Universidade Federal de Pernambuco

Centro de Informática

Graduação em Ciência da Computação

Sistema de recomendação de usuários para se seguir no GitHub baseado em interesses comuns

Proposta de Trabalho de Graduação

Trabalho apresentado ao Centro de Informática
da UFPE como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação.

Aluno: Marlon Reghert Alves dos Santos

Orientador: Ricardo Bastos Cavalcante Prudêncio

Agradecimentos

Foram os caminhos tortos da vida que vivi que me moldaram desta forma e me fizeram tomar as decisões necessárias para chegar aqui e ter a oportunidade de escrever este texto, não poderia ter sido de outra forma. Por isso, brevemente, tirarei algumas palavras para agradecer aqueles com quem divido as memórias e me ajudaram, direta ou indiretamente, a formar quem eu sou hoje.

Primeiramente agradeço aos meus familiares, que apesar de todos os problemas que enfrentamos, nunca deixaram de me apoiar para concluir esta etapa. Acima de todos os empecilhos, sempre estivemos juntos nos momentos de desespero, em especial meu pai, minha mãe, meus irmãos e minhas tias, só nós sabemos o que passamos juntos. Ainda mais singularmente, agradeço a minha irmã, por ser a maior inspiração em minha vida.

Meus amigos, não poderei citar cada um para não gerar brigas ou ciúmes, mas se sentirá tocado aquele que sabe que esteve ao meu lado em minha trajetória. Amigo aquele que me viu em momentos bons e ruins, e que já esteve perto ou distante, e que num grande resumo, me dá orgulho de saber que com ele dividi meu tempo de vida.

Aos colegas que fiz durante a graduação, em especial os de minha turma (2013.1) e agregados, vocês sem dúvida são parte disto. Foram muitos fins de semana abdicados para estudar e fazer os milhares de projetos, momentos difíceis que não me arrependo.

Todos do PET-Informática, onde fiz grandes amigos e tive a oportunidade de estar junto de alunos excelentes.

À maratona de programação, que talvez represente o ápice da minha graduação. Por muito tempo não achei que faria parte deste programa e nele tive a chance de aprender com alunos alto nível. Em especial ao *Ballions*, time com quem dividir a medalha de prata na final nacional.

A todos da In Loco, empresa na qual iniciei minha carreira profissional e fui muito bem acolhido num ambiente de aprendizado. Assim como aos meus colegas de equipe na atual empresa, Top Free Games, que me deram a oportunidade de iniciar uma nova jornada.

A Tomer Simis, com quem iniciei este trabalho na disciplina de Tópicos Avançados em Inteligência Artificial.

E por fim, é claro, a todos os professores que passaram pela minha vida, pois, escolheram para si um dos caminhos mais desafiadores no Brasil em que vivemos, ensinar.

Recife, Dezembro de 2017

Resumo

As redes sociais estão mudando a forma como o mundo se comunica, e com elas surgem vários desafios computacionais. Entre as principais *features* que uma rede social costuma ter, está a recomendação de conteúdo, podendo esse ser uma página com conteúdos relevantes ou um usuário com interesses em comum para iniciar-se uma amizade. O comportamento dos usuários dentro de uma rede varia de acordo com o contexto desta. Neste trabalho estudaremos o comportamento da relação entre os usuários da plataforma de compartilhamento de código mais famosa entre os desenvolvedores, o GitHub. Criaremos um sistema de recomendação de usuários baseado em *similaridades técnicas* e *distância de amizade*, deixando a ferramenta acessível através de uma interface web, e por fim exibiremos os resultados obtidos.

Abstract

The social networks are changing the way that the world communicate, a lot of computational challenges are emerging with them. Among the main *features* that a social network usually has, there is the content recommendation, that could be a page with relevant content or a user with common interests to start a friendship. Inside a network, the user's behavior varies according to the context of that. In this work we going to study the user's relationship behavior on the most famous codeshare platform among developers, the GitHub. We will create a user recommendation system based on *technical similarity* and *friendship distance*, leaving the platform accessible through an web interface, we will exhibit the obtained results by end.

Lista de Figuras

Capítulo 1

Introdução

1.1 - Contexto

Em nosso atual século vimos a introdução das redes sociais e como elas revolucionaram [1] a forma como nos comunicamos. *Facebook*, *Twitter*, *Google Plus, Instagram,* entre milhares outras, são hoje os nomes mais fortes da atual internet.

O surgimento das redes sociais trazem consigo grandes desafios tecnológicos em várias áreas e mais abertura para a inovação na computação. Entre essas oportunidades, estão os sistemas de recomendação, já presentes em todas essas redes, variando de acordo com a feature que cada uma delas provê. No Facebook, por exemplo, podemos observar a recomendação de amizade[2], sendo hoje uma área da empresa, especializada em melhorar a recomendação, além de outros tópicos como recomendação de Páginas. *Instagram* e *Twitter* também atuam forte para recomendar conteúdo aos seus usuários, em especial, outros usuários para se seguir.

Nas redes acima, a relação entre os usuários são mais fácil de serem observada, e mais fácil vemos as aberturas para criação de sistemas de recomendação de usuários, mas podemos aplicar o mesmo conceito em outras redes, que também manifestam o comportamento de interesse entre usuários. Podemos ver a abertura para aplicação deste conceito no GitHub[3], a plataforma para compartilhamento de código fonte baseada em Git mais usada do mundo.

No GitHub, em paralelo as suas *features* para compartilhamento de código, podemos ver várias características que manifestam o comportamento de rede social[4], por exemplo: Usuários podem seguir outros usuário, e assim, uns podem visualizar as atividades dos outros. Assim podemos ver abertura para o entendimento da motivação deste comportamento dentro dessa rede de características específicas e a criação de uma ferramenta que recomende usuários para serem seguidos.

Neste projeto, vamos usar deste artefato para construir um sistema de recomendação. Nele o usuário poderemos ver uma lista de usuários a serem seguidos, podendo escolher entre visualizar usuários de acordo com a similaridade técnica (pessoas próximas que

compartilham as mesmas linguagens de programação, seguem projetos similares, etc) e distância de amizade (seguem pessoas em comum, provavelmente estudam na mesma universidade ou trabalham na mesma empresa, etc). O sistema poderá ser usado através de uma interface web.

Por fim, vamos descrever os modelos utilizados para criar as recomendações, tecnologias e exibir os resultados obtidos, desde imagens da ferramenta até casos de estudo dos outputs das recomendações obtidas.

1.2 - Objetivo

O objetivo deste trabalho é criar um sistema de recomendação de usuários com base no comportamento social dos usuários do GitHub, sendo eles similaridade técnica e distância de amizade.

Iremos modelar um sistema de recomendação, definindo as variáveis necessárias para atingir um grau de assertividade aceitável para as duas modalidades (similaridade técnica e distância de amizade).

Os dados utilizados serão os metadados provenientes da API pública do GitHub. Com base neles, aplicamos os nossos modelos e os implementamos em uma ferramenta acessível através de uma interface web, onde o usuário escolherá entre ver uma lista de usuários a serem seguidos baseados em similaridade técnica ou distância de amizade, e então, exibiremos os resultados obtidos.

