



*Empoderando vidas.
Fortalecendo nações.*

Projeto BRA/12/018 - Desenvolvimento de Metodologias de Articulação e Gestão de Políticas Públicas para Promoção da Democracia Participativa

Produto 04 - Proposta de adaptações e incrementos para a interface do portal federal de participação social e suas ferramentas

com elementos visuais e de usabilidade para mecanismos de priorização de conteúdos e
autorregulação

Renato Fabbri



Secretaria-Geral da Presidência da República

Produto 04 - Proposta de adaptações e incrementos para a interface do portal federal de participação social e suas ferramentas

Contrato n. 2013/000566

Objeto da contratação: Aporte de conhecimentos e tecnologias para especificação de vocabulário e ferramentas assistidas que utilizam processamento de linguagem natural e análise de redes complexas para o conteúdo do portal da participação social.

Valor do produto: R\$ 21,600 (vinte e um mil e seiscentos reais)

Data de entrega: 10 de Setembro de 2014

Nome do consultor: Renato Fabbri

Nome da supervisora: Gabriella Vieira Oliveira Gonçalves



Secretaria-Geral da Presidência da República

Fabbri, Renato

Proposta de adaptações e incrementos para a interface do portal federal de participação social e suas ferramentas: com elementos visuais e de usabilidade para mecanismos de priorização de conteúdos e autorregulação / 2014.

Total de folhas: 47

Supervisora: Gabriella Vieira Oliveira Gonçalves

Secretaria: SNAS

Secretaria-Geral da Presidência da República

Palavras-chave: reconhecimento de padrões, redes complexas, processamento de linguagem natural, participação social.



Esta obra é licenciada sob uma licença Creative Commons - Atribuição-NãoComercial. 4.0 Internacional.



Empoderando vidas.
Fortalecendo nações.

Sumário

1	Introdução	7
1.1	Contexto e importância da consultoria	7
1.2	Contexto e importância do Produto	7
1.2.1	Objetivos	7
1.2.2	Resultados esperados	8
1.2.3	Caráter inovador	8
1.2.4	Aparato em software e hardware	9
2	Desenvolvimento	9
2.1	Etapas de desenvolvimento anteriores a este produto	11
2.1.1	Sistematização ontológica da participação online	11
2.1.2	Triplificação dos dados do participa.br	11
2.1.3	Levantamento do endpoint SparQL	11
2.1.4	Análises iniciais, modelos	11
2.2	Etapas de desenvolvimento deste produto	11
2.2.1	Reuniões com equipe do participa.br	11
2.2.2	Estudos de aprofundamento e amadurecimento	11
2.2.3	Escrita deste documento	11
2.2.4	Especificação da API de recomendação	12
2.2.5	Implementação do serviço para aquisição dos dados, processamento e entrega em JSON	12
2.2.6	Disponibilização das rotinas no IPython Notebook	12
2.2.7	Proposta de implementações no portal federal de participação social	12
2.3	Justificativa do método	12
2.4	Justificativa das fontes	12
2.5	Confronto entre os resultados esperados e os alcançados	13
3	Usos dos resultados	13
4	Conclusão	14
4.1	Comentários, sugestões, recomendações	14
4.2	Impacto do Produto para a elaboração, gestão e/ou avaliação de políticas públicas de participação social	14
4.3	Como o Produto deverá impactar o público-alvo das políticas públicas a que se refere	15
5	Agradecimentos	15



Empoderando vidas.
Fortalecendo nações.

A	Especificação da API de recomendação de recursos do participa.br	19
A.1	Instância online e especificações	19
A.2	Aspectos operantes	19
A.3	Aspectos planejados	19
A.4	Instância online e exemplos de uso	19
A.5	Atualização e manutenção	22
B	Rotinas de acesso e processamento de dados do participa.br para as recomendações	22
B.1	Estruturas auxiliares	22
B.2	Rotinas para recomendação de recursos	28
C	Parcela dos participantes que produziram texto ou interação	37
D	Infraestrutura do sistema de recomendações	40
E	Instalação e modificação do sistema de recomendações	41
F	Propostas de implementações na interface do portal federal de participação social	41
F.1	Implementação prevista no participa.br	41
F.2	Implementações propostas para o participa.br	42
F.3	Implementações confluentes com o participa.br	45
F.4	Implementações previstas e o próximo produto da consultoria	45



*Empoderando vidas.
Fortalecendo nações.*

Resumo

Este documento descreve rotinas de priorização de conteúdo e de autorregulação para o portal federal de participação social. Como parte da exposição e do Produto, foi implementado um sistema de recomendação de participantes para outros participantes e linha editorial do participa.br. Este sistema está online e visa atender a requisitos como os do ActionItem 3234 do Noosfero, que prevê recomendação de participantes, comunidades e empreendimentos. Como generalização da proposta, é delineada recomendação de recursos como trilhas, artigos, comentários e palavras. O consultor e equipe do participa.br entendem que a disponibilização destes recursos em formato aberto - e com documentações reativas para fácil geração de derivados, críticas e propostas - é instrumental para a democracia participativa e online, promovendo empoderamento e transparência.

Palavras-chave: reconhecimento de padrões, redes complexas, processamento de linguagem natural, participação social.



Empoderando vidas.
Fortalecendo nações.

1 Introdução

1.1 Contexto e importância da consultoria

Em confluência com o portal federal de participação social (Participa.br) e o Plano Nacional de Participação Social (PNPS), esta consultoria é um aporte de conhecimentos e tecnologias de web semântica, redes complexas e processamento de linguagem natural. O presente produto apresenta “formas de priorização de conteúdo e de autorregulação” na forma de um sistema de recomendação de recursos para participantes e linha editorial. No começo da seção 2 está explicitada a motivação para a implementação e a relação com o cumprimento dos termos deste Produto [1].

1.2 Contexto e importância do Produto

1.2.1 Objetivos

Este produto tem por objetivo principal a disponibilização de um sistema de recomendação de recursos do participa.br para usuários, tanto via critérios personalizados quanto considerando comunidades e linha editorial. Através deste sistema de recomendação, ficam facilitados, até mesmo prontamente disponíveis, diversos processos de autorregulação, de geração de resumos, de geração de relatórios e de análises informativas. Objetivos secundários são:

- Disponibilização de uma API HTTP, para uso no participa.br, conforme requisitado pela equipe do Participa.br em diversos itens dos produtos [2].
- A exposição destes algoritmos de recomendação aos visitantes web via uso de browsers comuns, para edição e execução dos trechos de código utilizados pela plataforma federal de participação social. Este objetivo foi satisfeito usando o IPython Notebook levantado para este trabalho [3].
- Aproveitamento do endpoint SparQL com os dados do participa.br, fortalecendo as tecnologias de dados linkados e web 3.0 [4].
- Em reunião com a consultora Daniela Feitosa, foi delineada a pertinência de um sistema de recomendação de perfis para um ActionItem em andamento para o participa.br [5]. Este produto visa suprir esta necessidade através da API disponibilizada para recomendações.
- A entrega das tecnologias livres com simplicidade e boa documentação, favorecendo o aproveitamento deste trabalho para melhoras, geração de derivados e novas e independentes tecnologias. Isso pode ser observado no repositório git público deste produto [6].
- Compatibilizar ao máximo a entrega deste produto às demandas da equipe do participa.br [2].



Empoderando vidas.
Fortalecendo nações.

- Realizar de forma precisa e pertinente a especificação deste quarto produto no Termo de Referência [1].

1.2.2 Resultados esperados

De imediato e mais central, o resultado do produto é iniciar um processo aberto de desenvolvimento e apropriação de análises e mecanismos de autorregulação para o portal federal de participação social.

Como resultados diretos deste Produto, constam:

- a habilitação para uso da API HTTP para recomendação de recursos do participa.br. Veja Apêndice A, D e E.
- A interface para apreensão e inovação dos algoritmos, não somente exemplificada ou projetada, mas operante e disponível. Veja Apêndice B.
- Um plano de implementação para o participa.br, que utiliza a API de recomendação para priorização de conteúdo e autorregulação. Veja Apêndice F.
- Transparência absoluta no trabalho, com toda a documentação e código computacional online em um repositório git que contem o histórico de implementação [6].
- Algoritmos implementados em código de simples leitura, para facilitar a implementação em outras linguagens como Ruby ou Javascript, quando houverem recursos mais maduros para estas linguagens ou por necessidade. No momento, Python possui mais e mais maduros recursos tanto para redes complexas quanto para processamento de linguagem natural.
- Habilitação de implementações em andamento, como o plugin de recomendação de perfis [5], para recomendar amigos para participantes.

1.2.3 Caráter inovador

Centralmente, este trabalho é inovador na aplicação de recursos de análise de redes sociais para empoderamento da sociedade civil, entregando as tecnologias e priorizando a reutilização. Este recurso é de vital interesse para a democracia participativa no contexto atual, com as revoluções da internet e das redes sociais. Permite, em última instância, que haja uma inteligência para aproveitamento das estruturas sociais, e que esta inteligência seja pública, transparente, minimizando vetores vigilantistas ou turvos.

Há a inovação na arquitetura em software, apresentando traços de web 3.0 como os dados linkados como base de conhecimento e os múltiplos recursos online acessados no funcionamento



Empoderando vidas.
Fortalecendo nações.

usual (ao menos Endpoint SparQL para acesso aos dados, API HTTP Flask para tratar os dados e gerar estatísticas e estruturas de interesse, Participa.br para interface e contexto pertinente).

