|  |
| --- |
| brasao |
| Demetrius Costa Rapello  Sistema de recomendação matérias de segundo nível |
| Dissertação de mestrado  Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico e Científico da PUC-Rio.  Orientador: Prof. Marco Antonio Casanova |
|  |
| Rio de Janeiro, 25 de janeiro de 2012. |

|  |
| --- |
| brasao |
| Demetrius Costa Rapello  Sistema de recomendação matérias de segundo nível |
|  |
| Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico e Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada. |
| Prof. Marco Antonio Casanova  Orientador e Presidente  Departamento de informática - PUC-Rio  Prof. 1  Departamento de informática - PUC-Rio  Prof. 2  Departamento de informática - PUC-Rio  Prof. 3  Departamento de informática - PUC-Rio  Rio de Janeiro, 25 de janeiro de 2012. |

|  |  |
| --- | --- |
|  | Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador. |
|  | Demetrius Costa Rapello  Graduou-se em Ciência da Computação pela Universidade de Augusto Motta (UNISUAM) em Dezembro de 2000. Tem experiência na área de Ciência da Computação, com ênfase em Desenvolvimento de Software. Tem trabalhado em analise de sistemas desde 1997. |
|  | Ficha Catalográfica |
|  | Demetrius Costa Rapello  Sistema de recomendação matérias de segundo nível/ Demetrius Costa Rapello; orientador: Marco Antonio Casanova. - Rio de Janeiro: PUC-Rio, Departamento de Informática, 2012.  v., 68 f.: il. ; 29,7 cm  Dissertação de Mestrado - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática  Referências bibliográficas incluídas.  Sistemas de Recomendação; Recuperação da Informação; Extração de Entidades Nomeadas; |

A Deus, aos meus pais, a minha esposa e aos meus filhos.

|  |
| --- |
| Agradecimentos |
| A Deus pelo apoio incondicional, conforto de coração e paz de espírito que tanto foram importantes para alcançar este objetivo.  Aos meus pais, Tarciso Rapello e Maria da Conceição Costa Rapello, pelo apoio, educação, carinho e dedicação.  A minha esposa Gabriela Barbosa da Silva Rapello, pelo apoio constante, carinho e compreensão.  Aos meus filhos Gabriel e Matheus que entenderam a ausência do pai com carinho e compreensão.  Ao meu orientador, prof. Marco Antonio Casanova, por sua dedicação, motivação, ensinamentos e orientação.  À Globo.com, pelo financiamento e auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.  Aos professores da Comissão examinadora.  A todos os amigos e familiares que de alguma forma contribuíram para a realização deste trabalho. |

|  |
| --- |
| Resumo |
| Demetrius Costa Rapello. **Sistema de recomendação matérias de segundo nível.** Rio de Janeiro, 2011. 68p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.  Sistemas de recomendação têm sido um mecanismo amplamente utilizado pelos grandes portais de internet nos dias atuais. Este grande crescimento é decorrente do frequente aumento do volume de informações disponíveis na web.  Este grande volume de informação impede que os usuários obtenham rapidamente o conteúdo que desejam, com isso ferramentas que facilitam a tomada de decisão tornam-se cada vez mais indispensáveis. Dentro deste contexto, os grandes portais de internet têm se utilizado de diferentes tipos de sistemas de recomendação para separar o conteúdo indesejado e sugerir informações relevantes para os seus usuários.  Neste trabalho apresentamos um modelo de sistema de recomendação de segundo nível, que visa auxiliar a equipe de jornalistas do portal de notícias G1 da globo.com, no processo de recomendação de notícias relacionadas para os usuários do site.  O modelo consiste na recomendação de matérias com base em **features** extraídas do próprio texto, que são combinadas para melhorar a recomendação. As **features** extraídas permitem a criação de consultas que são construídas na sintaxe **lucene** e são executadas contra nosso servidor de busca em **Solr**. O resultado da consulta é uma lista de matérias candidatas que é ordenada por: um **score** e data de publicação. Essa listagem é ainda reordenada com base em critérios de similaridade textual, onde as cinco matérias melhor posicionadas são entregues para o editor como recomendação. |
| Palavras-chave  Sistemas de Recomendação; Recuperação da Informação; Extração de Entidades Nomeadas. |

|  |
| --- |
| Abstract |
| Demetrius Costa Rapello. **Second level news recommended system.** Rio de Janeiro, 2011. 68p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.  ESCREVER. |
| Keywords  Recommended Systems; Information Retrieval; Named Entity Extraction. |

1. Introdução 13

1.1. Objetivo da dissertação 14

1.2. Organização da dissertação 14

2. Conceitos e trabalhos relacionados 16

2.1. Sistemas de recomendação 16

2.1.1. Baseados em conteúdo 17

2.1.2. Baseado em filtragem colaborativa 17

2.1.3. Baseado em nichos demográficos 18

2.1.4. Baseado em conhecimento 18

2.1.5. Baseado na comunidade 18

2.1.6. Sistemas híbridos de recomendação 18

2.2. Vector Space Model 19

2.3. Classificador Naive Bayes 19

2.4. Part-of-speech Tagging 21

2.5. Projeto Pure 21

2.5.1. Interface para registro de artigos do usuário 22

2.5.2. Treinamento do modelo probabilístico 22

2.5.3. Recuperação diária de novos artigos da base PubMed. 23

2.5.4. Recomendação dos artigos 24

2.6. Query by Document 24

3. GRNews – Sistema de recomendação matérias de segundo nível. 26

3.1. Arquitetura do sistema e sumário 26

3.2. Extrator de features 29

3.2.1. Termos mais frequentes 30

3.2.2. Tags HTML informativas 31

3.2.3. Texto em títulos de vídeos e fotos 32

3.2.4. Reconhecimento de entidades nomeadas 33

3.3. Seleção de candidatos 43

3.3.1. Definição dos critérios de filtragem e ordenação 43

3.3.2. Combinação de features 45

3.3.3. Recuperação das candidatas 46

3.4. Recomendação 47

3.4.1. Primeiro aspecto: Conceito de “está relacionada” 47

3.4.2. Segundo aspecto: Desempenho da comparação 48

3.5. Comentários do projeto 48

4. Experimento 51

4.1. O corpus 51

4.2. Critério para avaliação do sistema 52

4.3. Resultados obtidos 53

5. Conclusões e trabalhos futuros 59

5.1. Principais contribuições 60

5.2. Limitações do sistema 60

5.3. Trabalhos futuros 61

6. Bibliografia 62

7. APÊNDICE A 65

|  |
| --- |
| Lista de imagens |
|  |

Figura 1 – Modelo do classificador 20

Figura 2 – Exemplo de frases substantivas 25

Figura 3 – GRNews arquitetura 26

Figura 4 – Interface web do formulário de matéria 27

Figura 5 – Componente de matérias recomendadas 29

Figura 6 – Exemplo de matéria 32

Figura 7 – Processo de extração de entidades 36

Figura 8 – Acurácia do POS-tagger 39

Figura 9 – Resultado do classificador 1 40

Figure 10 – Resultado do classificador 2 40

Figura 11 – Exemplo de matéria publicada 46

Figura 12 – Tabela de acurácia por editoria com 5 recomendações e sem o fator de similaridade 66

Figura 13 – Tabela de acurácia por editoria com 5 recomendações com o fator de similaridade 67

Figura 14 – Tabela de acurácia por editoria com 10 recomendações e sem o fator de similaridade 68

Figura 15 – Tabela de acurácia por editoria com 10 recomendações e com o fator de similaridade 69

|  |
| --- |
| Lista de tabelas |

Tabela 1 – Campos do formulário de matéria 28

Tabela 2 – Formatos de n-grams 30

Tabela 3 – Comparativo de extratores 35

Tabela 4 – Lista de n-grams 37

Tabela 5 – Corpora para reconhecer entidades 38

Tabela 6 – Comparativo de entidades extraídas 41

Tabela 7 – Comparativo entre extratores 42

Tabela 8 – Exemplo de combinação de features 45

Tabela 8 – Representação do corpus 52

Tabela 9 – Acurácia por feature com 5 recomendações sem similaridade 54

Tabela 10 – Acurácia por feature com 5 recomendações com similaridade 56

Tabela 11 – Acurácia por feature com 10 recomendações sem similaridade 57

Tabela 12 – Acurácia por feature com 10 recomendações com similaridade 58

Lista de funções

Função 1 – Fórmula de distância dos cossenos 19

Função 2 – Função de recomendação do PURE 23

Função 3 – Cálculo do zcore do PURE 24

Função 4 – Cálculo de score do lucene 44

# Introdução

Estudos indicam que o volume de páginas indexadas nos principais buscadores (**Google** e **Yahoo**) aproximam-se da casa dos 20 bilhões [2,3]. Desta forma, podemos deduzir que o volume de conteúdo duplicado ou até mesmo de conteúdo não relevante também é grande.

Assim sendo, não é difícil perceber que os usuário estão perdendo mais tempo para encontrar as informações que desejam e estão cada vez mais dependentes de mecanismos de busca que, de certa forma, diminuem o tempo gasto na procura da informação. Ou seja, como os usuários não tem condições naturais de absorver todo esse conteúdo produzido, fica cada vez mais clara a necessidade de mecanismos que auxiliem o processo de tomada de decisão na web.

Os sistemas de recomendação são sistemas responsáveis por facilitar o processo de tomada de decisão quando buscamos por informações na web.

Estes sistemas procuram organizar as informações baseados em lógicas previamente definidas que fazem sentido no contexto da informação procurada. Por exemplo, quando estamos navegando na web em um site de compra de jogos e, nos deparamos com uma lista dos dez jogos mais procurados pelos usuários ou quando estamos numa locadora virtual e verificamos uma lista com os filmes mais vistos, estamos na verdade recebendo recomendações para facilitar nossas decisões.

Sistemas de recomendação têm sido largamente utilizados para atender a necessidade de tomada de decisão, pois em geral estes sistemas são construídos com o intuito de predizer as necessidades do usuário, oferencendo-lhes conteúdos com uma maior probabilidade de aproveitamento.

A necessidade de apoio a tomadade decisão na busca de informações na web, permite-nos afirmar que a aplicação de mecanismos de recomendação em sites de conteúdo noticioso como o portal G1, tendem a trazer bons resultados para os seus usuários do ponto de vista de informações relevantes.

## Objetivo da dissertação

Diante deste cenário, propomos um sistema de recomendação de segundo nível que auxilie os editores do portal de notícias G1 sugerindo conteúdos relacionados aos novos artigos produzidos com base apenas nas informações extraídas durante a confecção da nova notícia.

Atualmente a recomendação já é realizada pela equipe de editores do portal G1, entretanto o procedimento de recomendação de notícias é manual, o que significa que o editor precisa utilizar um procedimento para realizar a recomendação.