Capítulo 2

Fundamentos e representações

2.1 - Fundamentos

Em 2003, de dentro de um dormitório da universidade de Harvard, 4 alunos lançam um site, acessível apenas na rede interna de sua universidade, o *Facemash*, que anos depois, este protótipo viraria o Facebook[5], a rede social que revolucionou a forma como nos comunicamos.

Esta é apenas uma das histórias por trás do conceito *Rede Social*, como parte da terceira revolução da internet, que definem os atores por trás da forma como interagimos e em cima deste, nasce uma ciência para estudá-la.

Mas antes de falarmos da ciência em cima deste conceito ou dos grandes cases de sucesso como LinkedIn, Facebook ou Myspace, devemos entender o que é uma rede, sendo uma rede social um subtipo desta.

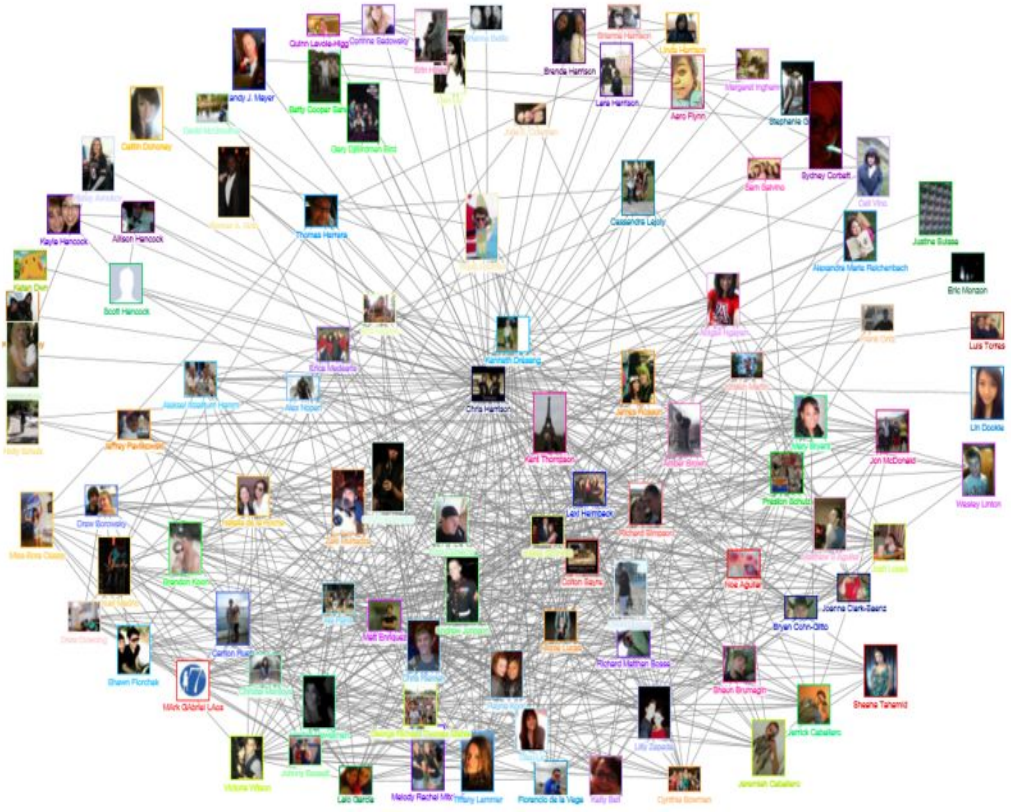
Pela sociologia, podemos entender uma rede como uma estrutura formada por indivíduos que se interagem direta ou indiretamente[6]. Dado o comportamento dos indivíduos nessa estrutura, podemos através de análises buscar padrões locais ou globais para entender ou prever o comportamento dos membros desta rede, ou seja, a dinamicidade da rede.

2.1.1 - Tipos de redes

Quando observamos uma rede, devido às diferentes características dos indivíduos que a compõem e as singularidades da forma como eles interagem, podemos ver no mundo físico e digital, diversos tipos de redes, cada uma com sua peculiaridade.

a. Redes de transporte[7] como a malha viária do país onde. Os indivíduos podem ser vistos como os aeroportos que se interagem através de vias aéreas com características únicas nessas como a distância, velocidade do ar, etc.

b. Redes de computadores[8], sendo a internet a mais famosa. Os participantes dessa rede podem ser vistos como os computadores dos usuários remetente/destinatário e os switches intermediários que são usados para executar a comunicação. Este tipo de rede também tem suas características singulares, como o link entre 2 agentes dessa rede existir ou não, e caso exista, a largura de banda do *upload* e *download* das mensagens.
c. Redes sociais[9], estas, comumente são associadas às redes onde os indivíduos virtualizam os comportamentos do mundo físico, a exemplo do Facebook, onde os indivíduos se relacionam, estabelecendo amizades que é uma via mútua, trocando mensagens e reagindo às ações dos outros, como curtir a foto de um amigo.



[Figura 1] Representação de um grafo em uma rede social onde os nós são pessoas e os links representam a relação de amizade

O que podemos perceber é que características como os indivíduos participando, a forma como se relacionam e as características desse relacionamento nos ajudam a modelar o tipo de rede no qual estamos trabalhando.

2.1.2 - Representação

Podemos analisar as redes de várias formas, inclusive, a estrutura da análise pode variar de acordo com a ciência no qual estamos inseridos. Neste trabalho, usaremos Grafos como a estrutura computacional para analisar as redes e consequentemente, estruturamos a rede do GitHub como um Grafo.

Neste modelo matemático, podemos enxergar os membros das redes com nós de um grafo e as arestas como a relação entre eles. Pesos nos nós e arestas nos ajudam a modelar a rede.

Exemplos:

a. No Facebook podemos observar as amizades como as arestas (bi-direcionadas) onde os extremos são usuários da rede.
b. Grafos com pesos nas arestas podem nos ajudar a representar a malha metroviária de uma cidade, onde os pesos representam o custo (distância ou gasto de energia) para o metrô se locomover entre dois pontos, sendo esses os nós do grafo.
c. Grafos com os pesos nos nós, poderiam nos ajudar a modelar um sistema de rodovias, onde os nós representam os pedágios que dão acesso às diferentes rodovias.

   Com isto em mente, no capítulo 3 descrevemos o grafo que melhor nos ajudará a modelar a rede em questão.

2.1.3 - Predição de Links

O comportamento dos indivíduos de uma rede geralmente são dinâmicos, e assim podemos tentar prever uma determinada modificação na rede. No Facebook, por exemplo, podemos citar o sistema de recomendação de amizades[1], que tenta nos apresentar uma lista de outros usuários, que provavelmente eu conheço no mundo real ou tenho interesse em conhecer, e consequentemente, adicioná-lo à minha lista de amigos.