Há inovação na difusão científica. Além dos métodos e tecnologias disponibilizadas, por exemplo, os termos Web Semântica, Processamento de Linguagem Natural e Redes Complexas são empregados em textos científicos e consistem em áreas que recebem pesquisas, revistas e até carreiras científicas inteiras. Neste Produto, este conteúdo está apresentado em português e se presta a facilitar contribuições de outras partes interessadas.

Os métodos de recomendação em si não foram confrontados exaustivamente com a literatura, mas possivelmente possuem também inovações nos procedimentos. Certamente há inovação no contexto de implementação, tanto de relevância social (portal federal de participação social) quanto de aparato tecnológico (web 3.0, métodos da física e de mineração de dados).

1.2.4 Aparato em software e hardware

O sistema de recomendação precisa ser mantido online, e isso implica na manutenção de uma estrutura em hardware e software que extrapola o objeto deste produto. As especificações desta arquitetura de software e hardware estão em um documento escrito pelo consultor a pedido da PR para a SNAS e DITEC [7]. Para fins de pesquisa e de entrega deste produto, o esquema da Figura 1 é realizado em máquinas de pesquisa da USP, concedidas para a pesquisa de doutorado do consultor. Tal configuração é razoável pois a pesquisa conflui e é potencializada por esta consultoria, mas está prevista uma infraestrutura própria da PR para estes serviços. Além disso, os usos atuais ainda são moderados, sem causar sobrecarga ao serviços de computação em nuvem da USP.

2 Desenvolvimento

O produto é descrito no Termo de Referência desta consultoria assim: “Documento com proposta de adaptações e incrementos para a interface do portal e suas ferramentas que inclua elementos visuais e de usabilidade para mecanismos de priorização de conteúdos e auto-regulação com base nos metadados gerados nativamente pela plataforma do portal e nas análises de PLN e RC produzidas pelas ferramentas assistidas”.

Dada a dimensão do participa.br, tanto da estrutura em software e das práticas participativas, quanto de mobilização humana, as propostas de adaptações para o portal são muitas e estão em diversos produtos deste e de outros consultores. Uma forma especialmente pertinente de realizar este produto, confluyente com o Termo de Referência desta consultoria, com o trabalho dos gestores e comunidade e com os produtos deste e de outros consultores [2], é a entrega de um sistema de recomendação de recursos para o participa.br.



Empoderando vidas.
Fortalecendo nações.

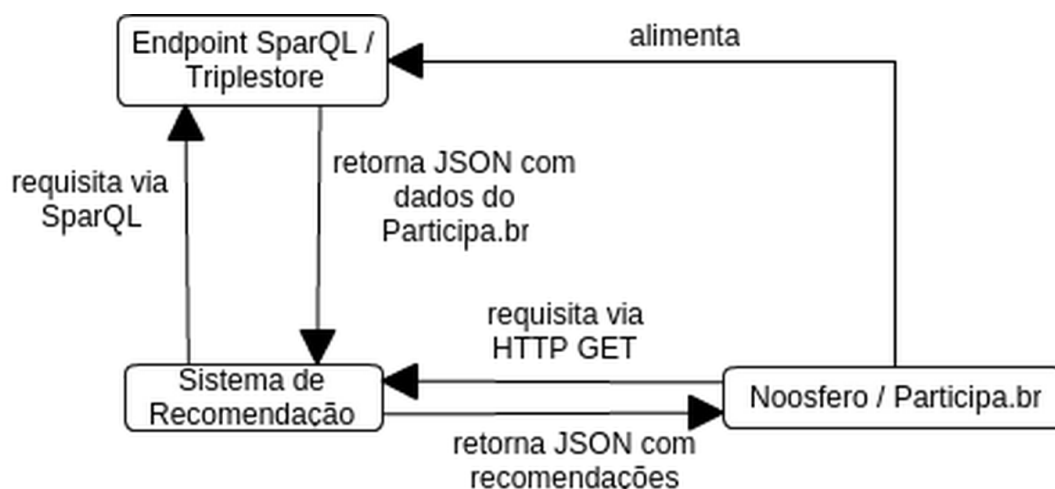


Figura 1: O sistema de recomendação e relações principais: com o frontend Participa.br e com a base de dados enriquecidos semanticamente.

Conforme discutido em reunião com gestores e comunidade do participa.br [8], em termos de *priorização de conteúdo*, quase todos os casos de interesse para o atual participa.br podem ser considerados rotinas de recomendação. Além disso, dadas as práticas atuais de redes sociais, os mecanismos de autorregulação do participa.br são e serão baseados na adição de conteúdo por espontaneidade dos participantes. Esta dinâmica de adição espontânea de conteúdo pelos participantes conflui com sistemas de recomendação de recursos, pois estes podem catalizar sem coerção os processos participativos.

Desta forma, foi idealizada a entrega de um sistema de recomendação de recursos do participa.br para seus usuários, comunidades e linha editorial. As subseções a seguir apontam etapas na construção do sistema de recomendação. Os Apêndices A e B concentram aspectos de operação do sistema. Os Apêndices D e E são voltados para os componentes e a instalação da versão operante deste sistema de recomendação de recursos. A seção 3 e o Apêndice F apreendem previsões e propostas de integração com o participa.br.



Empoderando vidas.
Fortalecendo nações.

2.1 Etapas de desenvolvimento anteriores a este produto

2.1.1 Sistematização ontológica da participação online

Através de estudos e reuniões presenciais e online, a Ontologia de Participação Social (OPS) foi revisada [9] e a Ontologia do Participa.br (OPA) foi feita [10].

2.1.2 Triplificação dos dados do participa.br

Feito um script para triplificar os dados do Participa.br, ou seja, para o enriquecimento semântico e escrita em RDF dos dados em Postgresql da instância Noosfero do Participa.br [11].

2.1.3 Levantamento do endpoint SparQL

Para uso dos dados triplificados, pode-se recorrer a diversos métodos de leitura e disponibilização. Um método-chave é a disponibilização dos dados rdf (*triple store*) em um *endpoint sparql*. Para os fins de testes, pesquisa e usos leves, está disponibilizado um endpoint SparQL em servidores da USP [4].

2.1.4 Análises iniciais, modelos

Análises dos dados do participa.br foram abertas no IPython Notebook, com ênfase no texto produzido e nas redes formadas [12].

2.2 Etapas de desenvolvimento deste produto

2.2.1 Reuniões com equipe do participa.br

Especial agradecimentos ao Ricardo Poppi, Ronald Costa, Enaile Ladanza, Joênio Costa, Daniela Feitosa e Fernando Cruz.

2.2.2 Estudos de aprofundamento e amadurecimento

Em especial, foram lidos todos os produtos entregues pelos consultores, (re)visitados cursos no coursera e literatura científica [2].

2.2.3 Escrita deste documento

Através da escrita deste documento, várias informações foram organizadas e sistematizadas, facilitando compreensão do contexto para especificação da API de recomendação e entrega do produto.



Empoderando vidas.
Fortalecendo nações.

2.2.4 Especificação da API de recomendação

Basicamente, é convencionado um padrão na montagem das URLs para recomendar recursos de tipos especificados, para destinatários e por métodos também especificados. O Apêndice A é dedicado a expor a convenção inicial entre URLs formadas e as recomendações que retornam.

2.2.5 Implementação do serviço para aquisição dos dados, processamento e entrega em JSON

Para a API da etapa acima, foi necessário desenvolver um servidor HTTP simples (em Flask). Veja pasta flask/ de [6].

2.2.6 Disponibilização das rotinas no IPython Notebook

Para expor os critérios de recomendação, disponibilizar análises e facilitar novas versões para as recomendações, as rotinas estão em um IPython Notebook. Veja o Apêndice B para uma exposição e detalhamento destas rotinas.

2.2.7 Proposta de implementações no portal federal de participação social

Com base no sistema de recomendações, nas demandas levantadas em reuniões e nos produtos já feitos por este e outros consultores, foi sistematizado um conjunto de propostas de acréscimos para o portal federal, detalhado no Apêndice F.

2.3 Justificativa do método

São usados diversos métodos neste produto. As justificativas se concentram nas tecnologias de web 3.0 (dados linkados e computação em nuvem), nas técnicas mais fundamentais/básicas/simples e nas demandas da equipe do participa.br, observadas através de reuniões e documentações produzidas recentemente pela equipe [2].

2.4 Justificativa das fontes

A equipe do participa.br é uma equipe estratégica da Presidência da República voltada para participação social. As fontes externas utilizadas, como artigos, livros e videos, são garimpadas como atual pesquisa acadêmica principal do consultor.



Empoderando vidas.
Fortalecendo nações.

2.5 Confronto entre os resultados esperados e os alcançados

No Termo de Referência desta consultoria, este produto é descrito como “Documento com proposta de adaptações e incrementos para a interface do portal”. Este produto entrega estas propostas a serem implementadas no portal federal de participação social. Além disso, para facilitar estas implementações e fazer a conexão com os desenvolvimentos ontológicos e de dados linkados, foi implementado e disponibilizado um sistema de recomendação de recursos para o participa.br. Um outro resultado para além do esperado, é a consideração cuidadosa da comunidade de democracia participativa, com a disponibilização das rotinas de recomendação no IPython Notebook [3] e uma visita aos produtos dos outros consultores [2].

3 Usos dos resultados

Há usos previstos do sistema de recomendação, pois são demandas da equipe que motivaram o seu desenvolvimento. Em especial, há o uso previsto no plugin de recomendação de perfis (ActionItem [5]).

Há usos próprios dos participantes, como consultas aos recursos recomendados para si e para outros, e usos para a linha editorial do portal. Estes usos podem ser feitos no próprio servidor que disponibiliza o serviço de recomendação. Podem ser adaptados para a interface do participa.br, com plugins ou temas apropriados. Algumas sugestões de implementações estão no Apêndice F.