Este procedimento consiste em: identificar palavras chave relevantes ao texto, para em seguida consultar no sistema de busca interna ou mesmo no google por textos que falem do mesmo assunto. Depois de lidos os textos e entendido que alguma similaridade existe entre eles, o editor copia seus links e os cola em uma área especificada dentro do novo texto produzido. A esta área damos o nome de “**saibamais**”.

Como pode ser observado, o processo de produção de conteúdo se torna mais lento porque o jornalista passa a dividir seu tempo entre a produção e a mineração de textos e recuperação de informação para realização de relacionamentos de conteúdos.

A expectativa do nosso sistema é prover um mecanismo automático de recomendação, que forneça ao jornalista uma lista de possíveis matérias relacionadas, de modo a minimizar o tempo gasto na produção de notícias da redação.

O objetivo intermediário é promover uma melhora no conteúdo relacionado utilizando diferentes critérios para geração das recomendações.

## Organização da dissertação

O restante deste documento está organizado em cinco capítulos, da seguinte forma:

O **Capítulo 2** apresenta o estado da arte em sistemas de recomendação e as ferramentas e *frameworks* existentes comparados a ferramenta proposta.

O **Capítulo 3** descreve a ferramenta desenvolvida para apoiar a recomendação de conteúdo na produção de matérias do G1.

O **Capítulo 4** apresenta os resultados obtidos com a experimentação da ferramenta contra um corpus real.

O **Capítulo 5** apresenta as conclusões e trabalhos futuros.

# Conceitos e trabalhos relacionados

Durante a fase de desenvolvimento desta pesquisa não foram observados trabalhos similares a recomendação de segundo nível. Contudo, destacamos neste capítulo dois trabalhos que apresentaram aspectos parecidos com a nossa proposta. Ainda neste capítulo, vamos apresentar os principais conceitos utilizados no decorrer da dissertação.

## Sistemas de recomendação

Sistemas de recomendação são softwares especializados em apresentar “opções” para serem usadas por seus usuários [5,6,18]. Esses sistemas auxiliam pessoas que não possuem muita experiência ou competência para pesquisar dados acerca de um determinado assunto.

Em linhas gerais, os sistemas de recomendação procuram oferecer as melhores opções de resposta para as necessidades dos usuários num processo de tomada de decisão.

Os sistemas de recomendação podem oferecer sugestões em diferentes domínios como por exemplo: que item comprar em uma loja virtual, que artigo ler em um site de notícias ou mesmo que restaurante visitar em uma cidade.

No sistema de recomendação do **IMDB** (**Internet Movie Database**), quando o usuário seleciona um filme do catálogo para ler a respeito, o sistema apresenta também uma lista de sugestões de filmes relacionados para o usuário.

Assim como o **IMDB**, o site de compras **Amazon** apresenta para cada produto selecionado pelo usuário, uma lista de recomendações de outros produtos que possam servir para o usuário dentro de uma mesma compra.

Sistemas de recomendação são usualmente personalizados com base em características individuais ou coletivas, todavia, os sistemas de recomendação não personalizados também tem o seu espaço. Sistemas de recomendação não personalizados são mais simples de serem implementados e geralmente são usados para a recomendações mais gerais como por exemplo, a lista dos “10 mais” de um determinado assunto ou tema.

Sistemas de recomendação personalizados tentam antecipar as necessidades do usuário analisando informações do seu perfil e levando em consideração as restrições de domínio para então recomendar sugestões. Ainda no site de compras da **Amazon**, uma vez realizada uma compra, as informações são armazenadas no perfil do usuário de modo que para as compras futuras o usuário receba como recomendação, não apenas itens relacionados pela categoria em que se encaixam, mas também por que estão relacionados com a última compra feita por ele.

Sistemas de recomendação variam de acordo com a técnica utilizada para descobrir as preferências do usuário. Entre os sistemas de recomendação mais comuns temos:

### **Baseados em conteúdo**

Sistemas de recomendação baseados em conteúdo tentam recomendar opções que são similares a algum item que o usuário já selecionou no passado usando como **features** informações extraídas do produto/conteúdo que o usuário consultou. Por exemplo, se um usuário leu um artigo sobre política em um portal de notícias, este portal poderia sugerir em consultas futuras artigos relacionados a política.

Segundo Yuanhaua et al. [1] os sistemas de recomendação baseados em conteúdo junto com os baseados em filtragem colaborativa são os mais difundidos e utilizados entre os sistemas de recomendação.

### **Baseado em filtragem colaborativa**

Sistemas de recomendação baseados em filtragem colaborativa, tentam recomendar ao usuário opções que foram utilizadas por usuários com os mesmos interesses. Por exemplo, se um grupo de usuários selecionam o produto X e em seguida selecionam o produto Y, o sistema de recomendação entende que usuários que acessam o produto X também acessam o produto Y e passa a recomendar o produto Y toda vez que algum novo usuário se interessar pelo produto X.

### **Baseado em nichos demográficos**

Este tipo de sistema de recomendação baseia-se no perfil demográfico do usuário para recomendar opções. Usuários de um determinado estado recebem recomendações diferentes de usuários de outros estados. Os nichos demográficos podem ser: idade, sexo, língua etc.

### **Baseado em conhecimento**

Sistemas de recomendação baseados em conhecimento tentam recomendar ao usuário opções baseando-se no conhecimento específico do domínio do sistema. Este modelo também é reconhecido como um modelo baseado em caso onde o problema é a analise das necessidades do usuário e a solução é o conjunto de opções a serem recomendadas.

Sistemas de recomendação com esta técnica tendem a trabalhar melhor que os outros no início, porém se não são acompanhados de componente de aprendizagem se tornam ineficientes.

Neste sistema o principio de recomendação está baseado na similaridade de uma solução para um dado problema.

### **Baseado na comunidade**

Este tipo de sistema de recomendação procura basear-se nas relações que o usuário possui com sua rede de amigos para realizar as recomendações. Neste modelo acredita-se que a recomendação de pessoas ligadas ao usuário tendem a ser mais efetivas. Esta abordagem tem se tornado bastante atrativa tendo em vista o grande crescimento das redes de relacionamento.

### **Sistemas híbridos de recomendação**

Sistemas de recomendação híbridos procuram utilizar técnicas de outros sistemas de recomendação de forma a complementar as deficiências entre as técnicas.

## Vector Space Model

**Vector space model** é um modelo proposto por Gerard Salton que é muito utilizado no campo de recuperação da informação. É um modelo onde documentos em textos são representados na forma de vetores de termos em que cada termo é composto de uma tupla (palavra, peso).

Segundo Christopher D. Manning et al. [12], os vetores mantém todas as palavras da coleção de documentos e não somente as palavras do documento em si. Assim as palavras que estão na coleção dos documentos mas não estão no documento que está sendo transformado em vetor recebem o peso 0. As demais palavras rebem um peso que significa um grau de importância e pode ser concebido de diversas formas. A forma mais comum de atribuição de pesos é o **TF/IDF** que mede a relação entre a frequência de um termo dentro do documento (**TF)** pelo inverso da frequência da palavra em todos os documentos da coleção (**IDF)**.

Os Vetores de documentos são então plotados em um espaço euclidiano onde cada termo do vetor representa uma coordenada no plano.

Desta forma, a similaridade entre dois vetores de termos pode ser conhecida através da fórmula de distância dos cossenos que procura medir o ângulo formado entre os pontos dos dois vetores de termos.

A distância dos cossenos é o resultado do produto escalar dos vetores dividido pelo produto das suas magnitudes:

Função 1 – Fórmula de distância dos cossenos

O fator de similaridade entre dois vetores estará entre 0 e 1, desde que os pesos não sejam negativos.

## Classificador Naive Bayes

Um classificador Naive Bayes é um classificador probabilístico baseado na aplicação do teorema de Bayes. Para melhor entender o Naive Bayes, vamos antes relembrar o conceito de classificação.

Classificação é uma das técnicas existentes em aprendizado de máquina, que segundo Jacob Perkins [7] é definida como a tarefa de atribuir um determinado rótulo a um dado texto de entrada de acordo com o reconhecimento de um padrão existente. Ou seja, o classificador aprende um padrão e com isso rotula as instâncias que correspondem a esse padrão.

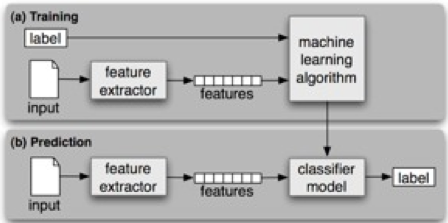


Figura 1 – Modelo do classificador

O aprendizado do classificador pode ser supervisionado ou não supervisionado.

Para este trabalho vamos nos limitar ao aprendizado supervisionado onde o classificador utiliza um corpus de treinamento que o ajuda a inferir padrões para serem usados mais adiante no processo de classificação.

Os padrões são encontrados pelo classificador com o auxilio de **features** que são passadas para ele durante o aprendizado supervisionado.

As **features** são características de informações que possibilitem ao classificador o entendimento do corpus.

Por exemplo: se pegarmos como teste um classificador que reconhece a probabilidade de uma fruta ser uma maça, podemos definir como **features** deste exemplo o peso da fruta, a cor e o seu formato. Estas **features** são facilmente identificadas por serem comuns a outras frutas variando-as umas das outras.

De forma simples, um classificador bayesiano é um classificador probabilístico que parte do pressuposto que a presença ou ausência de uma feature de uma classe não esta relacionada a presença ou ausência de outras features da mesma classe, daí o termo ***naíve***.

## Part-of-speech Tagging

Segundo Jacob Perkins [7], **part-of-speech tagging** é o processo de converter uma sentença no formato de uma lista de palavras para uma lista de tuplas onde cada tupla é composta de uma palavra e uma **tag.** Neste caso a **tag** refere-se a classe gramatical da palavra.

Em outras palavras, o **POS-tagger** é o mecanismo desenvolvido para identificar a classe gramatical de palavras em um determinado contexto.

A dificuldade nesta tarefa se dá porque as palavras podem ter mais de um **POS-tag** possível e com isso é indispensável o entendimento do contexto onde a palavra foi empregada para que haja a desambiguação da tag.

Na literatura encontramos o trabalho de Kepler e Finger [24], que apresentam a implementação de um **POS-tagger** para língua portuguesa com acurácia 95,51%.

## Projeto Pure

O projeto **PURE** [15] é um sistema para recomendação de artigos médicos que utiliza o princípio de recomendação baseado na filtragem de conteúdo. Em linhas gerais, o sistema executa uma classificação dos artigos preferidos do usuário para então recomendar outros artigos para leitura.

O sistema PURE pode ser entendido pelo fluxo de operações a seguir:

1. O usuário acessa o sistema para informar os artigos do seu interesse. Estes artigos são armazenados na base de dados do **PURE**.
2. Um sistema de aprendizado de máquina é aplicado para extrair as preferências do usuário com base nos seus artigos de interesse.
3. O sistema **PURE** consulta a base de dados **PubMed** para baixar os novos artigos publicados.
4. Os artigos baixados da **PubMed** são ordenados com base no modelo treinado com as preferências do usuário.
5. Os artigos são então apresentados para o usuário.