Podemos tentar distinguir as predições entre dois tipos:

a. Predição estática:

Neste tipo de predição, tentamos prever links[11] ocultos que nossa amostra de dados não conseguiu nos mostra. Como exemplo podemos citar a relação de causa entre drogas na medicina.

b. Predição temporal[12]

Este modelo será o utilizado em nosso trabalho, pois aqui nós tratamos de links futuros que devem iniciar entre os membros da rede.

As técnicas utilizadas na predição variam de acordo com o que estamos tentando estimar. Podemos nos basear em:

a. Características individuais dos nós como:
    i. Número de nós adjacentes
    ii. Peso do nó
    iii. Características singulares como *hub*
b. Características da topologia
    i. Baseadas em vizinhança como: *Common Neighbors, preferential attachment, triadic closeness[10]*.
    ii. Baseadas em distância como *Katz* e *Hitting time[10]*.

Ainda sobre a técnica *triadic closeness*, que usaremos na predição por distance técnica:
Nesta o objetivo é pré-processar a rede para entender a forma como usuários estão se relacionado, contando o número de tríades na rede.

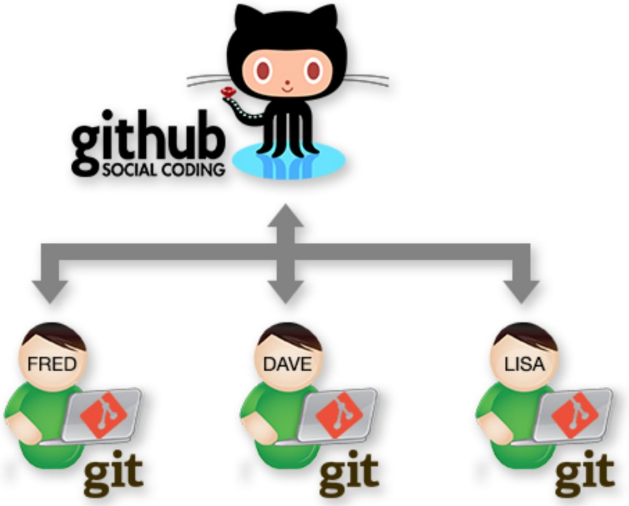
2.1.4 - O GitHub

À medida em que os anos se passam, mais projetos de desenvolvimento são lançados, e soluções de versionamento e compartilhamento de código emergem para resolver problemas como o versionamento de código, contribuição distribuída e outros[13].

Tecnologias como o Git surgem e passam a ingressar o *toolbelt* da Engenharia de software atual. Em cima desta tecnologia, várias outras ferramentas são desenvolvidas para ampliar e facilitar o uso do conjunto de *features* do Git, entre elas, o GitHub.

O GitHub além de facilitar a visualização dos projetos e uso da ferramenta, traz consigo um conjunto de features que agregam à ferramenta características que a transforma

em uma rede de compartilhamento de código utilizada por milhares de usuários ao redor do mundo.



[Figura 2] Representação da relação entre o Git e o GitHub.

A ferramenta hoje pode ser acessível através de uma interface Web e aplicações desktop e mobile. Mas, também fornece uma API para que aplicações terceiras sejam desenvolvidas em cima dos dados fornecidos, nós faremos uso desta API.

Exemplos de *features* do GitHub:

- Visualização de código através de uma interface web
- Seguir um repositório para acompanhar suas modificações
- Seguir outro usuário e acompanhar suas contribuições
- Favoritar um repositório
- Resumo das linguagens de programação utilizadas em determinado projeto
- Participar de uma organização com outros usuários mantendo projetos em comum
- Metadados dos usuários (cidade, email, nome, etc)

Capítulo 3

Sistema de recomendação de usuários no GitHub

3.1 Modelagem e técnicas

3.1.1 Introdução

Dado que agora conhecemos os fundamentos básicos de uma estrutura social representada por grafo e a ferramenta GitHub, podemos apresentar a ferramenta que desenvolvemos neste trabalho. Esta foi desenvolvida com base em análises do comportamento dos usuários na rede, modelagem do grafo, desenvolvimento de um modelo de predição para duas características (similaridade técnica e distância de amizade) e a aplicação e teste da mesma.

O nome escolhido para a aplicação foi *diggit,* que se apresenta-rá acessível através de uma interface *web*, na qual falaremos dos detalhes mais à frente.

3.1.2 Modelagem do sistema

Para ambas as recomendações, seja por similaridade técnica ou por distância de amizade, modelamos um grafo direcionado cujos nós representam os usuários do GitHub e as arestas direcionadas da forma (e,u) representam que o usuário *e* segue o usuário *u*. Essas informações são recuperadas através da API pública do GitHub, cuja documentação está disponível no site oficial da plataforma.

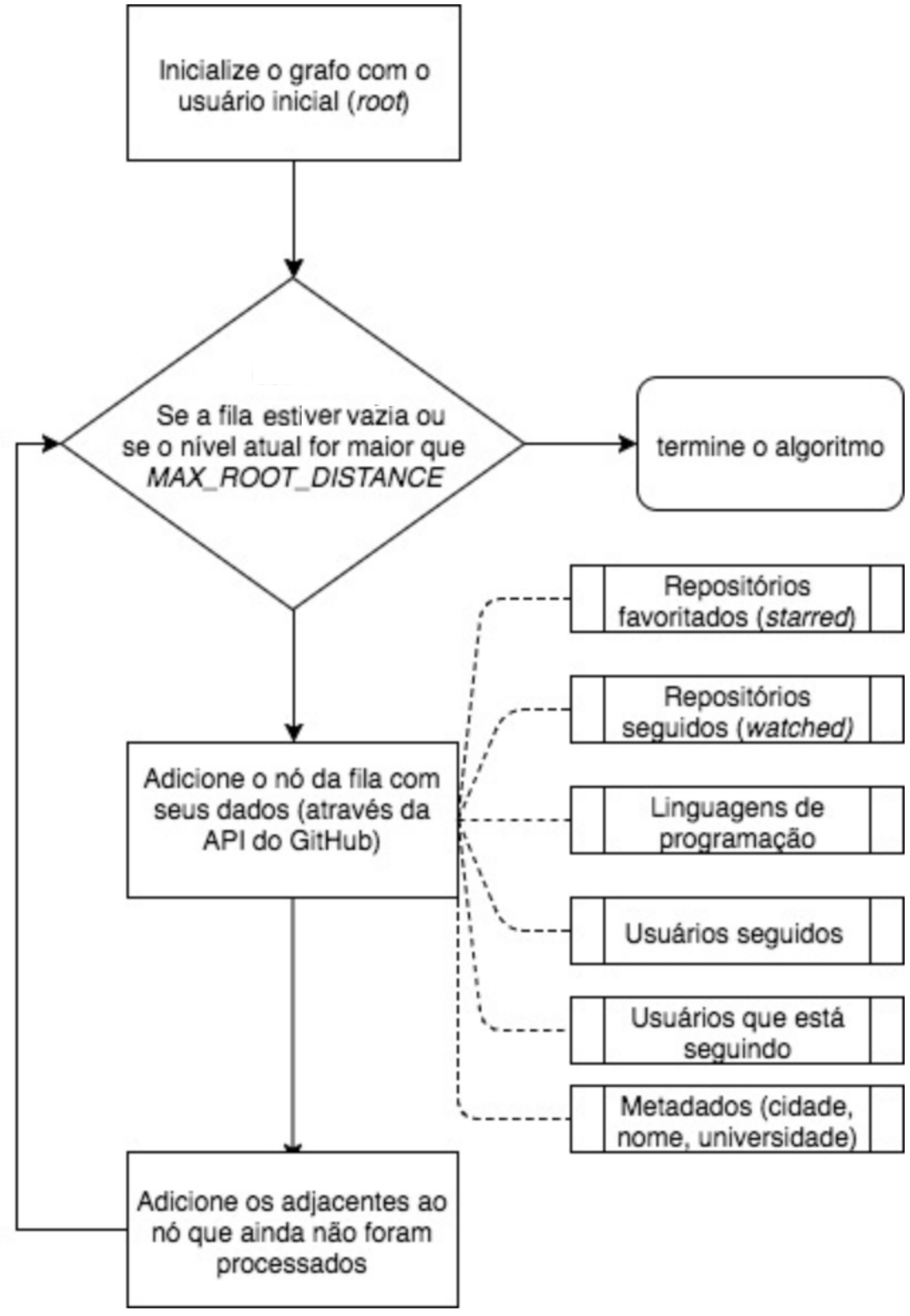
a. Construção do Grafo:

A construção do grafo é feita partindo do usuário de entrada como nó inicial e sua lista de adjacência são os usuários que ele segue.

Todo nó contém informações do usuário resgatadas a partir da API do GitHub (como a lista de adjacência citada acima) e também alguns metadados sobre sua forma no grafo de amizades. Abaixo citamos detalhes de como a construção é realizada e ilustrada na Figura 11.

- A forma como caminhamos será feita utilizando uma Busca em Largura, ignorando nós já processados previamente.
- O nó inicial estaria no level 0 da busca.
- O level máximo é definido através de um parâmetro *MAX_ROOT_DEPTH* setado durante o deploy da aplicação
- As informações do grafo são armazenadas de forma serializada no Redis, que é um banco de dados chave-valor, in-memory e de alta performance, assim as queries são extremamente performáticas em tempo.
- Relações transitivas (*backedges*) são contadas e armazenadas numa variável chamada (*transitive_count*) mas o nó incidente desta relação não é processado de novo para evitar loops infinitos, algo já esperado pela técnica de busca em largura. Armazenamos este número de relações transitivas pois será utilizado posteriormente na recomendação por amizade, pois utiliza uma técnica chamada Triadic Closeness, que aprende o formato padrão da rede num esquema de tríades para realizar as recomendações.
- Durante a construção do grafo, para cada nó não processado, nós recuperamos informações como: Repositórios seguidos, repositórios favoritados, lista de usuários que ele segue (utilizada para montar o grafo), linguagens de programação que contribui, cidade onde mora, organizações, quantidade de seguidores, etc. Tudo isto será armazenado numa estrutura e serializado no Redis. Estas informações serão utilizadas nas técnicas utilizadas para recomendar os usuários.

  Abaixo um fluxograma do nosso algoritmo de construção:



[Figura 11] Fluxograma de construção do grafo

Dado o grafo, neste momento dividiremos entre a recomendação por similaridade técnica e similaridade por distância de amizade

a. Similaridade técnica

   Neste modelo nosso objetivo é apresentar uma lista de usuários próximos do ciclo do social do usuário de entrada e que sejam tecnicamente similares ao usuário de entrada, que chamaremos de *root*.

   Para calcular o *score* de um nó no grafo, utilizamos a técnica similar a *Local Path*[14]. Nesta nós amos definir o *score* como uma soma, com pesos, entre o *score* atribuído aos nós candidatos levando em consideração nós adjacentes que também segue *root* e o *score* atribuído aos nós candidatos levando em consideração apenas os nós adjacentes que não seguem *root*. Estes dois *scores* serão identificados por LP2 e LP3.

   Em LP2, para calcular o *score* de um nó candidato ($curr_2$), nos baseamos nos nós adjacentes à este ($curr_3$), tal que $curr_3$ siga de volta a raiz, este conjunto chamaremos de *commons* e este *score* de LP2.

   E depois calcularemos um *score* para todas as tríades de nós *root,* $curr_2$ e $curr_3$, sendo $curr_3$ um nó adjacente a *curr* que não necessariamente segue *root*, chamemos este *score* de LP3.

   O *score* final para o nó curr será dado da seguinte forma:

$$score(curr_2) = LP2(curr_2) + LP3_{weight} * LP3(curr_2)$$

   $LP3_{weight}$ é um fator que irá decrementar o resultado de LP3 para que este tenha menos importância da recomendação, logo:

$$0 \leq LP3_{weight} \leq 1$$

   Na implementação deste projeto, através de resultados empíricos, percebemos que um bom valor para LP_{weight} seria 0.4. Para decidir este valor, nós, fixamos *MAX_ROOT_DEPTH = 2* e verificamos, a acurácia do projeto, pesquisando através

de uma amostragem de usuários a porcentagem dos usuários do resultado que passaria a ser seguido, este tipo de avaliação é melhor explicada no capítulo 5 (Experimentos).

O *score* também será normalizado numa escala de 0 à 5.

Para calcular os *scores LP2* e *LP3*, fazemos uso de uma função de similaridade que dado 3 três nós genéricos A, B e C, retorna a similaridade deles com base nas seguintes features:

- Número de repositórios seguidos em comum
- Número de repositórios watched em comum
- Número de linguagens em comum

A função de similaridade é definida da seguinte forma:

$$similar(curr_2,\ curr_3) = \frac{4*int_{favoritados}(curr_2,curr_3) + 6*int_{seguidos}(curr_2,\ curr_3) + int_{linguagens}(curr_2,\ curr_3)}{11}$$

sendo as funções $int_{favoritados}$, $int_{seguidos}$ e $int_{linguagens}$ calculam o score da interseção entre os dois nós para repositórios favoritados, repositórios seguidos e linguagens de programação em comum, respectivamente.

$$int_{favoritados}(nodeA,\ nodeB) = \frac{\left| node_A repositoriosFavoritados \bigcap node_B repositoriosFavoritados \right|}{\left| node_A.repositoriosFavoritados \bigcup node_B.repositoriosFavoritados \right|}$$

$$int_{seguidos}(nodeA,\ nodeB) = \frac{\left| node_A repositoriosSeguidos \bigcap node_B.repositoriosSeguidos \right|}{\left| node_A.repositoriosSeguidos \bigcup node_A.repositoriosSeguidos \right|}$$

$$int_{linguagens}(nodeA,\ nodeB) = \frac{\left| node_A.linguagensProgramação \bigcap node_B.linguagensProgramação \right|}{\left| node_A.linguagensProgramação \bigcup node_B.linguagensProgramação \right|}$$

Dado que entendemos a função de similaridade, vamos apresentar como calculamos LP2 e LP3:

Armazenamos o valor dos nós candidatos num dicionário denominado *LP2*, calculando soma das similaridade entre entre $(root, curr_2)$ e $(curr, curr_3)$ sendo $curr_2$ um nó que *curr segue* e que segue *root* como explicado mais acima.