Outro uso previsto, e que idealmente contará com incentivos da comunidade, é a evolução destes métodos de recomendação de perfis para melhor atender aos usos do portal e das comunidades. O consultor sugere que as comunidades sejam convidadas a apresentar uma ou mais pessoas com conhecimentos ou disposição para algoritmos e sistematizações. Esta pessoa poderia passar, por exemplo, por uma reunião para apreensão dos métodos de recomendação e análise, com vistas a proposição de melhoras e ampliação das funcionalidades.

As rotinas de recomendação estarão disponíveis online, e editáveis e executáveis no browser via IPython Notebook, o que deve facilitar dinâmicas de compartilhamento e amadurecimento destes processos. Este uso dos resultados deste produto é ainda mais central do que a API em si ou o uso dela dentro do participa.br. Mesmo assim, dada a complexidade das técnicas usadas, das tecnologias envolvidas e dos propósitos, o uso da API dentro do participa.br trará menos desafios que a apreensão, aproveitamento e melhora destes métodos pela comunidade de democracia participativa. Por isso, fica reforçada a pertinência da atenção da comunidade.

Um uso previsto é a potencialização de uma biblioteca digital, incluindo buscas e recomendações no participa.br. Também sobre biblioteca digital, mas agora na alimentação dela, análises dos dados do participa.br podem ser feitas diariamente e constar como documentos da biblioteca digital. Outras análises especiais podem entrar como itens na biblioteca digital, facilitadas com as rotinas já disponibilizadas, e facilitando posteriores.



Empoderando vidas.
Fortalecendo nações.

4 Conclusão

4.1 Comentários, sugestões, recomendações

O consultor recomenda que haja amadurecimentos em encontros, tanto para melhora deste legado do participa.br, quanto para apreensão dos métodos pelas comunidades interessadas.

As estruturas básicas para as análises e recomendações, que são as redes (de amizade e de interação) e os histogramas de palavras, podem ser convenientemente incluídos na triplificação dos dados, de forma que as recomendações e análises fiquem mais leves. Por hora, para viabilizar o uso, o sistema de recomendações refaz estas estruturas caso seja visitado o caminho **recomenda/atualiza**. Estas estruturas são:

- Rede de amizades.
- Rede de interação.
- Histograma de radicais das palavras de todos os textos (artigos e comentários) do participa.br.
- Seleção dos 400 radicais mais ocorrentes para caracterizar o domínio.
- Histograma de radicais de cada usuário e contagem das ocorrências das palavras selecionadas para caracterizar o domínio.

Os métodos de recomendação implementados utilizam separadamente critérios linguísticos ou de interação/relacionamento. Os casos e resultados já são interessantes e até complexos para uma primeira versão. Há implementações delineadas que utilizam ambos recursos linguísticos e de relacionamento (topológicos). Por hora, ambas as características podem ser utilizadas operando as pontuações das recomendações com base no texto com pontuações das recomendações com base em interação/relacionamento.

4.2 Impacto do Produto para a elaboração, gestão e/ou avaliação de políticas públicas de participação social

Este trabalho torna disponível uma porção de avaliações automáticas dos processos participativos que ocorreram ou ocorrerem no participa.br. Estas mesmas avaliações podem ser usadas continuamente, facilitando a gestão. A elaboração pode se beneficiar da análise das experiências passadas, facilitada pelos métodos aqui presentes.

Outro impacto pra a elaboração e gestão é basear o método de participação na utilização de métodos de recomendação de recursos. Por exemplo: pode-se recomendar que participantes com características X faça provocações. Estas provocações são enviadas para outros participantes



Empoderando vidas.
Fortalecendo nações.

como recomendações. Ainda a outros participantes pode ser recomendada a sistematização destes resultados. Um exemplo mais próximo da democracia participativa atual é uma etapa em que os participantes escrevem para recomendados participantes para fazer contato e iniciar discussões.

Para a gestão, pode ficar facilitado o fomento e a qualificação da participação. Por exemplo, o plugin de recomendação de amigos [5] deixa o portal mais interessante para o participante, pois apreende melhor seus relacionamentos e expõe os critérios de busca como instrumentos para sua participação.

4.3 Como o Produto deverá impactar o público-alvo das políticas públicas a que se refere

As comunidades afeitas à participação social poderão aproveitar estas análises para melhor assimilação dos processos participativos, para relatórios, para gerar novos métodos de recomendação que melhor atendam aos interesses específicos das comunidades ou aos interesses da democracia participativa.

Um impacto imediato é a transparência reforçada dos processos participativos, pois não somente os dados, mas as análises e as recomendações dos recursos do participa.br estão online e publicamente disponíveis para geração de derivados.

Outro impacto imediato é a valorização da participação. Sempre que um participante, por exemplo, fizer uma amizade ou comentar uma postagem ou outro comentário, ele estará constando nas redes envolvidas, modificará as análises atuais, e poderá ver seu nome e de outros em relações diversas.

5 Agradecimentos

O consultor Renato Fabbri agradece ao Joenio Costa pelo template em L^AT_EX para os produtos. Agradece à Daniela Feitosa pela reunião para demanda de recomendação de perfis. Agradece aos supervisores do trabalho realizado em torno do participa.br: Ricardo Poppi e Ronald Costa. Agradece ao labMacambira.sf.net e todas as comunidades de software e cultura livre que compõe esta contribuição.



Empoderando vidas.
Fortalecendo nações.

Referências

- [1] *Termo de Referência da consultoria a que se refere o presente produto.*
- [2] *Repositório Git da leitura dos produtos dos outros consultores.* <https://github.com/ttm/pnudExtra>.
- [3] *IPython Notebook com códigos dos produtos PNUD 3 e 4.* <http://200.144.255.210:8080/>.
- [4] *Endpoint SparQL com dados do Participa.br.* <http://200.144.255.210:8082/>.
- [5] *ActionItem do plugin de recomendação de perfis do participa.br.* <http://noosfero.org/Development/ActionItem3234>.
- [6] *Repositório Git do produto 4: documento e scripts.* <https://github.com/ttm/pnud4>.
- [7] “Especificação dos requisitos em hardware e software para o sistema de monitoramento do participa.br,” <http://tinyurl.com/m27wtfv>.
- [8] *Pad de proposta do produto 4: documento e scripts.* <https://etherpad.mozilla.org/pnud4>.
- [9] “Ontologia de participação social,” <http://tinyurl.com/p2doueu>.
- [10] “Ontologia do participa,” <http://tinyurl.com/lcccwop>.
- [11] “Especificação da triplicação dos dados do participa.br,” <http://tinyurl.com/k74z3yl>.
- [12] *Repositório Git do produto 3: documento e scripts.* <https://github.com/ttm/pnud3>.
- [13] “Elementary observations of textual production in interaction networks,” <http://tinyurl.com/owsmzfv>.
- [14] “Telões para streaming das redes sociais formadas pela atividade no twitter,” <http://ocupagov.meteor.com>.
- [15] “Telão para exibir estrutura formada por uma amostragem de 500 oscs (organizações da sociedade civil) quando relacionadas por órgão concedente e por área de habilitação,” <http://oscs.meteor.com>.
- [16] “Aplicativos dummy para tocar redes sociais,” <http://mm.meteor.com>
<http://mynsa.meteor.com>.
- [17] “Aplicativo dummy que integra recursos de diferentes redes sociais,” <http://mmisssa.meteor.com>.



Empoderando vidas.
Fortalecendo nações.

- [18] “Outro aplicativo dummy que integra recursos de diferentes redes sociais,” <http://mynsa.meteor.com>.
- [19] “Decupagem do bd do noosfero/participa.br,” <http://sourceforge.net/p/labmacambira/fimDoMundo/ci/r>
- [20] *Pad do vocabulário de participação social apontado pelo IPEA, Participa.br, OPA/OPS e Decreto PNPS*. <https://etherpad.mozilla.org/08092014>.



Empoderando vidas.
Fortalecendo nações.

Abreviações e jargão

RC: Redes Complexas

PLN: Processamento de Linguagem Natural

OPS: Ontologia de participação Social

OPA: Ontologia do Participa.br

MMISSA: Monitoramento Massivo e Interativo da Sociedade pela Sociedade para Aproveitamento

AARS: A Análise de Redes Sociais

MyNSA: Monitoring yields Natural Streaming and Analysis

PNPS: Plano Nacional de Participação Social

RDF: Resource Description Framework

HTTP: Hypertext Transfer Protocol

SPARQL: Simple Protocol and RDF Query Language

endpoint SPARQL: ponto de acesso, geralmente HTTP, a dados em RDF via buscas em SPARQL.

Participa.br: Portal federal de participação social.

IPython Notebook: instância online para rodar scripts Python

Meteor: arcabouço para páginas reativas e com funcionamento distribuído.

D3js: biblioteca de visualização de dados.



Empoderando vidas.
Fortalecendo nações.

A Especificação da API de recomendação de recursos do participa.br

As rotinas no Apêndice B podem ser adaptadas para os mais diversos fins. No sistema de recomendação disponibilizado há quatro campos principais e dois auxiliares:

- **Recurso:** o recurso a ser recomendado: participantes, comunidades, trilhas, artigos ou comentários.
- **Destinatário:** para quem está sendo feita a recomendação: participante, comunidade ou linha editorial. Campo auxiliar “idd” para id do destinatário (comunidade ou participante).
- **Método:** método para a recomendação: “topologico”, “textual” ou “hibrido”. Campo auxiliar de polaridade similar, dissimilar ou mista.