O propósito do **PURE** desenvolvido por Takashi Yoneya e Hiroshi Mamitsuka [15], é oferecer uma interface que facilite o processo de busca de artigos relevantes pela comunidade científica na base de dados **PubMed**. A base de dados **PubMed** mantém um grande acervo de artigos de biologia e medicina com um volume diário de atualização da ordem de centenas artigos. Diante deste cenário, os autores perceberam que um sistema de recomendação baseado em filtragem de conteúdo poderia auxiliar a comunidade científica no trabalho de consulta de novos artigos. Os módulos a seguir fazem parte do sistema **PURE**.

### **Interface para registro de artigos do usuário**

Para utilizar o sistema, o usuário precisa registrar os seus artigos de interesse na base de dados do **PURE**. Para esta atividade o usuário acessa a interface web do sistema e seleciona os artigos de sua preferência em uma listagem. Os artigos selecionados são então gravados no perfil do usuário e armazenados no banco de dados do **PURE**.

O usuário tem a permissão de adicionar novos arquivos e alterar sua lista de interesse.

### Treinamento do modelo probabilístico

Os artigos de interesse do usuário são utilizados para a concepção de um modelo probabilístico que procura identificar as preferências do usuário para novos artigos. Esse modelo é dividido em duas etapas:

#### **Seleção de palavras e atribuição de peso**

Nesta etapa o sistema trata os artigos do **PubMed** como um vetor de palavras ordenadas por peso. Estas palavras são obtidas a partir da eliminação de palavras irrelevantes para o sistema classificadas como stopwords.

As stopwords são obtidas por duas estratégias distintas, a primeira consiste na geração do **DF** e **TF-IDF** das palavras oriundas de uma porção aleatória de artigos da base de dados do **PubMed**. As palavras com alto **DF** (número de documentos onde a palavra aparece) ou com baixo **TF-IDF** (métrica que informa quanto a palavra é relevante para o documento) são consideradas stopwords. A segunda estratégia consiste em considerar como stopwords as palavras que respeitam as seguintes regras: 1) palavras com menos de 3 letras; 2) palavras sem caracteres alfabéticos; 3) palavras que aparecem no **Journal of Business Research** de Jan. 2005 a 2006.

Após a eliminação das stopwords é dado um peso para cada palavra restante do documento. Este peso é obtido pela verificação da distribuição da palavra pelo documento (**TF**).

#### **Etapa de Geração do modelo probabilístico**

As palavras selecionadas na etapa anterior são usadas para treinar um classificador probabilístico que será usado para gerar uma métrica de recomendação para os novos artigos. A função utilizada para computar o grau de recomendação de um artigo é dada pela fórmula abaixo:

Sendo **d** um artigo, **z** a variável correspondente ao cluster, **s** um campo existente na estrutura do artigo exe., título e **w** uma palavra do artigo temos:

Função 2 – Função de recomendação do PURE

Em seguida os autores treinam os parâmetros de probabilidade p(z) e ps(w|z) a partir dos artigos preferidos do usuário utilizando o algoritmo de Maximização de Expectativa (EM).

### **Recuperação diária de novos artigos da base PubMed.**

O sistema **PURE** diariamente executa uma operação de recuperação dos novos artigos publicados na base de dados do **PubMed**. Os novos artigos são armazenados na base de dados do sistema para serem classificados de acordo com as preferências de cada usuário.

### **Recomendação dos artigos**

Para cada artigo recuperado, são extraídas palavras que são usadas como base para geração do critério de ordenação que é dado de acordo com a função de recomendação definida anteriormente. Como forma de ajustar possíveis desvios do algoritmo de recomendação, os autores apresentam um score adicional para cada artigo. O Z-score é obtido através do agrupamento dos artigos em conjuntos de artigos com o mesmo número de palavras. Deste modo é separado o mínimo e o máximo graus de recomendação do grupo onde o Z-score do artigo é dado pela formula:

Função 3 – Cálculo do zcore do PURE

Os artigos com o maior Z-score são então recomendados para o usuário.

## Query by Document

Neste trabalho os autores defendem uma técnica de recuperação de conteúdo relacionado utilizando informações existem no texto para consulta na base de dados.

A primeira ideia dos autores Yin Yang et al. consiste em é extrair do texto o que eles chamam de “frases substantivas” que possam ser relevantes para pesquisar por conteúdo relacionado. Em seguida eles defendem que estas frases podem ser substituídas ou melhoras através do uso de fontes externas, no caso, o **Wikipédia**.

Para melhorar as frases com o uso do **Wikipédia**, eles utilizam um grafo de conceitos do **Wikipédia** onde os nós também representam frases substantivas de modo que eles podem recuperar as frase presentes no texto e em seguida melhorá-las percorrendo o grafo e substituindo-as por novos conceitos.

Para extrair as frases substantivas, eles se utilizaram de um **POS-tagger** onde eles consideraram todas as frases cujo padrão de formação respeitava o conceito de frase substantiva, definido no trabalho.

A figura a seguir mostra como os padrões de frases substantivas foram definidos por eles:

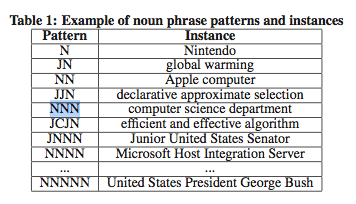


Figura 2 – Exemplo de frases substantivas

A partir deste ponto as frases são ordenadas de acordo com um **score** que é dado por dois mecanismos distintos. O primeiro mecanismo utiliza o **TF/IDF** dos termos na frase para atribuir um **score**, enquanto que o segundo computa o **score** baseado nas informações mutuas dos termos da frase.

Para validar o experimento os autores utilizaram o serviço **Mechanical Turk** da **Amazon** para avaliar a qualidade do documentos recomendados com uso das frases substantivas.

Basicamente, eles informavam o texto e os top 5 documentos que foram recuperados com uso das frases substantivas para os usuários, pedindo para que eles avaliassem se os documentos retornados eram relacionados ao texto ou não.

# GRNews – Sistema de recomendação matérias de segundo nível.

**GRNews** é um sistema de recomendações de matérias de segundo nível que visa prover aos jornalistas editores do portal G1 uma função para a geração de matérias relacionadas ao texto produzido. Esta geração leva em consideração apenas a análise de informações existentes no texto. Com estas informações, o sistema estabelece critérios de consulta de conteúdo relacionado e executa estas consultas contra a base de dados de matérias indexada no sistema de busca interno **Solr**. Os resultados obtidos são organizados em ordem de data de publicação decrescente e exibidos em uma listagem para o editor. Desta forma o editor pode relacionar conteúdos de diferentes critérios para compor o elemento de matérias relacionadas.

Neste capítulo discutiremos a arquitetura do sistema, bem como os seus componentes internos de modo a entender todo o seu funcionamento.

## Arquitetura do sistema e sumário

O principal objetivo do **GRNews** é gerar conteúdo relacionado a um determinado texto que é produzido pelo jornalista. Para isso o sistema conta com 3 componentes principais como pode ser visto na figura 3.

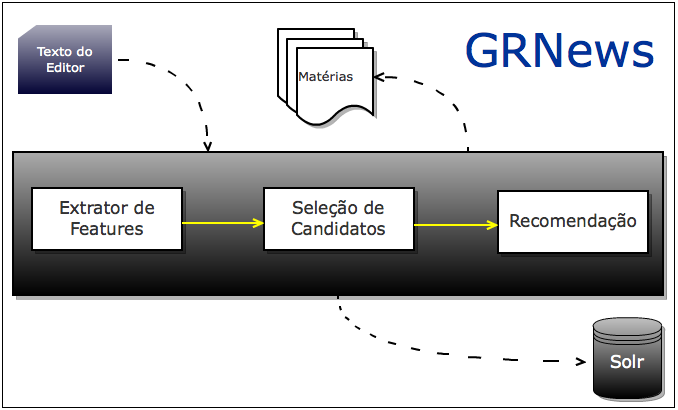


Figura 3 – GRNews arquitetura

Para acionar o motor do **GRNews** é necessário que o editor informe os parâmetros de entrada e para isso, o ponto de partida é o formulário de inclusão de matéria. O formulário de inclusão de matéria mostrado na figura 4, é apresentado ao editor como uma interface web. O formulário solicita ao editor o preenchimento de um conjunto de informações divididas em obrigatórias e não obrigatórias.

As informações obrigatórias são necessárias para configurar uma notícia que pode ser publicada.

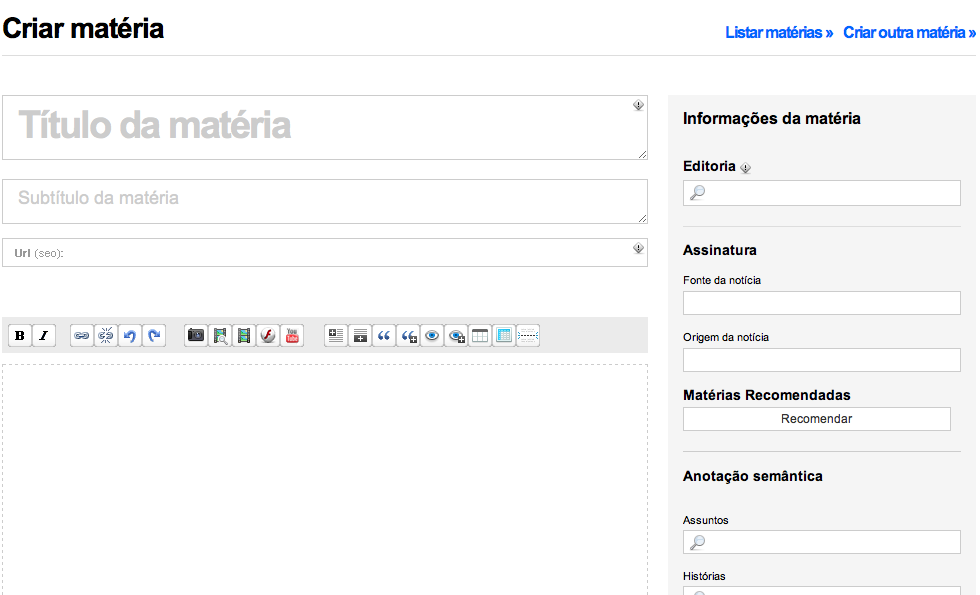


Figura 4 – Interface web do formulário de matéria

Dentre as informações solicitadas pelo formulário, vamos apresentar na tabela 1, apenas aquelas que serão usadas pelo sistema **GRNews** por entendermos que são suficientes para realizarmos a função de recomendação proposta.