Também armazenamos o número de nós no formato de $curr_2$ armazenando a quantidade através de um dicionário, para fins didáticos assuma que esse número é acessível através da função *commons(x),* onde x é um nó do grafo.

A função $LP2$ pode ser descrita da seguinte forma:

$$LP2(node_{curr}) = \sum_{node_{curr2} \in commons(node_{curr})} similar(node_{curr}, node_{root}) + TT_{weight} * similar(node_{curr2}, node_{root})$$

TT_{weight} é um fator que deve decrementar o *score* de similaridade do par $(node_{curr2}, node_{root})$ para dar mais importância ao par $(node_{curr}, node_{root})$.

$$0 \leq TT_{weight} \leq 1$$

Da mesma forma como estimamos $LP3_{weight}$, neste projeto estimamos $TT_{weight} = 0.4$

A função 3 tem um formato parecido com $LP2$, mas em vez de observarmos os nós no complemento do conjunto *commons(curr)*.

Outra diferença é que o *score* será apenas baseado no par $(curr_3, root)$, e aplicamos como fator à métrica em função da cardinalidade do conjunto $commons(curr_3)$. Utilizaremos uma função logarítmica para minimizar a importância do boost a partir de uma quantidade de usuários.

$$LP3(node_{curr}) = \sum_{node_{curr3} \in commons(node_{curr})'} (1 + log\left|commons(node_{curr3})\right|) * similar(node_{curr3}, node_{root})$$

Com base nessas funções conseguimos calcular o *score* de todos os nós no range do grafo gerado a partir de *root,* normalizados na escala de 0 à 5.

Ordenamos os *scores* e exibimos na tela os usuários com melhor *score*, ou seja, os maiores.

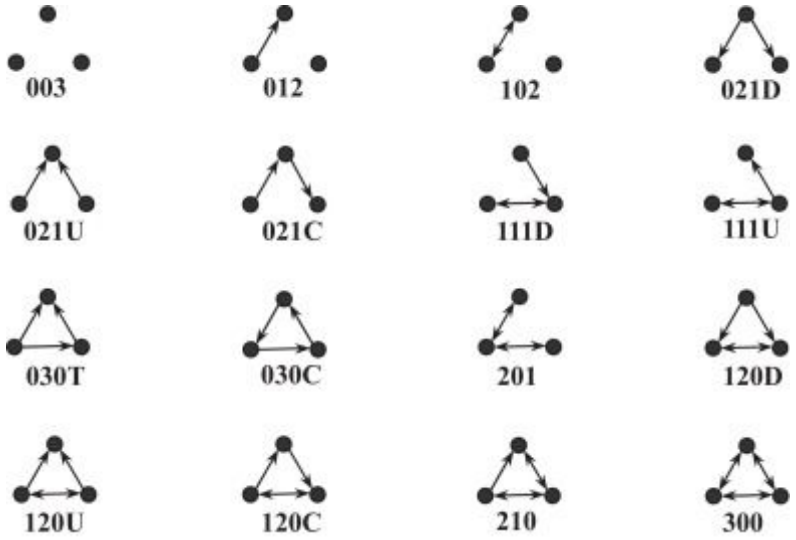
b. Distância de amizade

Nesta modalidade fizemos uso de uma modificação da heurística *Triadic Closeness*[15], técnica que estima o quão próximos estão nós desconectados em termos do número de tríades formadas através de nós intermediários que estão entre eles. Uma grande vantagem é que ela consegue materializar a forma como os indivíduos desta rede estão se comportando.

Durante o cálculo do *score*, também utilizamos a quantidade de seguidores do usuário no grafo, tanto singularmente quanto a relação pareada desse número. Mais a frente explicaremos como este artifício foi usado para otimizar os resultados.

Primeiramente, devemos entender o que são as tríades.

Uma tríade pode ser vista como uma combinação de arestas entre três nós, e o conjunto das tríades seriam todas as combinações possíveis.



[Figura 3] Todas as possíveis tríades para um dado set de 3 nós

Assim, pré-processamos o grafo e calculamos a frequência de cada uma dessas tríades entre 3 nós distintos, pois a função que calcula o *score* de um nó faz uso dessa frequência.

O pseudo-código abaixo define como podemos calcular a frequência das tríades:

para cada $node_a$ *no grafo, faça:*
  para cada $node_b$ *no grafo e diferente do* $node_b$*, faça:*
    para cada $node_c$ *e diferente do* $node_a$ *e* $node_b$*, faça:*
      conte a tríade $node_a \rightarrow node_b \rightarrow node_c$ *se ela existe não foi contada antes*

      conte a tríade $node_a \rightarrow node_b \rightarrow node_c \rightarrow node_a$ *se ela existe e não foi contada antes*

A função de *score* de proximidade (*proxScore*) entre dois nós genéricos do grafo, $node_a$ e $node_b$, se baseará em produto entre o uma função exponencial no número de seguidores que tanto seguem $node_a$ quanto seguem $node_b$, multiplicado por um *score* que se baseia no somatório do produto da relação das frequências das tríades ($node_a, node_b, node_c$), direta e bidirecionada com a tríade simples, com a função de relevância do $node_c$, onde $node_c$ é um nó adjacente à $node_b$.

$$proxScore(node_a,\ node_b) = CM_{weight}^{\,TC(node_a,\ node_b)} * \sum_{node_c \in adj(node_b)} S(node_c) * TS(node_{a,}\ node_{b,}\ node_c)$$

As funções auxiliares:

- $TC(node_{a,}\ node_b) \rightarrow$ número de usuários que seguem $node_a$ e que também seguem $node_b$. CM_{weight} define a base de uma exponencial decrescente em função de $TC(node_{a,}\ node_b)$.

  $$1 \leq CM_{weight} \leq 2$$

  Neste projeto assumimos CM_{weight} = 1.6 , este valor foi definido após testes com uma amostragem de usuários, verificando o quanto o resultado melhorava com esta constante.

- $adj(node_b) \rightarrow$ conjunto de usuários que $node_b$ segue

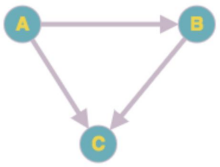
- S($node_c$) $\rightarrow$ *score* de relevância para $node_c$ baseado em uma exponencial decrescente no número de seguidores de $node_c$, ou seja:

- S($node_c$) = $1.001^{-FC(node_c)}$, onde FC($node_c$) representa o número de usuários que seguem $node_c$

- TS($node_a$, $node_b$, $node_c$) $\rightarrow$ Função que calcula o *score* para a tríade ($node_a$, $node_b$, $node_c$) baseada na frequência das tríades.