A url usada para consulta HTTP incorpora os parâmetros da forma usual: `http://<urlDoServidor>/recomenda?recurso=participante&destinatario=linha_editorial&metodo=topologico&polaridade=mis&ordenacao=embaralhada`

A.1 Instância online e especificações

A.2 Aspectos operantes

No momento de entrega deste produto, estão implementadas recomendações para os casos apontados na Tabela 1

A.3 Aspectos planejados

A Tabela 2 relaciona os recursos a serem implementados para o Participa.br.

A.4 Instância online e exemplos de uso

Está online uma instância do sistema de recomendação no endereço: `http://200.144.255.210:8081/sr`. Para atualizar estruturas auxiliares (pode demorar mais de 30 minutos de processamento), basta visitar o endereço: `http://200.144.255.210:8081/sr/atualiza`. Para realizar consultas, visite `http://200.144.255.210:8081/sr/recomenda?<opç~oesviamétodoGET>`.

Exemplos:

A URL `http://200.144.255.210:8081/sr/recomenda?recurso=participante&destinatario=linha_editorial&metodo=hibrido&polaridade=ambas&ordenacao=compartimentada` retorna JSON



Empoderando vidas.
Fortalecendo nações.

campo	opções	comentários
recurso	participante	recomenda participante que são potenciais amigos ou apropriadas para interação
destinatario	participante linha_editorial	o recurso é recomendado para o participante o recurso é recomendado para a linha editorial do site ou visitante sem registro no participa.br
idd	<id do destinatário>	id do destinatário caso o destinatário seja um participante. Este id é conforme o campo “identifier” da tabela “profiles” do Noosfero/Participa.br
metodo	topologico textual hibrido	usar redes de amizades ou interação para fazer as recomendações usar critérios textuais para fazer as recomendações usar ambos os critérios (topológico e textual) para fazer as recomendações. Por hora retorna ambas as recomendações feitas com base no critérios topológicos e textuais independentemente.
polaridade	simililar dissimilar ambas	retornar recomendações que visam similaridade entre os recursos (potenciais amigos, textos sobre tópicos relacionados, etc) retornar recomendações que visam dissimilaridade entre os recursos (participantes de setores diferentes da rede, vocabulário díspar). retornar ambas as recomendações similares e dissimilares
ordenacao	compartimentada embaralhada intercalada	retorna recomendações agrupadas por critério, na ordem das medições relativas aos critérios, em uma lista de critérios, cada um com um ranqueamento dos recursos e um valor medido. retorna recomendações embaralhadas, em uma lista de tuplas com três campos: [(recurso, medição, critério),(.),] retorna recomendações intercaladas, cada uma com um critério diferente, também uma lista de tuplas com três campos: [(recurso, medição, critério),(.),]

Tabela 1: Opções implementadas no atual sistema de recomendação. Por hora, recomenda para participantes e para linha editorial (ou visitantes), via critérios textuais ou topológicos, outros participantes para se articularem, sejam potenciais amigos, parceiros ou oponentes, dependendo dos critérios adotados para a recomendação.



Empoderando vidas.
Fortalecendo nações.

campo	opções	comentários
recurso	<p>comunidade</p> <p>artigo</p> <p>comentario</p> <p>palavra</p> <p>trilha</p> <p>etapa</p>	<p>recomenda comunidades de especial interesse para o destinatário</p> <p>recomenda artigo de especial interesse para o destinatário</p> <p>recomenda comentario de especial interesse para o destinatário</p> <p>recomenda palavra de especial interesse para o destinatário</p> <p>recomenda trilha participativa de especial interesse para o destinatário</p> <p>recomenda etapa participativa de especial interesse para o destinatário</p>
destinatario	<i>mesmas opções que o campo recursos</i>	—

Tabela 2: Opções a serem implementadas no atual sistema de recomendação. O destinatário, de início os participantes e a linha editorial, idealmente é qualquer recurso, a que se destina relacionar as recomendações. Veja Tabela 1 para critérios já implementados no momento de entrega deste produto.



Empoderando vidas.
Fortalecendo nações.

com diversos grupos de participantes recomendados para a linha editorial, por critérios topológicos e textuais.

A URL `http://200.144.255.210:8081/sr/recomenda?recurso=participante&destinatario=participante&idd=marcelobranco&metodo=hibrido&polaridade=ambas&ordenacao=compartimentada` retorna JSON com diversos grupos de participantes recomendados para o participante cujo identificar na tabela `profiles` é `marcelobranco`.

Embora o sistema deva funcionar normalmente e com os critérios aplicados corretamente, verificações e melhoras devem vir com o uso. Aliás, o consultor está à disposição caso desenvolvedores do Noosfero/Participa.br queiram requisitar dados ou recomendações em formatos que lhes sejam convenientes.

A.5 Atualização e manutenção

A URL `http://200.144.255.210:8081/sr/atualiza` atualiza as estruturas auxiliares de acordo com os dados disponíveis no endpoint SparQL. O processamento pode levar mais de 30 minutos, de forma que não se deve visitar o link a menos que haja razão para isso. Estas estruturas devem idealmente serem disponibilizadas no endpoint SparQL, junto aos dados do `participa.br`. A estrutura pronta pode ser adicionada à *triple store* através do endpoint Jena/Fuseki posteriormente à triplificação dos dados do `participa.br`. Caso não signifique sobrecarga para a instância do Noosfero/Participa.br, a criação destas estruturas devem ser incluídas no RDF pelo plugin que triplifica os dados do portal, feito por Daniela Feitosa.

B Rotinas de acesso e processamento de dados do participa.br para as recomendações

B.1 Estruturas auxiliares


```
In [9]: q="""SELECT ?participante1 ?participante2 ?aname ?bname
        WHERE {
            ?comentario dc:type tsioc:Comment.
            ?participante1 ops:performsParticipation ?comentario.
            ?participante1 foaf:name ?aname.
            ?artigo sioc:has_reply ?comentario.
            ?participante2 ops:performsParticipation ?artigo.
            ?participante2 foaf:name ?bname.
        }"""
sparql=SPARQLWrapper(URL_ENDPOINT_)
sparql.setQuery(PREFIX+q)
sparql.setReturnFormat(JSON)
results = sparql.query().convert()
d=x.DiGraph()
for interacao in results["results"]["bindings"]:
    nome_chegada=interacao["participante1"]["value"]
    nome_partida=interacao["participante2"]["value"]
    if (nome_partida,nome_chegada) in d.edges():
        d[nome_partida][nome_chegada]["weight"]+=1
    else:
        d.add_edge(nome_partida,nome_chegada,weight=1.)
__builtin__.d=d
```

Bag-Of-Words com todos os comentarios e artigos do participa.br

```
In [39]: q="SELECT ?comentario ?titulo ?texto WHERE \
        {?comentario dc:type tsioc:Comment.\
        OPTIONAL {?comentario dc:title ?titulo . }\
        OPTIONAL {?comentario schema:text ?texto .}]"
sparql=SPARQLWrapper(URL_ENDPOINT_)
sparql.setQuery(PREFIX+q)
sparql.setReturnFormat(JSON)
results = sparql.query().convert()
msgs_=results["results"]["bindings"]
msgs=[mm for mm in msgs_ if ("titulo" not in mm.keys()) or
      (("teste de stress" not in mm["titulo"]["value"].lower())
       and ("comunidade de desenvolvedores e nesse caso, quanto mais"
            not in mm["texto"]["value"].lower()))]
palavras=string.join([i["texto"]["value"].lower() for i in msgs])
palavras = ''.join(ch for ch in palavras if ch not in EXCLUDE)
#palavras = ''.join(ch for ch in palavras if ch not in EXCLUDE).encode('utf-8')
palavras_=palavras.split()
palavras__=[stemmer.stem(pp) for pp in palavras_ if pp not in STOPWORDS]
fdist_=k.FreqDist(palavras__)
# escolhendo as 400 palavras mais incidentes para referência
palavras_escolhidas=fdist_.keys()[:400]
__builtin__.palavras_escolhidas=palavras_escolhidas
__builtin__.fdist_=fdist_
```

```
In [100]: T=time.time()
q="SELECT ?cbody ?titulo ?abody WHERE \
    {?foo ops:performsParticipation ?participacao.\
    OPTIONAL { ?participacao schema:articleBody ?abody. }\
    OPTIONAL {?participacao dc:title ?titulo . }\
```



```

OPTIONAL {?participacao schema:text ?cbody .}}"
sparql=SPARQLWrapper(URL_ENDPOINT_)
sparql.setQuery(PREFIX+q)
sparql.setReturnFormat(JSON)
results = sparql.query().convert()
msgs_=results["results"]["bindings"]
msgs=[mm for mm in msgs_ if ("titulo" not in mm.keys()) or
    (("teste de stress" not in mm["titulo"]["value"].lower())
    or ("cbody" not in mm.keys() or ("comunidade de desenvolvedores e ne
        not in mm["cbody"]["value"].lower())))]
textos1=[i["cbody"]["value"] for i in msgs if "cbody" in i.keys()]
textos2=[i["abody"]["value"] for i in msgs if "abody" in i.keys()]
textos=textos1+textos2
# faz BoW e guarda num dict
texto=string.join(textos).lower()
texto_ = ''.join(ch for ch in texto if ch not in EXCLUDE)

texto__=texto_.split()
#texto__=[stemmer.stem(pp) for pp in texto_]
texto__=[stemmer.stem(pp) for pp in texto_ if (pp not in STOPWORDS) and
bow=k.FreqDist(texto__)
radicais_escolhidos=bow.keys()[:400]
__builtin__.radicais_escolhidos=radicais_escolhidos
__builtin__.bow=bow
print("demorou %.2f segundos para fazer a BoW"%(time.time()-T,))