Tabela 1 – Campos do formulário de matéria

|  |  |  |
| --- | --- | --- |
| Título | O título da notícia é um dado textual limitado em 255 caracteres alfa numéricos. | Obrigatório |
| Subtítulo | O subtítulo da notícia é um dado textual limitado em 1000 caracteres alfa numéricos. | Obrigatório |
| Corpo | O corpo da notícia é um dado textual sem limite de caracteres alfa numéricos. O Editor pode fazer uso de **mark-up** html na construção do conteúdo da notícia. | Obrigatório |
| Editoria Principal | Dado que agrupa o conjunto de notícias. Esta informação é selecionada na interface através de um caixa de seleção. Este agrupamento é criado pelos editores e geralmente refletem seções do site na internet. Exemplo de Editoria para o G1: Rio de Janeiro, Economia, Mundo | Obrigatório |
| Editorias Secundárias | Agrupamentos secundários para organização das notícias | Não Obrigatório |
| Entidades Associadas | Informações sobres entidades que são passadas manualmente pelo editor para enriquecimento do conteúdo | Não Obrigatório |

Tendo preenchido o formulário de matéria de acordo como as regras descritas na tabela 1, o editor pode solicitar a geração das matérias relacionadas.

A geração de matérias relacionadas é realizada através do acionamento do botão recomendar existente no lado direto do formulário de matéria, como pode ser visto na figura 4.

O editor pode optar por simular critérios para recuperação da informação utilizando-se das caixas de seleção de critérios logo acima do botão recomendar. Caso o usuário não selecione nenhum critério o sistema realiza a recomendação com base em um critério **default**. Após a recomendação, o sistema exibe uma lista das cinco matérias encontradas ordenadas por data de publicação. Caso não existam matérias relacionadas o sistema não apresenta o elemento de listagem.



Figura 5 – Componente de matérias recomendadas

## Extrator de features

No contexto do nosso trabalho, as **features** são características ou aspectos de um exemplo e a sua principal utilidade é a identificação de padrões que nos levem a extrair um melhor entendimento da matéria. Em nosso sistema, onde tratamos com documentos em texto, podemos utilizar como exemplo de **feature** a frequência de exibição das palavras no texto para descobrir quais palavras são mais relevantes para serem utilizadas como padrões de consulta de matérias relacionadas.

Escolher as melhores **features** é uma atividade de extrema importância para o sucesso da pesquisa.

Nossa **features** são baseadas em informações textuais extraídas do corpo da matéria definidas como **N-grams**.

**N-gram** ou **N-grama** em português, é o nome dado a um conjunto de palavras em sequência obtidos a partir de um texto. Os **n-grams** são classificados de acordo com o total de palavras que os compõem:

Tabela 2 – Formatos de n-grams

|  |  |  |
| --- | --- | --- |
| Brasil, Acre | Unigram | Uma palavra |
| São Paulo, Rio Negro | Bigram | Duas palavras |
| Rio de Janeiro | Trigram | Três palavras |
| São José de Itabapoana | Quadrigram | Quatro palavras |

Embora existam **n-grams** com números maiores de palavras, iremos adotar para este trabalho apenas **n-grams** com até três palavras.

Abaixo descrevemos algumas das features que usaremos no projeto **GRNews**.

### Termos mais frequentes

É o conjunto dos termos do documento ordenados por sua frequência de aparição no texto. Segundo Christopher D. Manning et al. [12], a frequência de um termo é o número de vezes que este termo aparece no documento.

A ideia é que um termo que aparece com muita frequência no texto pode indicar o tema central deste texto. Entretanto, deve-se levar em conta como dito por Jacob Perkins [7], que alguns termos que aparecem como mais frequência podem não apresentar um significado para o texto por serem muito comuns. Para corrigir este desvio, são utilizadas técnicas de exclusão desses termos que apresentam pouco significado. Uma dessas técnicas chamada **stopword** **list**, consiste na elaboração de uma listas de exclusão baseadas nos termos mais frequentes encontrados em uma coleção de documentos.

Uma vez removidos os termos não relevantes o texto é então submetido a duas etapas em sequência: limpeza e **tokenização**.

A etapa de limpeza, consiste na remoção do ruído presente no texto. São removidos os caracteres de pontuação bem como os marcadores html. Após esta etapa estamos aptos a realizar a tokenização do texto.

Ainda de acordo com Christopher D. Manning et al. [12], **tokenização** é o processo de quebrar o documento em pedaços baseando-se em um padrão de corte.

Durante nosso processo de tokenização, os termos são separados a medida que um espaço em branco aparece entre eles e são colocados em uma lista de termos também conhecido como **bag of word**.

A partir do **bag of word**, os **n-grams** são separados de acordo com o número de palavras nos seus respectivos vetores de **unigrams**, **bigrams** e **trigrams**. Depois, o sistema realiza uma contagem da aparição destes **n-grams** no documento e ao final ordena a lista partindo dos **n-grams** mais frequentes para os menos frequentes.

### Tags HTML informativas

**Tags HTML** informativas são marcadores em html que enfatizam a importância de um pedaço de texto dentro de um documento. Entendemos que estas **tags** traduzem o grau de relevância que o termo possui dentro do documento.

As **tags** reconhecidas como informativas para este trabalho são <**em**> que representa os textos escritos em itálico e <**strong**> que representa dos textos escritos em negrito.

As **tags** informativas são procuradas dentro do documento construído pelo editor e seus textos são tokenizados para extração dos **n-grams** existentes. Os **n-grams** encontrados são armazenados em um vetor de termos que é posteriormente utilizado no processo de seleção de matérias candidatas.

De acordo com a figura 6 abaixo, serão selecionados os termos “*Biblioteca Pública Dolor Barreira*” e “*Festival de Cinema e Cultura da Diversidade Sexual*.”



Figura 6 – Exemplo de matéria

### Texto em títulos de vídeos e fotos

Os termos que ocorrem dentro de estruturas que apresenta vídeos ou imagens podem representar fontes de informação valiosa para descobrir ou certificar qual é o tema central do documento.

Para isso o sistema procura pelas estruturas definidas com marcadores html que definem estes elementos afim de separar as informações existentes.

Os textos encontrados nestas estruturas são tokenizados para extração dos **n-grams** existentes. Os **n-grams** encontrados são armazenados em um vetor de termos.

Na figura 6 o texto: “**Quinta edição do festival For Rainbow começa nesta quinta-feira (27) (Foto: Divulgação**)”, será selecionado por representar o título da foto da matéria.

### Reconhecimento de entidades nomeadas

Durante a etapa de levantamento de **features**, observamos a necessidade de extrair do texto informações que representassem entidades nomeadas. A hipótese levantada era que essas informações poderiam ajudar no processo de recomendação de matérias. Contudo, de acordo com Richman [14], identificar entidades nomeadas é uma das tarefas mais importantes e complexas do processamento de linguagem natural. A maioria das pesquisas nesta área são restritas a um conjunto pequeno de idiomas e quase todos os métodos requerem um conhecimento linguístico refinado. Ainda, de acordo com Hiroyuki Toda e Patrick Schone [18], a tarefa de extração de entidades criada nos anos 1990 tem como objetivo o reconhecimento de unidades de informação importantes tais como: nomes de pessoas, nomes de organizações, nomes de localizações, datas, valores financeiros e etc.

A dificuldade aumenta nesta área por conta da falta de corpus para algumas línguas o que é o caso da língua portuguesa. Alguns trabalhos em língua portuguesa foram observados como por exemplo, o trabalho de Luiz Claudio Gomes Maia [13], que utiliza sintagmas nominais para classificação automática de documentos.

Para o reconhecimento das entidades nomeadas nos textos das matérias utilizamos duas abordagens:

A primeira abordagem foi procurar no mercado por ferramentas que pudessem atender a esta demanda. Estas ferramentas foram estudas e são explicadas no tópico **Ferramentas de mercado**.

A segunda abordagem foi desenvolver uma ferramenta própria para desempenhar essa função. No tópico **Ferramenta desenvolvida**, falamos a respeito do desenvolvimento e dos resultados obtidos.

Ao final das duas abordagens, apresentamos os motivos que levaram a utilização de uma abordagem em relação a outra.

#### Ferramentas de mercado

Procuramos na internet por ferramentas que realizassem a função de extração de entidades nomeadas a partir de um texto submetido. Nossa ideia era enviar todo o corpo da matéria e receber de volta apenas as entidades que foram encontradas.

Embora não fosse o critério mais importante, optamos por serviços web ao invés de **API**’**s**. Nesta linha encontramos quatro serviços que atendiam a demanda e estudamos suas implementações em nosso projeto.

A seguir vamos falar sobre os quatro serviços estudados:

##### Yahoo Term Extraction

O serviço do **yahoo** de extração de termos permite a seus usuários o acesso a uma **API** para análise de textos que fornece ao final uma lista de palavras ou frases relevantes em inglês extraídas do texto submetido. O serviço pode ser acessado através do protocolo **REST** e responde os dados de saída em formatos **XML** e **json**.

O serviço é gratuito. Porém, possui limite de requisições diárias estipulado em 5000 requisições/dia.

O serviço pode ser utilizado mediante o cadastramento e obtenção da chave de acesso.

##### Nltk

É um serviço web que funciona sob o protocolo **REST** para mineração de texto e processamento de linguagem natural. A **API** foi concebida com base nas premissas do **NLTK** **cookbook** e não tem fins comerciais de modo que, possui limites tanto para o número de requisições (1000 requisições diárias) quanto para o tamanho do texto enviado (10000 caracteres). O formato de saída pode ser em **XML** ou **json**.

##### Ltasks

Também é um serviço web que funciona sob o protocolo **REST** e que apresenta várias possibilidades de extração de informação do texto entre elas o reconhecimento de entidades nomeadas em língua portuguesa. Para utilização on-line é necessária a utilização de uma chave de acesso que é obtida através de um cadastro no site.

##### Zemanta

É uma ferramenta concebida para geração de conteúdo relacionado para **blogs**. Contudo, seus idealizadores proveem uma **API REST** que permite a extração de entidades contextualizadas ao texto submetido. Para fazer uso do serviço é necessário um cadastro e a obtenção de uma chave de acesso. O serviço é, em princípio, independente de idioma.

Abaixo segue a tabela comparativa das principais características dentre as ferramentas observadas:

Tabela 3 – Comparativo de extratores

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Características | **ltasks** | **yahoo** | **nltk** | **zemanta** |
| Tem suporte ao idioma português | sim | não | sim | sim |
| Linguagem de desenvolvimento | Java | - | Python | - |
| Limite de acesso diário | - | 5000 | 1000 | 1000 |
| Tamanho máximo do texto | - | - | 10000 | 10000 |
| Possui código aberto | não | não | não | não |

Dentre as ferramentas observadas, o serviço **Ltasks** foi o que apresentou o melhor aproveitamento na extração de entidades. Porém, a ausência de um código aberto para aprimoramento do algoritmo, a limitação de acessos ao serviço e o tempo gasto em cada requisição são pontos desfavoráveis ao uso destas ferramentas.

#### Ferramenta desenvolvida

Dada a dificuldade de encontrar uma ferramenta eficiente que pudesse ser usada e alterada para atuar na atividade de extração de entidades nomeadas, partimos para o desenvolvimento de uma solução caseira utilizando conceitos de aprendizado de máquina.

