Kleinberg, J. (2010). *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press.

Falkowski, T. et al. (2006). Mining and visualizing the evolution of subgroups in social networks. In *WI*, pages 52–58.

Gousios, G. (2013). The GHTorrent Dataset and Tool Suite. In *MSR*, pages 233–236.

Goyal, R. et al. (2018). Identifying unusual commits on GitHub. *Journal of Software: Evolution and Process*, 30(1).

Rocha, L. M. A. et al. (2016). Análise da Contribuição para Código entre Repositórios do GitHub. In *SBBD - Short Papers*, pages 103–108.

Said, A. et al. (2018). CC-GA: A clustering coefficient based genetic algorithm for detecting communities in social networks. *Applied Soft Computing*, 63:59–70.

Thung, F. et al. (2013). Network Structure of Social Coding in GitHub. In *CSMR*, pages 323–326.

Tomasello, M. V., Vaccario, G., and Schweitzer, F. (2017). Data-driven modeling of collaboration networks: a cross-domain analysis. *EPJ Data Science*, 6(1):1–25.

Tsay, J. et al. (2014). Influence of social and technical factors for evaluating contribution in GitHub. In *ICSE*, pages 356–366.

Vaz de Melo, P. O. S. et al. (2015). RECAST: Telling apart social and random relationships in dynamic networks. *Performance Evaluation*.