```

demorou 128.67 segundos para fazer a BoW

```

In [70]: print(u"são %i textos postados e %i comentários,\n\
    totalizando %i palavras informativas em %i radicais"%
    (len(textos1), len(textos2),len(texto__), fdist.B()))

```

são 16253 textos postados e 1554 comentários,
totalizando 2477698 palavras informativas em 18233 radicais

BoW de cada participante

```

In [105]: T=time.time()
q="""SELECT DISTINCT ?participante
    WHERE {
        ?foo dc:contributor ?participante .
    }"""
sparql=SPARQLWrapper(URL_ENDPOINT_)
sparql.setQuery(PREFIX+q)
sparql.setReturnFormat(JSON)
results = sparql.query().convert()
participantes_=results["results"]["bindings"]
participantes=[i["participante"]["value"] for i in participantes_]
# inicia loop
if "palavras_escolhidas" not in dir(__builtin__):
    print(u"rode BoW antes, para saber do vocabulário geral do portal")
else:
    palavras_escolhidas=__builtin__.palavras_escolhidas
bows={}
for participante in participantes:
    # puxa todos os comentarios de cada usuario
    # e os article bodys

```

```

q="""SELECT DISTINCT ?abody ?cbody
WHERE {
  <%s> ops:performsParticipation ?participacao.
  OPTIONAL { ?participacao schema:articleBody ?abody. }
  OPTIONAL { ?participacao schema:text ?cbody. }
  OPTIONAL {?comentario dc:title ?titulo . }
}""%(participante,)
sparql = SPARQLWrapper("http://localhost:82/participabr/query")
sparql.setQuery(PREFIX+q)
sparql.setReturnFormat(JSON)
results = sparql.query().convert()
results_=results["results"]["bindings"]
results__=[mm for mm in results_ if ("titulo" not in mm.keys()) or
(("teste de stress" not in mm["titulo"]["value"].lower()) or
("cbody" not in mm.keys() or
("comunidade de desenvolvedores e nesse caso, quanto mais"
not in mm["cbody"]["value"].lower())))]

textos1=[i["cbody"]["value"] for i in results__ if "cbody" in i.keys()]
textos2=[i["abody"]["value"] for i in results__ if "abody" in i.keys()]
textos=textos1+textos2
# faz BoW e guarda num dict
texto=string.join(textos).lower()
texto_ = ''.join(ch for ch in texto if ch not in EXCLUDE)
texto__=texto_.split()
texto___=[stemmer.stem(pp) for pp in texto__ if (pp not in STOPWORDS)]
fdist=k.FreqDist(texto___)
ocorrencias=[fdist[i] for i in radicais_escolhidos]
bows[participante]=(fdist,ocorrencias)
__builtin__.bows=bows
len(texto___)
print("%.2f segundos para fazer BoW de cada participante"%(time.time()-T,))

```

895.30 segundos para fazer BoW de cada participante

```

In [103]: pickle.dump( g, open( "pickledir/g.p", "wb" ) )
pickle.dump( d, open( "pickledir/d.p", "wb" ) )
pickle.dump( bow, open( "pickledir/bow.p", "wb" ) )
pickle.dump( radicais_escolhidos, open( "pickledir/radicais_escolhidos.p",
pickle.dump( bows, open( "pickledir/bows.p", "wb" ) )

```




*Empoderando vidas.
Fortalecendo nações.*

B.2 Rotinas para recomendação de recursos

Rotinas para recomendação de recursos

Estas rotinas são acionadas pelo sistema de recomendação de recursos e dependem das estruturas auxiliares.

```
In [72]: from SPARQLWrapper import SPARQLWrapper, JSON
import networkx as nx, nltk as n, numpy as np
import string, time, random, __builtin__
```

```
In [13]: URL_ENDPOINT="http://localhost:82/"
URL_ENDPOINT+=URL_ENDPOINT+"participabr/query"
EXCLUDE=set(string.punctuation+u'\u201c'+u'\u2018'+u'\u201d'+u'\u2022'+u'\u201e')
STOPWORDS=set(k.corpus.stopwords.words('portuguese'))
PREFIX="""PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ops: <http://purl.org/socialparticipation/ops#>
PREFIX opa: <http://purl.org/socialparticipation/opa#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/terms/>
PREFIX tsioc: <http://rdfs.org/sioc/types#>
PREFIX sioc: <http://rdfs.org/sioc/ns#>
PREFIX schema: <http://schema.org/>"""
stemmer = k.stem.RSLPStemmer()
```

carregando as estruturas auxiliares:

```
In [45]: g=pickle.load( open( "pickledir/g.p", "rb" ) )
         d=pickle.load( open( "pickledir/d.p", "rb" ) )
         bow=pickle.load( open( "pickledir/bow.p", "rb" ) )
         radicaais_escolhidos=pickle.load( open( "pickledir/radicaais_escolhidos.p",
         bows=pickle.load( open( "pickledir/bows.p", "rb" ) )
```

realizando estruturas derivadas:

```
In [19]: d_=x.Graph()
          efoo=d.edges(data=True)
          for e in efoo:
              if d_.has_edge(e[0],e[1]):
                  d_[e[0]][e[1]]["weight"]+=e[2]["weight"]
              else:
                  d_.add_edge(e[0],e[1],weight=e[2]["weight"])
          eg=g.edges()
          ed=d.edges(data=True)
          ed_=d_.edges(data=True)
```

função auxiliar:

```
In [21]: def NL(narray):  
          return narray/narray.sum()
```

Recomendação de participantes:

recomendaParticipante(destinatario, idd=0,
metodo="topologico",polaridade="ambas",ordenacao="compartimentada"):

```
In [89]: #destinatario="linha_editorial"
destinatario="participante"
idd="rfabbri"
metodo="hibrido" # topologico+textual
polaridade="ambas" # similar e dissimilar
ordenacao="compartimentada"
recomendacoes=[]
```

Recomendação de participantes para linha editorial:

```
In [90]: if destinatario=="linha_editorial":
        if metodo in ("topologico","hibrido"):
            ###
            # topologico
            # puxa a rede em si, retorna geral
            wd=d.degree(weight="weight")
            wd_=[(i,wd[i]) for i in wd.keys()]
            wd_.sort(key=lambda x: -x[1])
            recomendados=[i[0] for i in wd_]
            pontuacao=[i[1] for i in wd_]
            criterio="numero de interacoes (forca do participante no grafo de
            recomendacoes.append({"recomendados":recomendados,
                                   "pontuacao":pontuacao,
                                   "criterio":criterio})

            ud=d.degree()
            ud_=[(i,ud[i]) for i in ud.keys()]
            ud_.sort(key=lambda x: -x[1])
            recomendados=[i[0] for i in ud_]
            pontuacao=[i[1] for i in ud_]
            criterio="quantidade de participantes com que interagiu (grau do p
            recomendacoes.append({"recomendados":recomendados,
                                   "pontuacao":pontuacao,
                                   "criterio":criterio})

            gd=g.degree()
            gd_=[(i,gd[i]) for i in gd.keys()]
            gd_.sort(key=lambda x: -x[1])
            recomendados=[i[0] for i in gd_]
            pontuacao=[i[1] for i in gd_]
            criterio="quantidade de participantes amigos"
            recomendacoes.append({"recomendados":recomendados,
                                   "pontuacao":pontuacao,
                                   "criterio":criterio})

        if metodo in ("textual","hibrido"):
            ###
            # textual
            # usa Bow para comparar os usuarios com a media geral,
            # retorna dos mais típicos e os outliers
            ocorrencias=[bow[i] for i in radicais_escolhidos]
            bow=n.array(ocorrencias, dtype=float)
            bowNL=NL(bow)
            rec=[]
```

```

for uri_ in bows.keys():
    bow_=n.array(bows[uri_][1],dtype=n.float)
    if bow_.sum():
        distancia=n.sum(bowNL-NL(bow_))**2
        rec.append((uri_,distancia))
rec.sort(key = lambda x: x[1])
recomendados=[i[0] for i in rec]
pontuacao=[1/(i[1]+1) for i in rec]
criterio="semelhanca com o vocabulario do participa.br como um todo"
recomendacoes.append({"recomendados":recomendados,
                        "pontuacao":pontuacao,
                        "criterio":criterio})

```

Recomendação de participante para participante

```

In [91]: if destinatario=="participante":
        uri="http://participa.br/profile/"+idd
        if metodo in ("topologico","hibrido"):
            ###
            # todos os participantes x_n com que interagiu,
            # na ordem decrescente de interação:
            # {d_i}_0^n, I[x_n]>=I[x_(n-1)],
            # com I a intensidade interacao, o número de mensagens trocadas
            if uri in d_.nodes():
                x_n=d_[uri]
                x_n=[(i,x_n[i]["weight"]) for i in x_n.keys()]
                x_n_.sort(key=lambda x: -x[1])

                # é feita sugestão dos participantes que não são amigos:
                # x_n!=g_n, g_n um amigo
                if uri in g_.nodes():
                    viz=g_.neighbors(uri)
                    x_n_=[i for i in x_n_ if i[0] not in viz]
                    recomendados=[i[0] for i in x_n_]
                    pontuacao=[i[1] for i in x_n_]
                    criterio="numero de interacoes"
                    recomendacoes.append({"recomendados":recomendados,
                                            "pontuacao":pontuacao,
                                            "criterio":criterio})

            ###
            # avançado e talvez desnecessário: recomenda usuários
            # com quem os amigos mais interagiram
            # e q jah n sao amigos do participante que recebe a recomendação
            # pode ficar pesado quando o usuário tiver muitos amigos

            ###
            # achar amigo de amigo, excluir amigos e recomendar
            if uri in g_.nodes():
                vizs=g_.neighbors(uri)
                vizs_=set(vizs)
                vv=[]
                for viz in vizs:
                    vv+=g_.neighbors(viz)
                vv_=list(set(vv))
                candidatos=[(i,vv.count(i)) for i in vv_ if i not in vizs_]