A ferramenta de extração de entidades nomeadas desenvolvida possui um processo que separa as atividades em fases encadeadas onde dado um texto de entrada, o sistema processa e gera como saída uma lista de possíveis entidades reconhecidas.

Conforme a figura 7, a ferramenta de extração de entidades proposta possui 3 fases que são apresentadas na sequencia:

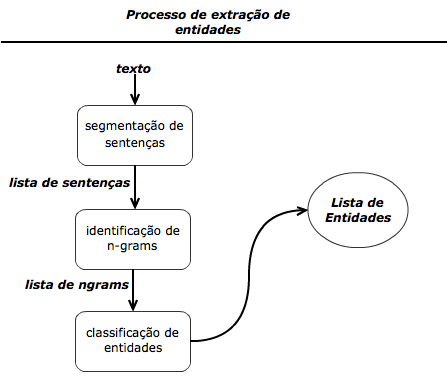


Figura 7 – Processo de extração de entidades

##### A fase de segmentação de sentenças

Nesta fase, o texto recebido como entrada é submetido a uma limpeza como visto anteriormente no processo de identificação de termos mais frequentes. Após a limpeza, o texto é quebrado em sentenças que indicam fim ou pausa do período. Esta quebra em sentenças é realizada por acreditarmos que as entidades aparecem como **n-grams** dentro dos períodos e nunca entre eles. Deste modos os caracteres de pontuação: **ponto final**, **ponto** e **ponto e virgula,** foram tratados como delimitadores de sentença.

Após a tokenização, as sentenças são armazenadas em um vetor de sentenças para ser processado na etapa seguinte.

##### A fase de identificação de n-grams

Nesta etapa as sentenças são tokenizadas em palavras e os **n-grams** são então separados de acordo com o número de palavras.

Considerações importantes a respeito dos **n-grams**:

1. Durante a identificação de **unigrams** (**n-gram** com apenas uma palavra), são removidas as palavras consideradas **stopwords**. Para tanto utilizamos o mesmo procedimento adotado para a montagem da lista de palavras mais frequentes.
2. Durante a identificação de **n-grams**, levamos em consideração algumas regras de formação dos **n-grams** para diminuir a lista de ocorrências possíveis, estas regras levam em consideração o **POS-tagger** treinado, que é explicado mais adiante, para revomer formações de **n-grams** inválidas como: verbos, artigos, adjetivos entre outros.
3. Durante a identificação dos **n-grams**, identificamos também a posição do **n-gram** dentro do texto de origem para efeito de eliminação de **n-grams** contidos em outros **n-grams**.

Ex. Dado o texto: **São José de Itabapoana é uma cidade linda**.

Se fizéssemos a separação dos **n-grams** sem levar em consideração a posição dos mesmos no texto acima, teríamos a seguinte conclusão:

Tabela 4 – Lista de n-grams

|  |  |
| --- | --- |
| São, José, Itabapoana, Cidade, Linda | Unigrams |
| São José, Cidade Linda | Bigram |
| José de Itabapoana | Trigram |
| São José de Itabapoana | Quadrigram |

Contudo, se observarmos a identificação das posições, teremos apenas os **n-grams** “*São José de Itabapoana*” e “*Cidade Linda*” selecionados. Respeitando-se as considerações explicadas anteriormente.

Assim, ao final da fase de identificação de **n-grams** são identificados **n-grams** distintos separados em vetores por seus respectivos números de palavras.

##### A fase de classificação de entidades

Para esta fase era necessário um componente que dado um **n-gram** identificado no texto, o componente respondesse ‘*sim’* se reconhecesse que este **n-gram** representava uma entidade ou ‘*não*’ caso contrário. Diante desta necessidade, ficou clara a ideia de construir um classificador.

Como não precisávamos descobrir que tipo de entidade estávamos tratando, se representava uma organização ou uma pessoa, decidimos desenvolver um classificador binário bayesiano capaz de responder a probabilidade deste **n-gram** ser ou não uma entidade nomeada.

A primeira ideia foi treinar o classificador com base na classe gramatical dos termos presentes no **n-gram**. Afim de reconhecer que um dado padrão de formação de **n-gram** tem grande probabilidade de ser ou não entidade. Com isso, utilizamos um corpus constituído de entidades nomeadas da base semântica da globo.com (tabela 5) para treinar o classificador.

Tabela 5 – Corpora para reconhecer entidades

|  |  |  |
| --- | --- | --- |
| PESSOAS | Políticos, Artistas e Personalidades de dentro e fora do brasil. | 31380 |
| ORGANIZAÇÕES | Empresas e Instituições | 1798 |
| LUGARES | Países e estados e cidades brasileiras e algumas cidades do mundo | 11290 |
|  |  | 44468 |

Para reconhecer a classe gramatical dos termos do **n-gram**, ainda precisávamos desenvolver uma outra parte do problema que era o **POS-tagger**.

Para treinar o nosso **POS-tagger** utilizamos o corpus **mac-morpho** disponível no Nltk toolkit.

O **MAC-MORPHO** é um corpus em língua portuguesa do brasil que possui milhões de palavras extraídas de artigos jornalísticos divididos em 10 seções diferentes do jornal a folha de São Paulo em 1994.

Para treinar o **POS-tagger**, utilizamos o módulo de **tagger** da API do nltk de modo que obtivemos a acurácia de aproximadamente 90% como mostra a figura abaixo:



Figura 8 – Acurácia do POS-tagger

Com o **POS-tagger** preparado, pudemos realizar o treinamento do classificador.

Para treinar o classificador, utilizamos duas abordagens diferentes na seleção de features. Na primeira abordagem, utilizamos 3400 matérias publicadas onde tokenizamos as suas sentenças e extraímos os **n-grams**. Para cada **n-gram** extraído, se fosse encontrado na lista de entidades já conhecidas atribuíamos um rótulo de entidade. Caso contrário recebia o rótulo de **n-gram**. Depois adicionávamos todas as entidades conhecidas na lista de **n-grams** de modo que forçamos o entendimento do classificador para o reconhecimento de alguns padrões.

Como dito anteriormente, para facilitar o trabalho do classificador, estabelecemos como features para esta primeira abordagem as seguintes características: “*Classe”* que guarda a classe gramatical dos termos do **n-gram**, “*Primeira Letra”* que guarda a informação se a primeira letra do **n-gram** é maiúscula ou minúscula e “*Tamanho”* do n-gram. Dessa forma obtivemos uma acurácia contra a base de testes de cerca de 92% como pode ser visto na imagem a seguir.



Figura 9 – Resultado do classificador 1

A segunda abordagem para treinar o classificador foi a utilização de sentenças completas extraídas do corpus do G1 com entidades anotadas pelos jornalistas. Em cada sentença, um determinado **n-gram** estava marcado como entidade. Desta forma, para cada entidade, selecionávamos além das características já selecionadas no treinamento anterior as informações pos1, pos2, pre1, pre2 que significam respectivamente as classes gramaticais dos termos uma posição a frente, duas posições a frente, uma posição atrás e duas posições atrás. Com esse classificador, obtivemos a acurácia de aproximadamente 94% como pode ser visto na figura abaixo:



Figure 10 – Resultado do classificador 2

Embora não tenha sido o propósito direto da dissertação, percebemos a importância do reconhecimento de entidades nomeadas no processo de recomendação de conteúdo relacionado. E por este motivo, um estudo mais elaborado neste campo pode trazer boas contribuições futuras.

Tanto o **POS-tagger** quanto o classificador de entidades, podem ser melhorados se aprimorados com técnicas mais eficientes de extração de features ou até mesmo com a utilização de outros modelos de aprendizado de máquina como o **SVM** ou o **Perceptron estruturado**.

Em linhas gerais, o classificador foi desenvolvido para reconhecer entidades apenas baseando-se no padrão de construção gramatical dos termos que compõem a entidade. Por exemplo, o **n-gram** “Rio de Janeiro”, possui a seguinte formação gramatical segundo o **POS-tagger** desenvolvido: **NPROP+PREP+N**. Desta forma a probabilidade obtida pelo nosso classificador para o **n-gram** ser uma entidade é de 98%.

Para nosso trabalho, foram tratadas como entidades **n-grams** com probabilidades acima de 80% para ser uma entidade.

Na tabela 6 temos um comparativo das entidades que foram reconhecidas pelos serviços estudados comparados ao nosso extrator. Podemos notar o desempenho do nosso classificador que foi capaz de reconhecer para o texto do exemplo todas as entidades mapeadas.

“*A* ***Polícia Federal*** *informou que foi instaurado um inquérito, após uma denúncia para o* ***DNPM*** *de crime de usurpação de bens públicos decorrentes da exploração ilegal de pedras preciosas.* ***Rogério Castro****, responsável pela agência de recursos naturais do* ***Ibama****, que atuava em* ***Minas Gerais****, foi destituído do cargo.”*

Tabela 6 – Comparativo de entidades extraídas

|  |  |
| --- | --- |
| Ltasks | Polícia Federal, DNPM, Rogério Castro |
| Yahoo | pedras preciosas, bens, castro |
| Nltk | Castro, Rogério Castro, após uma denúncia para, cargo, responsável pela agência, DNPM, Federal informou que foi instaurado um inquérito, usurpação de bens |
| Zemanta | Brazil, South America, States, People, Business, Lake Chapala, Oaxaca, LinkedIn |
| Nosso Extrator | Policia Federal, Rogerio Castro, Minas Gerais, DNPM, Ibama |

Para comparar as ferramentas de mercado para extração de entidades nomeadas com a ferramenta desenvolvida preparamos um corpus com 1000 sentenças anotadas com uma entidade conhecida. Este corpus foi extraído da base de dados semântica do portal G1.

O tamanho do corpus foi limitado em 1000 sentenças devido ao limite de requisições de algumas das ferramentas de mercado.

As sentenças foram organizadas em tuplas onde o primeiro valor era a própria sentença e o segundo valor era o nome da entidade nomeada como mostra o exemplo abaixo:

*(“Entre as mais de 300 atrações estão shows com os cantores Lulu Santos,* ***Marcelo D2****, Preta Gil e Cidade Negra.” , “Marcelo D2”)*

Desenvolvemos um teste onde para cada sentença, nós realizávamos a requisição aos serviços para a extração das entidades e comparávamos a lista de entidades obtidas com a entidade anotada para a sentença. Caso a entidade anotada fosse encontrada na lista de entidades extraídas por aquele serviço, marcávamos um acerto para o serviço.

Com isso chegamos a tabela abaixo que aponta o aproveitamento em percentual de acertos por serviço:

Tabela 7 – Comparativo entre extratores

|  |  |
| --- | --- |
| Ltasks | 63% |
| Zemanta | 32% |
| Nltk | 42% |
| Yahoo | 33% |
| Nosso Extrator | 77% |

Como podemos observar na tabela 7, nossa ferramenta de extração de entidades apresentou uma assertividade maior que as ferramentas de mercado e por isso optamos por seu uso para extrair a lista das entidades prováveis que foram encontradas no texto.