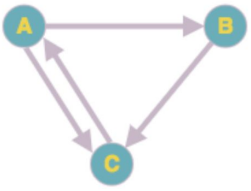
Primeiro vamos descrever as funções que retornam a frequência de cada tipo de tríade, obtido previamente através do pré-processamento, pois estas serão utilizadas na função *TS*:

$$fd(node_a,\ node_b,\ node_c)$$



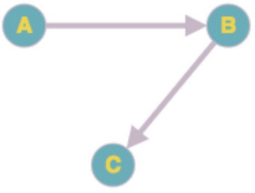
[Figura 4] Tríade direcionada

$$fb(node_a,\ node_b,\ node_c)$$



[Figura 5] Tríade bi-direcionada

$$fp(node_a,\ node_b,\ node_c)$$



[Figura 6] Tríade padrão

$$TS(node_a,\ node_b,\ node_c) = 1, \text{ se } fp(node_a,\ node_b,\ node_c = 0,$$

caso contrário:

$$TS(node_a,\ node_b,\ node_c) = min(10 * \frac{fd(node_a, node_b, node_c) + fb(node_a, node_b, node_c)}{fp(node_a, node_b, node_c)}) + 1$$

Com isso conseguimos estimar o *score* de proximidade (*proxScore*) entre quaisquer dois nós. Logo, para encontrar o conjunto de nós para ser recomendados, basta, para todo nó no grafo, calcular o *proxScore* em relação a *root*. O pseudo-código abaixo ilustra a ideia.

rank = {}

para cada nó (node) no grafo, faça:

ignore o nó caso seja a raiz (root)

rank ← rank + (node, proxScore(root, node))

ordene rank baseado no proxScore dos nós

retorne rank

E então exibimos na tela os usuários mais bem rankeados.

Capítulo 4

Implementação - Web UI

4.1 Escolhas tecnológicas

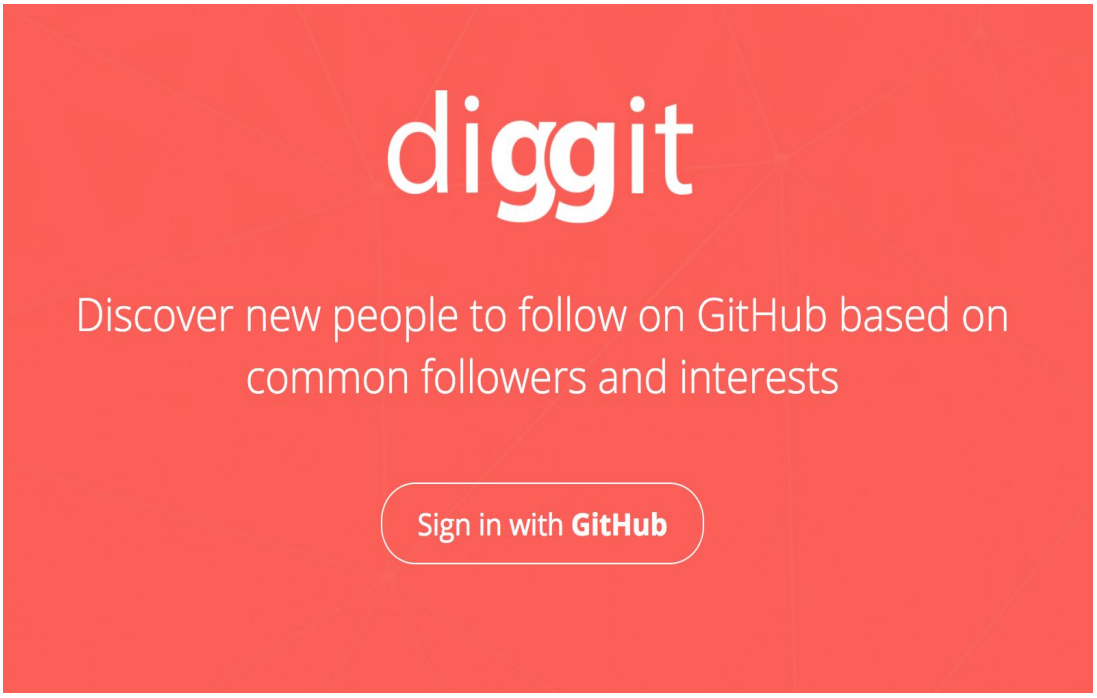
Descritas abaixo:

- Implementamos a ideia acima usando NodeJS (v8.5.0) provendo uma interface web para acessar a aplicação.
- O banco de dados Redis (v2.6.17) foi utilizado para cachear o grafo, evitando sua reconstrução.
- A interface foi desenvolvida com o uso de frameworks web open source como o Bootstrap.
- Libs javascript open source auxiliares foram utilizadas para facilitar o desenvolvimento do servidor, como o cliente de acesso ao redis e o wrapper de acesso à API do GitHub, mas todo o cálculo do *score*, métricas, heurísticas, etc foram feitas *in-house*.

4.2 Interface web

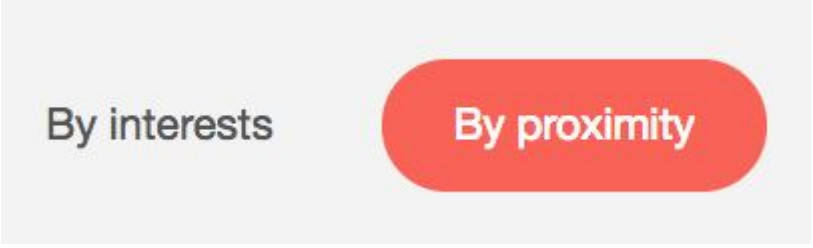
Nesta seção apresentaremos o resultado final da aplicação que implementa toda a ideia descrita nas seções anteriores.

Nesta tela inicial (Figura 12) podemos ver a logo da aplicação, denominada de "*Diggit*" e uma legenda que busca explicar o objetivo da mesma. Logo abaixo da legenda, autenticação (*Sign in with GitHub*), que fará a autenticação com o GitHub. Ao entrar na aplicação pela primeira vez, é pedido que você se autentique com o GitHub via OAuth. Neste tipo de autenticação, o usuário será questionado se permite a aplicação resgatar seus dados do GitHub, caso aceite, ele será redirecionado para a tela com as recomendações (Figura 9). O processamento de construção do grafo é executado durante a transição entre essas duas telas.