```

```

candidatos.sort(key=lambda x: -x[1])

recomendados=[i[0] for i in candidatos]
pontuacao=[i[1] for i in candidatos]
criterio="mais amigos em comum"
recomendacoes.append({"recomendados":recomendados,
                        "pontuacao":pontuacao,
                        "criterio":criterio})

###
if polaridade in ("dissimilar","ambas"):
    recomendacoesD=[] # para recomendacoes com polaridade dissimilar
    ### maiores geodesicas partindo do destinatario.
    if uri in g.nodes():
        caminhos=x.shortest_paths.single_source_shortest_path(g,uri)
        caminhos_=[caminhos[i] for i in caminhos.keys()]
        caminhos_.sort(key=lambda x: -len(x))
        #distantes=[(i[-1],len(i)) for i in caminhos_]
        recomendados=[i[-1] for i in caminhos_ if len(i)>2]
        pontuacao= [len(i) for i in caminhos_ if len(i)>2]
        criterio="participantes na mesma rede de amizades, mas mais
        recomendacoesD.append({"recomendados": recomendados,
                                "pontuacao":pontuacao,
                                "criterio":criterio})

    # feito para amigos, agora com a rede de interacao
    if uri in d.nodes():
        caminhos=x.shortest_paths.single_source_shortest_path(d,uri)
        caminhos_=[caminhos[i] for i in caminhos.keys()]
        caminhos_.sort(key=lambda x: -len(x))
        recomendados=[i[-1] for i in caminhos_ if len(i)>2]
        pontuacao= [len(i) for i in caminhos_ if len(i)>2]
        criterio="participantes na mesma rede de amizades, mas mais
        recomendacoesD.append({"recomendados":recomendados,
                                "pontuacao":pontuacao,
                                "criterio":criterio})

    # participantes de outras componentes conexas com relacao ao d
    if uri in g.nodes():
        comps=x.connected_components(g)
        # caso haja duas componentes conexas
        if len(comps)>1:
            recomendados=[]
            # caso sejam exatamente duas componentes:
            if len(comps)==2:
                for comp in comps:
                    if uri not in comp:
                        # recomenda a componente toda
                        recomendados+=[(i,1) for i in comp]
                        criterio="participantes da unica componente
            # caso sejam mais de duas componentes:
            else:
                for comp in comps:
                    if uri not in comp:
                        # escolhe participante da componente
                        recomendados.append((random.sample(comp,1)
                        criterio="participante de componente de am
            recomendados_=[i[0] for i in recomendados]
            pontuacao=[i[1] for i in recomendados]
            recomendacoesD.append({"recomendados": recomendados_,
                                    "pontuacao":pontuacao,
                                    "criterio":criterio})

if metodo in ("textual","hibrido"):
    # acha amigos

```



```

if uri in g.nodes():
    amigos=g.neighbors(uri)
else:
    amigos=[]
# verifica se bow eh vazia
# listar pelos que tem vocabulário mais semelhante
# segundo critério de menor distancia euclidiana
bowi=bows[uri]
if bowi[0].N() == 0:
    # bow do destinatario vazia, usando media geral:
    ocorrencias=[bow[i] for i in radicais_escolhidos]
    bowi=n.array(ocorrencias,dtype=float)
else:
    bowi=n.array(bowi[1],dtype=float)
uris=bows.keys()
rec=[]
for uri_ in uris:
    if (uri_ != uri) and (uri_ not in amigos):
        bowj=n.array(bows[uri_][1],dtype=n.float)
        distancia=n.sum(NL(bowi)-NL(bowj))**2
        rec.append((uri_,distancia))
rec.sort(key = lambda x: x[1])
if len(rec)>0:
    recomendados=[i[0] for i in rec]
    pontuacao=[1/(i[1]+1) for i in rec]
    criterio="semelhanca dentre vocabularios E (0,1]. Calculo: sem
    recomendacoes.append({"recomendados":recomendados,
                           "pontuacao":pontuacao,
                           "criterio":criterio})

if metodo=="hibrido":
    # fazer medida composta de vocabulario e proximidade na rede de in
    # fazer medida composta de vocabulario e proximidade na rede de am
    # pega amigo de amigo, rankeia por media de amigos em comum e voca
## polaridade negativa:
    # pega amigo de amigo, rankeia por inverso da media de amigos em c
pass

```

In [92]: len(recomendacoes)

Out[92]: 3

Ajustando polaridade da recomendação

Se de similaridade, dissimilaridade ou ambas

```

In [93]: if polaridade in ("dissimilar","ambas"):
        try:
            recomendacoesD
        except:
            recomendacoesD=[]

```

```
# inversao das ordenacoes anteriores
for i in xrange(len(recomendacoes)):
    recomendacoesD.append({})
    recomendacoesD[i]["recomendados"]=recomendacoes[i]["recomendados"]
    recomendacoesD[i]["pontuacao"]=recomendacoes[i]["pontuacao"][::-1]
    recomendacoesD[i]["criterio"]="INVERTIDO: "+recomendacoes[i]["criterio"]
if polaridade == "ambas":
    recomendacoes=recomendacoes+recomendacoesD
else:
    recomendacoes=recomendacoesD
```

Formato de entrega das recomendações

Se embaralhada ou intercalada

```
In [94]: # o embaralhamento e intercalação são cortezias da api
if ordenacao=="embaralhada":
    recs=[]
    for i in xrange(len(recomendacoes)):
        tanto=int(len(recomendacoes[i]["recomendados"])*0.1)
        if tanto < 2:
            tanto=min(5,len(recomendacoes[i]["recomendados"]))
        for j in xrange(tanto):
            recomendado=recomendacoes[i]["recomendados"][j]
            pontuacao=recomendacoes[i]["pontuacao"][j]
            criterio=recomendacoes[i]["criterio"]
            recs.append((recomendado, pontuacao, criterio))
    random.shuffle(recs)
    recomendacoes=recs
if ordenacao=="intercalada":
    recs=[]
    cond=1
    cont=0
    while cond:
        for i in xrange(len(recomendacoes)):
            if cont<len(recomendacoes[i]["recomendados"]):
                recomendado=recomendacoes[i]["recomendados"][cont]
                pontuacao=recomendacoes[i]["pontuacao"][cont]
                criterio=recomendacoes[i]["criterio"]
                recs.append((recomendado, pontuacao, criterio))
            cont+=1
            if cont>=7:
                cond=0
    recomendacoes=recs
```

```
In [95]: len(recomendacoes)
```

```
Out[95]: 9
```

Recomendação de demais recursos

```
In [101]: def recomendaComunidade(destinatario, idd, metodo="hibrido",polaridade="ambas"):
```

```

# recomenda por vocabulario em comum do usado na comunidade com o part.
#### puxar dados:
# texto produzido pela comunidade
# participantes na comunidade

# por possuir membros amigos ou que interagiram muito
# por possuir mais amigos de amigos

# semelhanca de vocabulario

# mais amigos de pessoas com quem interagiu
# mais pessoas que interagiram com seus amigos
# por media de amigos e vocabulario utilizado
pass
def recomendaTrilha(destinatario, idd, metodo="hibrido", polaridade="mista")
# que prazo final nao tenha passado
# e prazo inicial esteja proximo
# que possui amigos que colaboraram
# que possui amigos e pessoas que interagiram com o destinatario
# cujos textos sao proximos aos do participante
pass
def recomendaArtigo(destinatario, idd, metodo="hibrido", polaridade="mista")
if destinatario=="participante":
    uri="http://participa.br/profiles/"+idd
    if metodo in ("topologico","hibrido"):
        # que seja de amigo ou de pessoa com quem interagiu
        if uri in d.nodes():
            x_n=d_[uri]
            x_n=[(i,x_n[i]["weight"]) for i in x_n.keys()]
            x_n.sort(key=lambda x: -x[1])
            # busca artigo destes participantes, recomenda os artigos
            count=0
            q=""
            for participante in x_n[:5]:
                q+="""SELECT ?artigo%i, abody%i
                WHERE {
                    <%s> ops:performsParticipation ?artigo%i.
                    ?artigo%i schema:articleBody ?abody%i.
                }
                """%(count,count,participante[0],count,count,count)
                count+=1
            sparql=SPARQLWrapper(URL_ENDPOINT_)
            sparql.setQuery(PREFIX+q)
            sparql.setReturnFormat(JSON)
            results = sparql.query().convert()
            # adiciona recomendação
        if uri in g.nodes():
            vizinhos=g.degree(g.neighbors(uri))
            # adiciona recomendação
    if metodo in ("textual","hibrido"):
        # que tenha vocabulario parecido ou proximo
    pass

def recomendaComentario(destinatario, idd, metodo="hibrido", polaridade="mista")
# que seja de amigo ou de pessoa com quem interagiu
# que tenha vocabulario parecido ou proximo
# que tenha maior media de ambas
pass
def recomendaPalavra(destinatario, odd, metodo="hibrido", polaridade="mista")
pass

```



In []:



*Empoderando vidas.
Fortalecendo nações.*

C Parcela dos participantes que produziram texto ou interação

```
In [12]: from SPARQLWrapper import SPARQLWrapper, JSON
import cPickle as pickle
```

```
In [3]: URL_ENDPOINT="http://localhost:82/"
URL_ENDPOINT_=URL_ENDPOINT+"participabr/query"
PREFIX="""PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ops: <http://purl.org/socialparticipation/ops#>
PREFIX opa: <http://purl.org/socialparticipation/opa#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/terms/>
PREFIX tsioc: <http://rdfs.org/sioc/types#>
PREFIX sioc: <http://rdfs.org/sioc/ns#>
PREFIX schema: <http://schema.org/>"""
```

```
In [19]: q="""SELECT ?participant
        WHERE {
            <http://participa.br/> dc:contributor ?participant.
        }"""
sparql=SPARQLWrapper(URL_ENDPOINT_)
sparql.setQuery(PREFIX+q)
sparql.setReturnFormat(JSON)
results = sparql.query().convert()
```

```
In [20]: results_=results["results"]["bindings"]
participantes=[i["participant"]["value"] for i in results_]
```

```
In [21]: len(participantes)
```

```
Out[21]: 3825
```

```
In [22]: g=pickle.load( open( "pickledir/g.p", "rb" ) )
d=pickle.load( open( "pickledir/d.p", "rb" ) )
bows=pickle.load( open( "pickledir/bows.p", "rb" ) )
```

```
In [23]: len(bows)
```

```
Out[23]: 3825
```

```
In [24]: com_rastro=[part for part in participantes if
                    (part in g.nodes()) or (part in d.nodes()) or (sum(bows[part][1])>0)]
```

```
In [25]: len(com_rastro)
```

```
Out[25]: 872
```

```
In [32]: print("%.2f por cento dos participantes são passíveis de serem recomendados e de rec
          ((len(com_rastro)/float(len(participantes)))*100,))
```

22.80 por cento dos participantes são passíveis de serem recomendados e de receberem recomendações personalizadas

```
In [ ]:
```




Empoderando vidas.
Fortalecendo nações.

D Infraestrutura do sistema de recomendações

A estrutura geral em software e hardware está delineada na subseção 1.2.4 e é tema de todo este documento. Este apêndice descreve os scripts do sistema de recomendação.

O sistema usufrui do endpoint SparQL para adquirir os dados. O usuário/cliente do sistema de recomendação acessa URLs via HTTP e recebe JSON, da forma usual. As rotinas de recomendação em si estão no Apêndice B. Dada a maturidade das bibliotecas em Python, a facilidade de desenvolvimento e a capacidade da comunidade em aproveitar scripts simples na linguagem, todo o sistema de recomendação está em Python. O servidor HTTP feito em Flask, o acesso ao endpoint SparQL (jena) pelo SPARQLWrapper, o processamento de linguagem natural em NLTK, os aproveitamentos das redes complexas em NetworkX. Bibliotecas padrão da linguagem (*builtin* na *Python Standard Library*), e que estão em uso no sistema de recomendação, são: json, cPickle, string, random, `__builtin__`.

O sistema está na pasta flask/ do repositório público [6]. As bibliotecas podem todas serem instaladas via pip e, para iniciar o sistema, basta chamar `$ python recomenda.py` ou apontar o apache para a pasta via WSGI. O sistema aproveita os binários inclusos ou levará até 30 minutos para iniciar com dados mais recentes, pois é necessária a criação das estruturas auxiliares descritas na seção 4.1. Este tempo de processamento é um dos principais motivadores para incluir estas estruturas nos dados triplicados, disponíveis no endpoint SparQL.

Pode-se considerar os arquivos do sistema de recomendação, disponíveis na pasta flask/, assim:

- **configuracao.py**: variáveis de configuração utilizadas em mais de um arquivo, como a URL do endpoint SparQL, as stopwords consideradas, os caracteres desconsiderados e os prefixos (namespaces) usados para as consultas SparQL. Atualmente o script possui menos de 20 linhas de código.
- **auxiliar.py**: rotinas de criação das estruturas auxiliares. Por hora são quatro: 1) redes de amizade e 2) de interação. 3) Histograma de palavras usadas em todo participa.br e 4) por cada usuário. Atualmente o script possui ≈ 150 linhas de código.
- **rotinasRecomendacao.py**: as rotinas de recomendação de recursos propriamente ditas. Utilizam as estruturas auxiliares e opções dadas pelo usuário/cliente. Retorna recomendações. Atualmente, o script possui mais de 350 linhas de código.
- **recomenda.py**: este arquivo levanta o servidor Flask e repassa as opções escritas na URL (método GET) para as rotinas em **rotinasRecomendacao.py**. Atualmente, o script possui menos de 100 linhas de código.



Empoderando vidas.
Fortalecendo nações.

E Instalação e modificação do sistema de recomendações

O sistema de recomendação pode ser instalado em poucos passos. Por exemplo, em um sistema Ubuntu, os seguintes passos são suficientes:

1. Clone do repositório: `$ git clone http://github.com/ttm/pnud4`.
2. Na pasta flask/, inicia serviço com `$ python recomenda.py` (ou aponta apache para a pasta com wsgi).
3. Caso necessário, arrumar configurações de acesso ao endpoint SparQL no arquivo `configuração.py`.

Para fins de uso do sistema de recomendações dentro do Participa.br, é apropriado o uso de infraestrutura dedicada e que assegure a disponibilidade do serviço. Idealmente, este sistema deve estar junto and endpoint SparQL com os dados do participa.br. Para usos em frontends próprios, este mesmo sistema pode ser instalado em servidores de computação em nuvem gratuitos, como o heroku. Este uso de tecnologias leves e bastante difundidas visa facilitar a geração de materiais derivados. O IPython Notebook também se presta a este fim, com as rotinas expostas para execução e modificação pelo visitante que as acessam por browsers usuais (firefox, chromeum) [3].

F Propostas de implementações na interface do portal federal de participação social

Há implementações previstas e implementações propostas para funcionamento interno ao portal federal de participação social (participa.br). Há também implementações em andamento e confluente com o participa.br, mas para funcionamento externo à instância Noosfero/Participa.br, com autonomia das bases de dados e frontends. Parte destas implementações estão sendo estudadas para o próximo produto desta consultoria.

F.1 Implementação prevista no participa.br

O ActionItem [5], prevê uma página em que são recomendados participantes e comunidades para o usuário logado. Os métodos de recomendação apresentados neste Produto são mais formais e aprofundados (e numerosos) do que os requisitados pelo ActionItem. Por outro lado, não estão triplicadas as relações de pertencimento a comunidade, assim como não estão as tags associadas aos usuários e comunidades. Assim, fica prevista a implementação do plugin do participa.pr que disponibiliza uma página com usuários recomendados. Pode haver adaptação do plugin para melhor atender à profundidade e abrangência do sistema de recomendação feito, como para compatibilizar com limitações como a atual ausência de recomendação de comunidades.



Empoderando vidas.
Fortalecendo nações.

Há também a previsão de ampliar o conjunto dos dados triplicados, incluindo tags e o pertencimento às comunidades e permitindo ao atual sistema recomendar comunidades e considerar as tags.

F.2 Implementações propostas para o participa.br

Como recomendação geral, o consultor propõe que o participa.br tenha seus processos participativos catalizados com recomendações. Os instrumentos para esta catálise devem ser apresentados ao participante para que ele aproveite o legado para propósitos próprios, como o encontro de parceiros ou grupo que se opõe, ou participantes com propostas e bagagens diferentes.

No sistema de recomendação entregue junto a este Produto, os critérios das recomendações são sempre enviados junto às recomendações em si, na ordem e com pontuação. É solicitado ao exibir o critério de recomendação e opção para mais recomendações sob aquele e outros critérios. O IPython Notebook foi levantado para convidar visitantes à proposição de novas formas de recomendação.

Estes métodos de recomendação são *em geral* ranqueamentos. Estes ranqueamentos podem orientar análises e escolha de participantes para papéis especiais. Em reuniões da equipe do participa.br foram por vezes sugeridas que trilhas participativas tenham etapas com base nestas atribuições fruto de tecnologias e métodos livres, abertos e formais. Por exemplo: conforme [13], os periféricos produzem mais substantivos e podem ficar responsáveis por uma primeira etapa de levantamento de tópicos de interesse, como substantivos e termos. Em um segundo momento, os hubs, que produzem mais adjetivos (também segundo [13]), podem qualificar estes assuntos de interesse. A construção de textos a partir destes assuntos e qualificações podem ser feitas pelos intermediários, por suas características também diferenciadas, por exemplo por não serem externos na rede como os periféricos mas e tampouco viscerais como os hubs. Estes textos, talvez selecionados, podem ser colocados em consulta pública de comentário por parágrafo, como já utilizado no participa.br.

Para o uso destes critérios para etapas e trilhas participativas, o grupo precisa já possuir atividade/rastro no participa.br (ou instâncias com dados também triplicados), de preferência no próprio grupo. Caso contrário não há como aplicar os critérios pois não há estrutura para ser analisada. Dentre os que já possuem rastro digital (preferencialmente no participa.br, mas podem ser também páginas do facebook ou registros de tweets), o consultor recomenda que estas trilhas sejam concebidas com a comunidade, junto a algum especialista em redes ou participação social. O consultor responsável por este produto se disponibiliza para auxiliar essa interface com a comunidade na criação da trilha e suas etapas, por entender a pertinência do processo, embora esteja além da consultoria em si.