## Seleção de candidatos

A fase de seleção de candidatos consiste em realizar consultas no servidor de busca interno **solr** para identificar um conjunto de matérias que possam ser aproveitadas para a recomendação. Esta consulta precisa levar em consideração alguns critérios que permitam diminuir os ‘ruídos’ durante a fase de recomendação bem como reduzir o espaço de amostra da base de dados.

O projeto **Google News,** que pode ser estudado no trabalho de Abhinandan e Mayur [16], faz uso de uma fase de separação de artigos candidatos que utiliza como características para sua seleção informações como: a edição da notícia, o idioma, a data de publicação e seções selecionadas pelo usuário.

Nosso mecanismo de seleção de candidatos é dividido em três etapas bem definidas como: Definição de critérios de ordenação e filtragem, combinação de **features** e recuperação das candidatas.

Na primeira etapa, vamos adotar alguns critérios semelhantes ao modelo explicado no projeto do **Google News**.

Na etapa de combinação de features, vamos fazer uso das **features** extraídas pelo extrator de **features** combinando-as para melhorar os resultados retornados na seleção das matérias candidatas a recomendação.

Na etapa de recuperação, iremos montar consultas escritas na sintaxe **lucene** para executar buscas sobre nosso servidor **solr**.

### Definição dos critérios de filtragem e ordenação

O primeiro critério para reocupação das candidatas se dá pela **data de publicação**. Quanto mais próxima for a publicação da matéria relacionada à data de criação do novo texto, acreditamos que maior será a chance dos textos falarem do mesmo assunto. Este é um critério de ordenação.

O segundo critério para seleção é fornecido pelo próprio editor. A **editoria** da matéria permite-nos filtrar candidatos da mesma editoria aproximando com isso a relação de conteúdo entre o texto e as matérias candidatas. Este é um critério de filtragem.

O último critério e o mais importante é o **score** da matéria relacionada que é mais um critério de ordenação. O **score** é dado de acordo com a função de similaridade existente no **solr** que faz uso da biblioteca **lucene** [22]. O **score** possui a seguinte fórmula:

Função 4 – Cálculo de score do lucene

Onde:

* ***tf(t in d)*** ou **Term** **Frequency**, define o número de vezes que um termo t aparece no documento d. Isto implica em documentos com o maior número de termos encontrados recebem um maior score.
* ***idf(t)*** ou **Inverse** **Document** **Frequency**. Significa o inverso da frequência nos documentos. Frequência nos documentos é o número de documentos em que um termo t aparece. Ou seja quanto mais raro o termo maior é o valor do **idf** e melhor é o score.
* ***coord(q,d)*** é um fator baseado no número de termos da consulta que foram encontrados em um determinado documento. Tipicamente, um documento que possui mais termos da consulta em seu corpo, tende a ter um melhor score.
* ***queryNorm(q)*** é um fator de normalização que visa permitir a comparação de **scores** entre consultas diferentes.
* ***t.getBoost()*** é o peso dado a um termo t para uma determinada consulta. Ou seja, em tempo de execução de consulta é possível definir pesos para termos específicos da consulta aumentando assim o **score** de documentos que possuem estes termos.
* ***norm(t,d)*** é o produto de um conjunto de pesos que são atribuídos em tempo de indexação dos documentos.

Em linhas gerais, esse **score** visa ordenar as matérias de acordo com a similaridade existente entre os documentos indexados e as consultas elaboradas na etapa de combinação de **features**.

### Combinação de features

Na etapa de combinação**,** as **features** extraídas do corpo da matéria são combinadas para alcançar o melhor resultado durante a recuperação das matérias candidatas.

Nesta etapa, realizamos todas as combinações possíveis com as features extraídas para descobrir quais features combinadas obtém a melhor assertividade na etapa de recomendação.

No capítulo de experimento, vamos detalhar o processo de combinação das features e mostrar os resultados de recomendações com as features combinadas.

Como exemplo, para a matéria “**Reabertura do Zoológico de Goiânia é adiada para 2012**” que pode ser vista na figura 11, quando combinamos as features **unigrams** mais frequentes e entidades nomeadas, tivemos as seguintes informações segundo a tabela 8:

Tabela 8 – Exemplo de combinação de features

|  |  |
| --- | --- |
| Termos mais Frequentes | parque, animais, adiada, estão, mês, prefeitura, reabertura, reinauguração |
| Entidades | Reabertura do Zoológico, Zoológico de Goiânia, Recursos Naturais Renováveis, Ministério Publico Federal, Cristiane Borges Miguel, Instituto Brasileiro, Zoológico, Goiânia, Ibama, MPF, Amma |



Figura 11 – Exemplo de matéria publicada

### Recuperação das candidatas

A etapa de recuperação das candidatas consiste em construir consultas em sintaxe **lucene** [22], utilizando os **n-grams** obtidos após com a combinação das features.

Uma consulta é montada para cada combinação de **feature** e é submetidas ao **solr** através de sua **API** de consulta.

As consultas são realizadas com utilização dos operadores lógicos **OR** e **AND** que permitem combinar os **n-grams** dentro da consulta.

Após a tradução das features em sintaxe **lucene**, o incremento dos filtros das editorias “Goiás” e “Brasil” e das ordenações por “**score**” e “data de publicação”, temos a seguinte consulta:

**(((parque) OR (Ibama) OR (animais) OR (prefeitura) OR (Reabertura do Zoológico) OR (Zoológico de Goiânia) OR (Recursos AND Naturais AND Renováveis) OR (Ministério AND Publico AND Federal) OR (Cristiane AND Borges AND Miguel) OR (Instituto AND Brasileiro) OR (Previsão) OR (Zoológico) OR (Goiânia) OR (MPF) OR (Amma)) AND (editoria\_principal\_s:"Goiás" OR editoria\_principal\_s:"Brasil" ) ) isIssued:true type:texto publisher:G1 sort=score desc, issued, rows=50**

Esta consulta retorna o conjunto das 50 primeiras matérias, pois limitamos os resultados retornados no parâmetro **rows**.

## Recomendação

A fase de recomendação consisti em sugerir ao editor as matéria que apresentam a melhor similaridade com a matéria que está sendo criada. Para isso, após a seleção de matérias candidatas, aplicamos o **vector space model** para gerar um fator de similaridade entre as matérias.

Após a aferição da medida de similaridade, ela é multiplicada ao **score** dado a matéria pelo servidor de busca e a listagem de candidatas é então reordenada.

Em seguida a reordenação da listagem, são selecionadas as 5 matérias com o **score** mais elevado e com isso o sistema **GRNews** encerra seu ciclo de recomendação.

O Processo de auferir a similaridade entre as matérias candidatas e a matéria que esta sendo construídas é contestável para nosso sistemas sobre dois aspectos:

### Primeiro aspecto: Conceito de “está relacionada”

Segundo Yuanhua et al. [1], a medida de similaridade sozinha não é suficiente para capturar o relacionamento entre duas matérias. Um exemplo dado por eles é o caso de matérias com conteúdos duplicados onde a medida de similaridade é alta porém, o grau de relacionamento não existe por se tratarem do mesmo assunto. Ainda no trabalho de Yuanhua et al. [1], os autores definem em seu trabalho 3 critérios afim de modelar uma função capaz de entregar uma melhor relação entre as matérias na fase de recomendação, são eles: Relevância e novidade, Clareza de conexão e Suavidade de transição.

Continuando neste tema, Fekade Getahun et al. [9] entendem que o fator de relacionamento pode ser explicado por 3 ideias simples: **Inclusão**, quando uma matéria está inserida em outra. **Intercessão**, quando duas matérias apresentam alguns conceitos em comum. E **Opostas** quando não apresentam conceitos em comum. Além destes trabalhos, é possível encontrar outros que tratam do tema de relacionamento de conteúdo mostrando-nos a dificuldade que o tema implica.

### Segundo aspecto: Desempenho da comparação

Outra questão importante é o desempenho do sistema levando se em consideração o volume de comparações que precisam ser executadas para assegurar uma listagem final de matéria recomendadas. Durante a fase de seleção de candidatas, selecionamos 50 matérias. Desta forma para cada recomendação no sistema **GRNews**, estamos realizando 50 comparações de similaridade usando a função de similaridade baseada na distância dos cossenos. Não foi observado durante este trabalho uma possibilidade de melhoria no algoritmo ou na abordagem utilizada.

## Comentários do projeto

As principais tecnologias descritas abaixo, foram utilizadas para projetar, desenvolver e suportar a ferramenta pois são tecnologias de código aberto, gratuitas, do conhecimento técnico do autor e por se adequarem as práticas utilizadas no contexto da empresa de onde será aplicado o experimento.

***PYTHON*:** Linguagem de programação criada no inicio dos anos 90 por Guido van Rossum. Considerada uma linguagem de alto nível, permite a escrita de código conciso, enxuto e poderoso. Possui uma estrutura de código simples de modo que a maioria dos programadores conseguem facilmente ler e entender um programa escrito com ela.

Python vem com um grande conjunto de bibliotecas nativas que permitem desde o desenvolvimento com funções matemáticas até o uso de **parsing** de **XML**.

Em seu modo iterativo, é possível criar, executar e avaliar o resultado de funções. Python é uma linguagem que suporta diversos paradigmas de programação desde a orientação a objetos até a programação funcional.

Python é uma linguagem multiplataforma e tem licença aberta para uso.

***NLTK***: É uma caixa de ferramentas escrita em python que apresenta um conjunto de bibliotecas e funções para o processamento de linguagem natural e para a análise de textos. Foi originalmente criado com o propósito de ensinar mas com o passar dos anos, tem sido adotado pela indústria e por pesquisadores.

O **toolkit** possui alguns corpora onde podemos realizar treinamentos de classificadores e **POS-tagger.**

Durante nosso trabalho faremos uso do corpus **MAC-MORPHO** encontrado no **toolkit**.

***SOLR*:** É uma plataforma de busca corporativa de código aberto oriunda do projeto Apache Lucene. Suas características principais incluem pesquisa de texto, pesquisa facetada, **clustering**, integração com banco de dados e a capacidade de indexação de documentos ricos (por exemplo, Word, PDF). Solr é altamente escalável, fornecendo pesquisa distribuída e replicação dos índices.

Solr é escrito em Java e funciona como serviço de busca que roda dentro de um **container** de **servlet** como o **Tomcat**. **Solr** utiliza a biblioteca **Lucene** de busca em seu núcleo para indexação de textos e a realização de pesquisas.

Solr tem interfaces de consulta em REST/XML e JSON que a tornam fácil de usar a partir de praticamente qualquer linguagem de programação.

***GIT*:** é um sistema de controle de versão de código aberto e gratuito que permite que se trabalhe com diversas versões de arquivos organizados em um diretório e localizados local ou remotamente, mantendo-se suas versões antigas e os logs de quem e quando manipulou os arquivos. É especialmente útil para se controlar versões de um software durante seu desenvolvimento, ou para composição colaborativa de um documento.