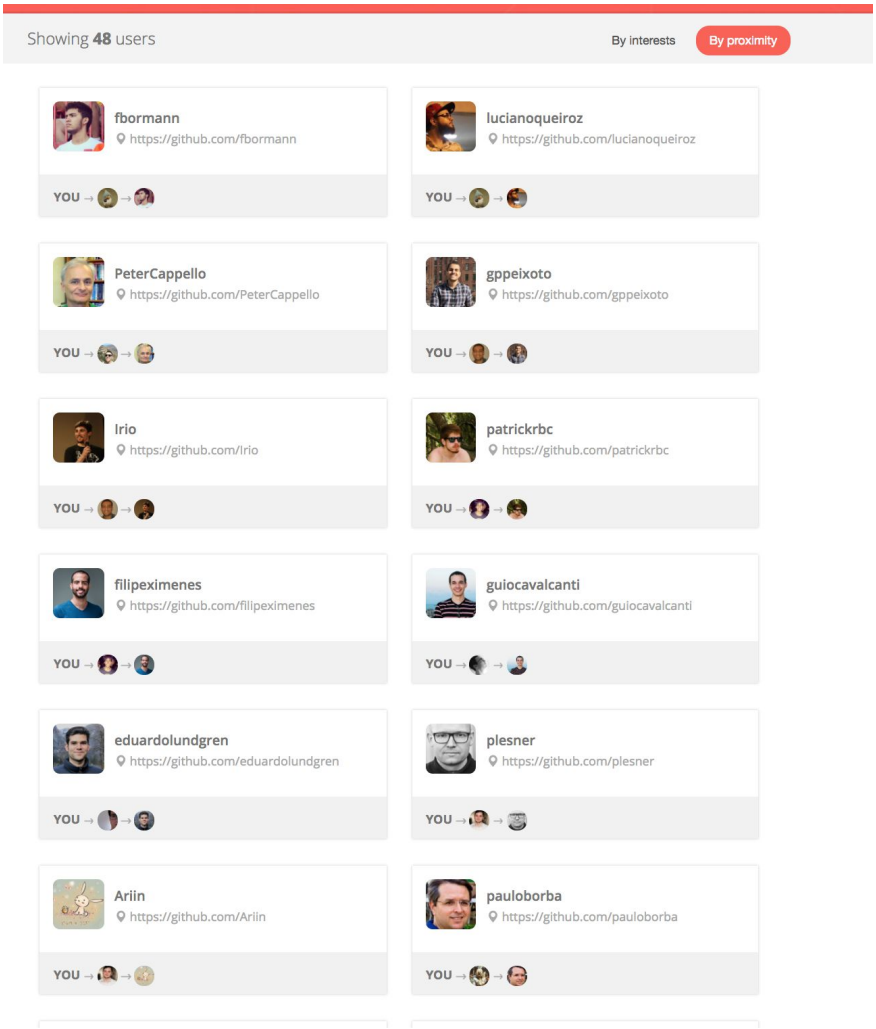
Figura[12] Tela de entrada da aplicação

Dois botões no canto direito da tela (Figura 8) darão a opção de escolher recomendação por similaridade técnica ou distância de amizade. Nesta aplicação, optamos por chamar as técnicas através de pseudônimos a fim de facilitar a UX. A distância por amizade é está representada pela *label* "*By Proximity*" e a recomendação por similaridade técnica é representada pela *label* "*By interests*". Ao clicar em uma das labels, o processamento da lista de usuários será executado e em seguida exibido na próxima tela.



[Figura 8] Botão para escolha da técnica

Dada a opção escolhida, a lista de usuários, ordenados com os que obtiveram maior *score* ao topo, será exibida (Figura 9).



[Figura 9] Tela com a lista de recomendação, os mais bem rankeados ao topo

Nesta tela vemos vários usuários do GitHub recomendados ao usuário inicial através da técnica de distância por amizade. Cada usuário está em uma *div* que informa seu identificador (nick) no GitHub, link para o seu repositório e um caminho entre o usuário inicial e este usuário recomendado.

Capítulo 5

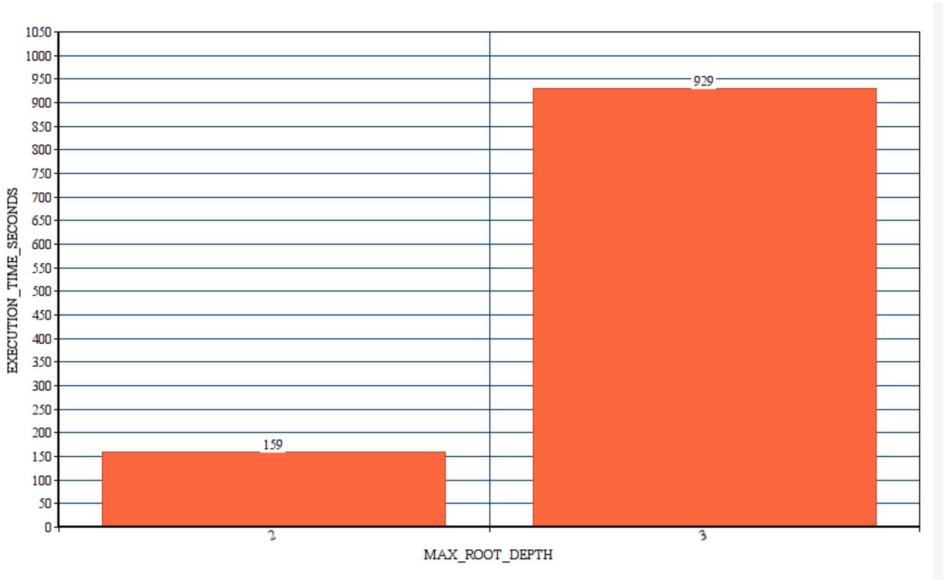
Experimentos

5.1 Experimentos e resultados

Fizemos alguns experimentos relacionados a performance computacional da aplicação e dos resultados.

5.1.1 Performance

Medimos a variação do max depth, q pode melhorar o processamento das tríades e melhorar o resultado mas, em contrapartida, aumenta o tempo necessário de pré processamento.

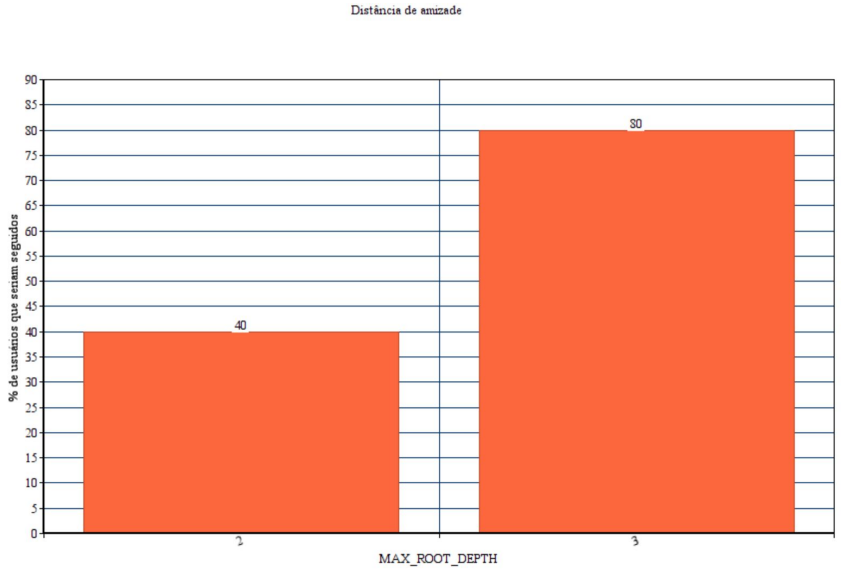


[Figura 10] Comparação do tempo de execução em relação ao nível máximo de construção do grafo (*MAX_ROOT_DEPTH*)