Para grupos sem rastros digitais prévios, pode-se usar estratégias, como a expectativa de que $\approx 5\%$ vai realmente se engajar, pois é a parcela típica de hubs. De forma semelhante, pode-se observar outros grupos do participa e aproveitar abertamente e em comunidade as propriedades.



Empoderando vidas.
Fortalecendo nações.

O consultor recomenda a criação de dashboards para diferentes recursos do participa.br: usuários, comunidades, trilhas, comentários, palavras, etc. Estes dashboards podem ser externos ao participa.br, disponibilizados juntos do sistema de recomendação (em flask) ou em meteor. Caso seja pertinente, estas dashboards podem ter ID liberada pelo sistema do Noosfero para iframes, de forma que interfaces para estatísticas dos recursos estejam prontamente acessíveis, por exemplo, via lightboxes (balões) nas páginas do participa.br.

O consultor considera importante que seja dada alguma visibilidade aos dados que estão sendo doados e aos usos que estão sendo feito, isso para o visitante e para o participante. Talvez através destes dashboards, dos diferentes recursos do Participa.br, seja alcançado um aproveitamento mais generalizado destes bens que são os dados e tecnologias participativos.

Ao visitar a instancia online do participa.br para conceber possibilidades de implementação/a-proveitamento de outras recomendações e análises, as possibilidades destacadas pelo consultor são:

- Ao clicar na barra verde do rodapé, com número de usuários, tags, comentários e acessos, pode ser aberto um dashboard com uma análise geral do participa, como as recomendações da linha editorial deste Produto, ou análises como as contidas no produto anterior desta consultoria [12].
- Junto aos três pilares Participe!, Proponha! e Mobilize!, pode estar Analise!
- Na comunidade de ajuda, item Participe!, como estímulo, podem estar apontados dashboards do participa.br como um todo ou da própria comunidade.
- Na comunidade de ajuda, item Mobilize!, podem ser disponibilizadas receitas para difusão de informação, como o movimento em direção aos centrais, o uso de centralidades excludentes ou de técnicas de controle de Baràbasi. Estas receitas apontam participantes para serem acionados pelo participante interessado na mobilização, com base nas redes do participa.br ou outras redes doadas ou apontadas por participantes.
- Todo artigo, comentário, palavra, além de acesso a um dashboard que analisa aquela instância do recurso, deve poder também ser usado para recomendação de recursos do próprio participa.br. Caso seja pertinente, o consultor poderá auxiliar a implementação destas recomendações a partir de recursos, e os dashboards de cada recurso.
- Na página inicial, logo abaixo do campo reservado para trilhas, pode haver um campo para recomendação de recursos, com as opções do sistema de recomendação de recursos, com os critérios expostos e recomendação de aproveitamento.



Empoderando vidas.
Fortalecendo nações.

- Na página inicial, nos campos de comunidades e de pessoas, acrescentar, ao lightbox de opções, dados sobre a pessoa ou ao menos um link para o dashboard. Enriquecer as lightboxes dos recursos da página com dados dos ranqueamentos e dashboard.
- Na página inicial, observar recomendações atuais dos recursos e aproveitar este novo sistema de recomendações para auxiliar linha editorial (que escolhe o conteúdo de capa do participa.br).
- Nas páginas com postagem de blog, disponibilizar uma análise das palavras e radicais mais incidentes e quais estão sendo usadas para classificação (que estão dentre as 400 mais frequentes que são escolhidas). Junto a esta análise, recursos recomendados a partir da postagem e os critérios usados.

Duas implementações sugeridas, menos focadas no portal e mais na funcionalidade, são:

- Habilitar plugin para etiquetação desenvolvido pela Daniela Feitosa para o Noosfero/Participa.br. Este plugin permitirá a etiquetação de recursos por participantes do participa.br. Essa etiquetação é essencial para a classificação automática e enriquecimento das análises já existentes para o participa.br. A etiquetação por vários participantes pode melhorar em muito a qualidade dos dados etiquetados, representando amostras maiores e mais diversas.
- Fomentar mecanismo de doação de dados (tweets, redes do facebook, dados inseridos por formulário, outras redes, etc). Junto a este mecanismo de doação, abrir acesso público aos dados doados. Interfaces como scrapperWiki e IPython Notebook podem facilitar derivação das análises destes dados doados. No fim das contas, esta demanda aponta para um sistema de monitoramento aberto.

Os produtos dos outros consultores também apontam direções valiosas para desenvolvimento. Em especial: confluir plugin para personalização de bloco de estatísticas (da página inicial) com as análises propostas neste produto e no produto anterior do consultor.

Há três vetores nos desenvolvimentos propostos:

- Dashboards para cada recurso, com as análises entregues via JSON para o participa mediante requisições HTTP.
- Uso de qualquer dos recursos para recomendação de outros recursos.
- Exposição de roteiros de aproveitamento da rede social do participante e do participa.br para processos participativos, geralmente envolvendo difusão e coleta de informação. Por exemplo: um participante pode acionar pessoas periféricas da sua rede para iniciar um processo amplo de difusão de informação.



Empoderando vidas.
Fortalecendo nações.

Para documentação automática da atividade no participa.br, fortalecendo a transparência e a apropriação da democracia participativa, é útil a criação de análises periódicas diárias, semanais, mensais, semestrais, etc. Estas análises podem ser incorporadas na biblioteca digital de participação social.

Assim que a biblioteca digital estiver acessível, o sistema de recomendação para recursos deve contemplar recursos da biblioteca digital, tanto para recomendar recursos do participa.br quanto para serem recomendados a partir de recursos do participa.br.

F.3 Implementações confluentes com o participa.br

Potencializada pela web de dados (web 3.0, web semântica, dados linkados), é pertinente o desenvolvimento rápido e distribuído de utilidades e análises. Neste contexto, além do arcabouço já exposto (endpoint SparQL, servidor Flask, Noosfero/participa.br), cabe a concepção de frontends com desenvolvimento facilitado, de baixo custo e potentes para visualização e construção coletiva de estruturas via streaming. No âmbito desta consultoria, foram feitos diversos aplicativos voltados para streaming de estruturas sociais e visualização. As possibilidades convergiram empiricamente para o uso do Meteor como framework para desenvolvimento em Node.js e D3.js para visualização. Para observar outras redes sociais (e.g. Twitter, Facebook), ambos Python e Node.js mostraram soluções operantes, porém em Python está mais sólida a comunicação com as APIs, além da pronta disponibilidade de ferramentas de ponta para as análises de redes e de linguagem. Como o D3.js é leve e baseado no padrão SVG, e o Meteor é reativo às mudanças no banco de dados, a solução é apropriada para usos coletivos e simultâneos, assim como para streaming de estruturas sociais [14, 15, 16, 17, 18].

Outra implementação recomendada pelo consultor é a triplificação dos dados de outras instâncias (como do portal Cidade Democrática ou do sistema Autorregulação Algorítmica). Deve-se pensar cuidadosamente em como integrar este legado na biblioteca de participação social em concepção pelo SNAS/SGPR, tanto na confluência das catalogações e da produção de conteúdo (análises), quanto na consideração da biblioteca digital para as análises e para a recomendação de recursos.

F.4 Implementações previstas e o próximo produto da consultoria

Há diversas formas de realizar o quinto produto desta consultoria. Portanto, abaixo estão apontadas possibilidades decantadas, não metas estabelecidas, passíveis de serem realizadas por diversas partes interessadas na democracia participativa:

- Ampliar triplificação dos dados do participa.br com ao menos: 1) relações de ser membro de comunidade, 2) tags relacionadas às comunidades e aos usuários, 3) dados suplementares do campo 11 da tabela profiles (composto) [19].



Empoderando vidas.
Fortalecendo nações.

- Levantamento de um vocabulário de participação social. Considerados os vocabulário enviados pelo IPEA, o Decreto 8243/14 (PNPS), Ontologias de Participação Social e do Participa.br (OPS e OPA), e o vocabulário incidente no Participa.br. Disponibilização em SKOS e/ou para D-Space. Epad em [20].
- Integração deste arcabouço semântico com a biblioteca digital de participação social sendo erigida pela SNAS/SGPR.
- Ampliar recomendações para os outros recursos. Implementar métodos apontados pela equipe ou literatura ou elaborados para este caso do participa.br.
- Aprofundamento do tratamento linguístico, e.g. com lematização e classes morfossintáticas.
- Scripts que, para análise dos dados, use dados de instâncias diferentes através do endpoint sparql que serve os dados do participa.br. Scripts que acessem diferentes endpoints para as análises (eg. dbpedia). Instâncias de integração dos dados do participa.br com outras instâncias (e.g. frontends).
- Triplificação dos dados do Cidade Democrática para disponibilização junto aos do participa.br via endpoint SparQL.
- Triplificação dos dados da Autorregulação Algorítmica para disponibilização junto ao participa.br via endpoint SparQL.
- Estabelecimento de equivalências entre usuários e informação de alias.
- Encaixe dos perfis do participa.br nos padrões vCard e FOAF.
- Organização da OPA e da OPS com relação à triplificação dos dados.
- Triplificação dos dados do AllOurIdeas usado pelo Participa.br, o que inclui as consultas públicas.
- Estudo do pwiki e aproveitamento para o sistema de recomendação ou dashboards.
- Conjunto de perguntas simples respondidas com consultas ao endpoint SparQL.
- Uma página estática com instruções sobre como usar essa inteligência tecnológica disponibilizada pelo participa.br.
- Implementação total ou parcial das opções para recomendação relacionadas na Tabela 2, seção A.3.



*Empoderando vidas.
Fortalecendo nações.*

- Implementação dos sistema de dashboards, recomendações e receitas de uso, descrito na seção F.2.