***MYSQL*:** é um sistema de gerenciamento de banco de dados (**SGBD**) que utiliza a linguagem **SQL** como interface. Desenvolvido como projeto de código aberto.

# Experimento

Para avaliar o **GRNews** e testar a acurácia do processo de recomendação de matérias relacionadas, desenvolvemos um conjunto de experimentos que certificam os módulos do sistema sobre um corpus de matérias já relacionadas manualmente.

Primeiro vamos detalhar como esse **dataset** foi obtido, depois vamos definir uma heurística para avaliação e em seguida vamos executar uma função especialmente desenvolvida para combinar as diversas features obtidas no capítulo 3 de modo a descobrir que combinação possui a maior assertividade sobre o corpus.

Ao final vamos apresentar os resultados obtidos e vamos discutir as possibilidade de melhoria sobre o sistema **GRNews**.

## O corpus

O corpus utilizado pertence ao portal de noticias **G1** da **Globo.com** e possui um total de 492 mil matérias no período de fevereiro de 2010 a outubro de 2011 divididas em 162 editorias.

Deste corpus, 72 mil matérias possuem o componente de matéria relacionada, o “**saibamais**”. Este componente matem os links das matérias que foram relacionadas pelo editor.

Para diminuir o tempo de execução dos algoritmos reduzimos o corpus para um total 1440 matérias já relacionadas o que representa a 2% do total de matérias relacionadas manualmente. Também reduzimos as representação das editorias ficando apenas com as 15 principais editorias de acordo com o seu número de matérias publicadas.

Decidimos distribuir de forma igualitária as matérias em suas editoria de modo a promover uma visão das editorias com maior percentual de acertos, assim temos a seguinte distribuição:

Tabela 8 – Representação do corpus

|  |  |  |
| --- | --- | --- |
| 1 | Brasil | 96 |
| 2 | São Paulo | 96 |
| 3 | Rio de Janeiro | 96 |
| 4 | Minas Gerais | 96 |
| 5 | Economia | 96 |
| 6 | Política | 96 |
| 7 | Mundo | 96 |
| 8 | Espírito Santo | 96 |
| 9 | Pop & Arte | 96 |
| 10 | Auto Esporte | 96 |
| 11 | Concursos e Emprego | 96 |
| 12 | Ciência e Saúde | 96 |
| 13 | Música | 96 |
| 14 | Mercados | 96 |
| 15 | Tecnologia e Games | 96 |
|  | TOTAL | 1440 |

Foram necessárias etapas de limpeza do corpus para garantir que todos os links de matérias apontavam para matérias existentes na nossa base de dados. Isso se fez necessário porque o cadastro de matérias relacionadas é aberto e permite a ligação à matérias externas. Assim sendo, matérias que possuíam pelo menos um link externo, foram descartadas durante a preparação do corpus.

Ao final desta etapa de limpeza, temos então um corpus de matérias relacionadas manualmente com links válidos e que corresponde a 2% do corpus total de matérias relacionadas manualmente.

Durante a etapa de limpeza, as matérias foram indexadas no servidor de busca **solr** que dá suporte ao sistema **GRNews**.

## Critério para avaliação do sistema

Para avaliar o sistema foi necessária a construção de um módulo de combinação das **features** extraídas pelo extrator de **features** como dito no capítulo anterior.

Para cada combinação de **features,** o módulo percorre o corpus separado com matérias previamente relacionadas e realiza para cada matéria uma recomendação de n matérias onde n é o número de matérias que serão recomendadas. Após a recomendação o sistema compara as matérias que foram relacionadas manualmente com as que foram recomendadas pelo **GRNews** e em seguida extrai a interseção. Se a interseção não for nula, consideramos que a recomendação foi válida.

## Resultados obtidos

Para avaliar a hipótese de melhoria da recomendação com **features** extraídas do texto, estipulamos um **baseline** para o projeto **GRNews**.

Considerando apenas os **unigrams** mais relevantes do texto, submetemos o sistema **GRNews** ao processamento de um conjunto de recomendações contra o corpus de teste.

Com o resultado das recomendações, conseguimos estabelecer um percentual de aproximadamente **38%** de acurácia, como pode ser observado na tabela 8.

A partir deste **baseline**, iniciamos um conjunto de tentativas onde variamos as **features**, o número de matérias recomendadas e a utilização da medida de similaridade para tentar aumentar a acurácia do **GRNews** nas recomendações.

A seguir vamos apresentar os resultados obtidos com cada rodada de recomendações afim de estudar os dados e eleger as melhores combinações de features para o **GRNews**.

A primeira tentativa foi descobrir a melhor combinação de critério sobre todo o corpus de teste utilizando como base um total de 5 recomendações e ainda dispensando o fator de similaridade das matérias uma vez que ele poderia não ser determinante para encontrarmos a melhor relação. Com isso obtivemos a tabela abaixo:

Tabela 9 – Acurácia por feature com 5 recomendações sem similaridade

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Total Geral | | | | | | | | | | |
| **ubtce** | **0.471160528** |  | uthc | 0.443363447 |  | | hce | | 0.408617095 | |
| ubthce | 0.470465601 |  | ubc | 0.44266852 |  | | ce | | 0.405837387 | |
| uthce | 0.46838082 |  | ubhc | 0.44266852 |  | | bt | | 0.40514246 | |
| utce | 0.466990966 |  | tce | 0.441973593 |  | | bth | | 0.403752606 | |
| ubte | 0.46143155 |  | thce | 0.441278666 |  | | he | | 0.397498263 | |
| ubthe | 0.46143155 |  | bte | 0.439888812 |  | | e | | 0.395413482 | |
| ubtc | 0.460736623 |  | bthe | 0.439193885 |  | | uhc | | 0.394718555 | |
| ubthc | 0.460736623 |  | ut | 0.435719249 |  | | uc | | 0.389854065 | |
| ubhce | 0.457956915 |  | uth | 0.434329395 |  | | uh | | 0.389159138 | |
| ubce | 0.45656706 |  | bhce | 0.432244614 |  | | **u** | | **0.384294649** | |
| uthe | 0.45656706 |  | uhe | 0.432244614 |  | | thc | | 0.382209868 | |
| ute | 0.455872133 |  | ub | 0.431549687 |  | | tc | | 0.380125087 | |
| ubt | 0.451702571 |  | ue | 0.431549687 | |  | | bc | | 0.375955525 | |
| ubth | 0.451702571 |  | ubh | 0.43085476 |  | | bhc | | 0.375955525 | |
| btce | 0.448922863 |  | bce | 0.429464906 |  | | bh | | 0.357887422 | |
| ubhe | 0.448922863 |  | te | 0.427380125 |  | | th | | 0.357887422 | |
| bthce | 0.448227936 |  | the | 0.426685198 |  | | t | | 0.357192495 | |
| ube | 0.446838082 |  | bhe | 0.422515636 |  | | b | | 0.355107714 | |
| uhce | 0.446838082 |  | be | 0.419735928 |  | | hc | | 0.161223072 | |
| uce | 0.445448228 |  | btc | 0.419735928 |  | | c | | 0.136900625 | |
| utc | 0.444753301 |  | bthc | 0.419735928 |  | | h | | 0.03752606 | |

**u – unigrams, b – bigrams, t – trigrams, h – html tags, c – captions, e – entidades.**

Para entender as combinações na tabela acima e na demais, é necessária a interpretação dos rótulos utilizando a legenda mostrada no rodapé da tabela 8.

Neste primeiro resultado observamos que utilizando as **features** combinadas **ubtce**, conseguimos um percentual de acurácia de aproximadamente 47%, o que representa uma melhoria de aproximadamente 24% sobre o **baseline**.

Se olharmos as distribuição por editoria detalhadas no apêndice A, observamos que *São Paulo* e *Rio de Janeiro* obtiveram as melhores taxas de assertividade com aproximadamente 78% e 82% respectivamente. Levando-se em consideração a combinação mínima para estas categorias temos para *São Paulo* a sequência **ubt** enquanto que para o *Rio de Janeiro* temos a sequência **ubce**.

Da mesma forma, temos também as editorias com o pior percentual de acurácia são elas: *Mercados* com aproximadamente 19%, *Economia* com 28% e *Minas Gerais* com aproximadamente 27%. Nestas categorias as melhores sequências de features foram: **ubt**, **utce** e **ube**.

Uma primeira conclusão possível sobre estas informações está no fato de em editorias mais organizadas, ou seja, com menos conteúdo diversificado, a recomendação é mais assertiva. O contexto de *Economia* é mais abrangente que os contextos de *Rio de Janeiro* e *São Paulo*. *Economia* apresenta matérias do brasil e do mundo enquanto as editorias *Rio de Janeiro* e *São Paulo,* são focadas no noticiário local o que torna o contexto muito próximo.

Uma outra hipótese é a qualidade da equipe de jornalistas por editoria. Como *Rio de Janeiro* e *São Paulo* têm mais visibilidade, os jornalistas dispõem de uma atenção maior no relacionamento do conteúdo. Esta hipótese explicaria o fato de *Economia*, *Mercados e* *Minas Gerais* apresentarem uma acurácia baixa.

A segunda tentativa, foi em razão da dúvida sobre o fator de similaridade. Durante este teste, apenas habilitamos o fator de similaridade para ajustar o **score** das matérias com maior similaridade textual. Com isso pudemos observar quanto o fator de similaridade entre as matérias influencia no nosso algoritmo. O novo **score** é dado pela multiplicação simples entre o **score** anterior e o fator de similaridade entre as duas matérias dado pela distância do cossenos.

Tabela 10 – Acurácia por feature com 5 recomendações com similaridade

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Total Geral | | | | | | | |
| **ubthce** | **0.49409312** |  | bthce | 0.478109798 |  | u | 0.457261987 |
| ubtce | 0.493398193 |  | uhce | 0.476719944 |  | uh | 0.457261987 |
| utce | 0.492008339 |  | uthc | 0.476025017 |  | btc | 0.451007644 |
| uthce | 0.492008339 |  | bthe | 0.47533009 |  | bthc | 0.451007644 |
| ubce | 0.491313412 |  | uce | 0.47533009 |  | bth | 0.44266852 |
| ubhce | 0.491313412 |  | utc | 0.47533009 |  | hce | 0.44266852 |
| ubthe | 0.489228631 |  | bte | 0.474635163 |  | bt | 0.441973593 |
| ubte | 0.488533704 |  | thce | 0.473245309 |  | ce | 0.441278666 |
| ubhe | 0.48714385 |  | bhce | 0.472550382 |  | he | 0.438498958 |
| ubtc | 0.48714385 |  | tce | 0.471855455 |  | e | 0.436414177 |
| ube | 0.486448923 |  | bce | 0.471160528 |  | tc | 0.426685198 |
| ubc | 0.485753996 |  | the | 0.471160528 |  | thc | 0.424600417 |
| ubthc | 0.485059069 |  | te | 0.470465601 |  | bhc | 0.41695622 |
| ubhc | 0.484364142 |  | ut | 0.469770674 |  | t | 0.41695622 |
| ute | 0.484364142 |  | uth | 0.469075747 |  | th | 0.415566366 |
| uthe | 0.484364142 |  | bhe | 0.467685893 |  | bc | 0.414871438 |
| ub | 0.479499653 |  | uhe | 0.467685893 |  | bh | 0.403057679 |
| ubt | 0.479499653 |  | be | 0.466990966 |  | b | 0.400972898 |
| ubh | 0.478804726 |  | uc | 0.466296039 |  | hc | 0.134120917 |
| ubth | 0.478804726 |  | ue | 0.466296039 |  | c | 0.113273106 |
| btce | 0.478109798 |  | uhc | 0.465601112 |  | h | 0.031966644 |

Como podemos observar, o uso do fator de similaridade entre os textos das matérias implica em uma melhoria de aproximadamente 29% sobre o **baseline** estabelecido.