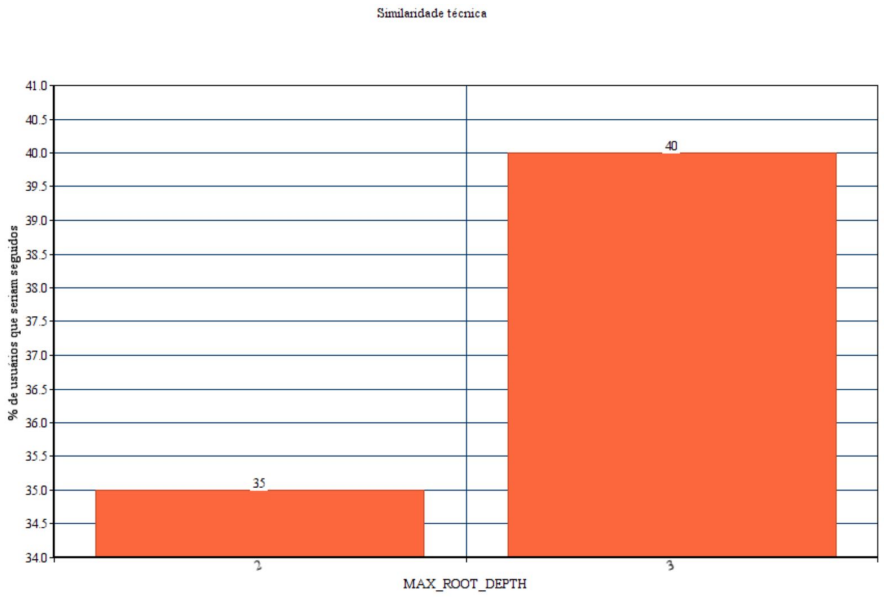
Aumentar o parâmetro MAX_ROOT_DEPTH para 3 aumentou o tempo necessário para construir o grafo em 147%. Apesar desse aumento, precisamos analisar o quanto os resultados melhoraram em relação a essa troca.

5.1.2 Acurácia

Para medir a acurácia em relação a ampliação do parâmetro *MAX_ROOT_DEPTH*, nós exibimos os resultados para um conjunto de uma amostragem de 15 usuários do GitHub, e pedimos para apontarem a quantidade de usuários que eles passariam a seguir dentre a lista apresentada, e obtivemos os seguintes valores para cada uma das métricas:



[Figura 10] Distância de amizade



[Figura 11] Similaridade técnica

Percebemos que a expansão do nível máximo apenas aumentou em 14% para a técnica de similaridade técnica, enquanto que dobrou a acurácia para a distância de

amizade, portanto, este fator pode ser crucial para ampliar a acurácia da recomendação.

Conclusão

Neste trabalho tivemos a oportunidade de entender mais sobre os modelos de rede, com um foco maior nas redes sociais, e em seguida, implementar um sistema de recomendação de amizades para uma das plataformas mais importantes da engenharia de software, o GitHub.

Através da segmentação da recomendação entre pessoas próximas tecnicamente similares e possíveis amigos próximos, conseguimos dar um foco maior em features diferentes e diversificar as técnicas por trás das recomendações. Essa bifurcação se mostrou atrativa e pode ser ampliada para outros tipos de similaridade como especialistas em linguagens, colegas de trabalho, especialistas em tecnologias/frameworks específicos, etc.

A implementação provendo uma interface gráfica facilitou o teste e visualização do sistema em prática, e o uso da cache facilitou o uso da plataforma nas iterações seguintes.

Os sistemas de recomendações e suas técnicas estão a cada dia mais maduros. Com o uso de novas técnicas, o desenvolvimento de novas tecnologias de Big Data e ampliação da base de dados, sem dúvida, estamos dando grandes passos nesta área. Confio que este projeto ajudará na expansão do conhecimento em prol da humanidade.

Referências bibliográficas

[1] Coyle, C. L. and Vaughn, H. (2008), **Social networking: Communication revolution or evolution?**. Bell Labs Tech. J., 13: 13–17. doi:10.1002/bltj.20298

[2] N. B. Silva, I. R. Tsang, G. D. C. Cavalcanti, I. J. Tsang, **A Graph-Based Friend Recommendation System Using Genetic Algorithm**. The 2010 IEEE Congress on Evolutionary Computation (CEC), Barcelona, July 2010.

[3] Yu Wu , Jessica Kropczynski , Patrick C. Shih , John M. Carroll, **Exploring the ecosystem of software developers on GitHub and other platforms**, Proceedings of the companion publication of the 17th ACM conference on Computer supported cooperative work & social computing, February 15-19, 2014, Baltimore, Maryland, USA [doi>10.1145/2556420.2556483]

[4] Laura Dabbish , Colleen Stuart , Jason Tsay , Jim Herbsleb, **Social coding in GitHub: transparency and collaboration in an open software repository**, Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work, February 11-15, 2012, Seattle, Washington, USA [doi>10.1145/2145204.2145396]

[5] Phillips, S. (2007, July 25). A brief history of Facebook. Guardian. Retrieved from www.guardian.co.uk/technology/2007/jul/25/media.newmedia

[6] Wasserman, Stanley; Faust, Katherine (1994). "Social Network Analysis in the Social and Behavioral Sciences". *Social Network Analysis: Methods and Applications*. Cambridge University Press. pp. 1–27. ISBN 9780521387071

[7] M. E. J Newman (2003). "The Structure and Function of Complex Networks", p178.

[8] M. E. J Newman (2003). "The Structure and Function of Complex Networks", p176.

[9] M. E. J Newman (2003). "The Structure and Function of Complex Networks", p174.

[10] Liben-Nowell, D. and Kleinberg, J. (2007), The link-prediction problem for social networks. J. Am. Soc. Inf. Sci., 58: 1019–1031. doi:10.1002/asi.20591

[11] 1. Lise Getoor and Christopher P. Diehl. Link mining: a survey. SIGKDD Explor. Newsl., Vol. 7, No. 2, December 2005.

[12] D. Dunlavy, T. Kolda, and E. Acar. Temporal link prediction using matrix and tensor factorizations. TKDD, 2011.

[13] Blischak JD, Davenport ER, Wilson G (2016) A Quick Introduction to Version Control with Git and GitHub. PLoS Comput Biol 12(1): e1004668. https://doi.org/10.1371/journal.pcbi.1004668

[14] L. Lu, C.-H. Jin, T. Zhou, Similarity index based on local paths for link prediction of complex networks, Physical Review E 80 (2009) 046122.

[15] Schall, D. Soc. Netw. Anal. Min. (2014) 4: 157. https://doi.org/10.1007/s13278-014-0157-9