Olhando para a tabela geral, o que pode ser destacado é a mudança na sequência de features combinadas que obteve a melhor acurácia. Saímos da sequência **ubtce** para a sequência **ubthce.** Como fator de similaridade, as informações extraídas de **tags html** informativas ganharam mais relevância.

Observando a distribuição por editoria, podemos destacar uma melhora percentual em quase todas com exceção das editorias: *Brasil* que teve uma ligeira queda e *Ciência e Saúde* que não variou com o fator de similaridade aplicado. Ainda observando as editorias, assim como na tabela geral, quase todas as combinações vencedoras foram mudadas em relação ao teste executado sem o fator de similaridade.

Até este ponto estávamos realizando apenas 5 recomendações por matéria. Nas tabelas a seguir vamos querer verificar qual o percentual de melhora que será obtido se aumentarmos o número de itens recomendados de 5 para 10.

Executaremos então os dois procedimentos anteriores porém alterando o número de recomendações para 10.

Tabela 11 – Acurácia por feature com 10 recomendações sem similaridade

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Total Geral | | | | | | | |
| **uthce** | **0.53856845** |  | uthc | 0.520500347 |  | bth | 0.483669215 |
| ubthce | 0.537873523 |  | uce | 0.51980542 |  | uhc | 0.483669215 |
| ubtce | 0.537178596 |  | utc | 0.51980542 |  | bt | 0.482974288 |
| utce | 0.537178596 |  | btce | 0.517720639 |  | uc | 0.481584434 |
| ubthc | 0.534398888 |  | uhe | 0.516330785 |  | uh | 0.47533009 |
| ubc | 0.533703961 |  | bthce | 0.515635858 |  | hce | 0.474635163 |
| ubtc | 0.533703961 |  | ue | 0.514246004 |  | ce | 0.473940236 |
| ubhc | 0.533009034 |  | ut | 0.51285615 |  | **u** | **0.470465601** |
| ubhce | 0.532314107 |  | uth | 0.51285615 |  | he | 0.469770674 |
| ubt | 0.532314107 |  | bte | 0.510771369 |  | e | 0.469075747 |
| ubth | 0.532314107 |  | thce | 0.509381515 |  | bc | 0.447533009 |
| ubthe | 0.532314107 |  | bthe | 0.508686588 |  | bhc | 0.446143155 |
| ubte | 0.53161918 |  | tce | 0.508686588 |  | thc | 0.444058374 |
| ubh | 0.530924253 |  | bhce | 0.499652536 |  | tc | 0.443363447 |
| ub | 0.529534399 |  | the | 0.499652536 |  | bh | 0.428075052 |
| ubce | 0.529534399 |  | te | 0.498957609 |  | b | 0.426685198 |
| uthe | 0.529534399 |  | bce | 0.498262682 |  | th | 0.425295344 |
| ute | 0.528839472 |  | bhe | 0.493398193 |  | t | 0.42390549 |
| ubhe | 0.526059764 |  | btc | 0.492703266 |  | hc | 0.143849896 |
| ube | 0.523280056 |  | bthc | 0.492008339 |  | c | 0.122307158 |
| uhce | 0.521890202 |  | be | 0.491313412 |  | h | 0.029186935 |

Com 10 recomendações a acurácia alcança aproximadamente 54%. Porém, podemos perceber que o **baseline** para 10 recomendações também é alto, cerca de 47%. Desta forma, é possível notar que o aumento percentual com o uso de outras **features** é de 15% o que é um ganho menor em relação ao mesmo teste com 5 recomendações.

Isto significa que com 10 recomendações é mais fácil acertar. Até mesmo fazendo uso de features simples como é o caso do **baseline**.

Mas de qualquer forma para efeito de recomendação, é correto dizer que conseguimos um grau maior de precisão em matérias relacionadas quando aumentamos o número de matérias recomendas.

Quando observamos por exemplo a recomendação por editoria, percebemos que a editoria *Rio de Janeiro* alcança uma acurácia de aproximadamente 91%.

O experimento seguinte apenas comprova o que já foi visto anteriormente na recomendação com utilização do fator de similaridade. Como pode ser visto na tabela 12, o ganho percentual sobre o **baseline** aumenta para aproximadamente 21%.

Tabela 12 – Acurácia por feature com 10 recomendações com similaridade

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Total Geral | | | | | | | |
| **ubtce** | **0.566365532** |  | ubh | 0.548992356 |  | bthc | 0.520500347 |
| ubthce | 0.565670605 |  | uhce | 0.548992356 |  | btc | 0.519110493 |
| Ubte | 0.564975678 |  | ub | 0.546907575 |  | u | 0.515635858 |
| ubthe | 0.564280751 |  | utc | 0.546212648 |  | uh | 0.513551077 |
| uthce | 0.562195969 |  | uthc | 0.544127867 |  | bth | 0.512161223 |
| Utce | 0.561501042 |  | bte | 0.54343294 |  | bt | 0.510076442 |
| ubce | 0.560806115 |  | bthe | 0.542738013 |  | hce | 0.501737318 |
| ubthc | 0.560806115 |  | ue | 0.542738013 |  | ce | 0.500347464 |
| Ube | 0.560111188 |  | uhe | 0.541348158 |  | he | 0.500347464 |
| ubtc | 0.559416261 |  | bhce | 0.53856845 |  | e | 0.498262682 |
| ubhce | 0.558721334 |  | bce | 0.537873523 |  | tc | 0.489923558 |
| ubhe | 0.558026407 |  | thce | 0.537873523 |  | thc | 0.487838777 |
| ute | 0.558026407 |  | tce | 0.536483669 |  | bhc | 0.472550382 |
| uthe | 0.558026407 |  | the | 0.533703961 |  | t | 0.472550382 |
| ubth | 0.556636553 |  | be | 0.533009034 |  | th | 0.471855455 |
| ubhc | 0.554551772 |  | bhe | 0.533009034 |  | bc | 0.469770674 |
| ubt | 0.554551772 |  | te | 0.533009034 |  | bh | 0.460736623 |
| ubc | 0.553161918 |  | ut | 0.533009034 |  | b | 0.459346769 |
| uce | 0.55038221 |  | uth | 0.53161918 |  | hc | 0.157053509 |
| btce | 0.549687283 |  | uc | 0.526059764 |  | c | 0.131341209 |
| bthce | 0.548992356 |  | uhc | 0.523974983 |  | h | 0.035441279 |

# Conclusões e trabalhos futuros

Com este trabalho abordamos o problema da recomendação de segundo nível que é um problema similar a recomendação baseada em filtragem de conteúdo porém, com um fator maior de subjetividade pois trata da recomendação sem a utilização do perfil de consumo do usuário.

Sistemas de recomendação já estão consolidados como ferramentas importantes que auxiliam os usuário de internet em diversos contextos. Seja em lojas virtuais ou em indicações de filmes em cartaz, os sistemas de recomendação tem facilitado enormemente a tomada de decisão dos usuários na internet.

Partindo destas boas características, propomos um sistema de recomendação que auxiliasse os jornalista na produção de conteúdo relacionado às matérias publicadas para os usuários do site na web.

Como esta recomendação é feita para o jornalista, não utilizamos os dados de perfil de usuário que são típicos em sistemas de recomendação baseados em filtragem de conteúdo. Por este motivo, nominamos nosso modelo para modelo de sistema de recomendação de segundo nível.

O **GRNews** foi construído com módulos acopláveis como um **framework** onde podemos substituir partes do sistema por componentes que forneçam os melhores resultados. Como exemplo, as **features** apresentadas no capitulo 3 podem ser adicionadas e ou removidas do sistema sem afetar o funcionamento do sistema como um todo.

A implementação do nosso sistema, o **GRNews**, é independente de plataforma de publicação de conteúdo podendo ser acoplada a qualquer base de dados de matérias.

Nosso primeiro piloto foi com a base de dados de matérias do portal **G1**. Porém, temos a previsão de aplicar também o modelo sobre a base de dados de matérias do portal **Globoesporte**.**com**.

## Principais contribuições

A principal contribuição desta pesquisa refere-se ao modelo de recomendação de segundo nível. Pois trata-se da utilização de técnicas de recomendação para suporte a tomada de decisão da equipe de jornalistas.

Como não encontramos na literatura um trabalho similar, entendemos que esta abordagem pode ser utilizada para auxiliar outras equipes de jornalistas na publicação de conteúdos em outros portais de internet.

Como decorrência da pesquisa visualizamos outra contribuição que é o extrator de entidades nomeadas. Embora o nosso extrator não supere outros extratores existentes na literatura, ele pode ser melhorado e utilizado em outros trabalhos de processamento de texto.

## Limitações do sistema

Algumas limitações podem ser observadas durante o desenvolvimento da pesquisa. Podemos começar com as **features** que foram utilizadas. Poderíamos aumentar o número de **features** utilizadas levando em consideração por exemplo os **links** existentes nas matérias. Observamos alguns trabalhos na literatura que utilizaram os documentos referenciados por **links** no corpo do texto como possíveis fontes de informações.

Outra limitação está no melhoramento da informação extraída. Alguns trabalhos na literatura utilizaram uma abordagem de expansão de conceitos, ou seja, dado um conceito extraído o sistema utilizava um serviço de suporte que permitia identificar informações relacionadas a este conceito. Poderíamos utilizar a **Wikipédia** para extrair mais informações relacionadas as **features** extraídas.

Com relação a ordenação das recomendações, poderíamos utilizar um conceito de **PageRank** para identificar aquelas matérias mais populares no momento da recomendação ao invés de utilizar apenas a medida de similaridade textual.

## Trabalhos futuros

Como uma primeira iniciativa de trabalho futuro, pretendemos criar sub grupos das notícias existentes nas editorias que tiveram uma acurácia baixa. O objetivo destes sub grupos é identificar se com grupos mais homogêneos o sistema **GRNews** consegue elevar o percentual de acurácia. Esta hipótese foi levantada devido a observação das editorias “*Rio de Janeiro”* e “*São Paulo*” que obtiveram os melhores índices de assertividade.

Outra linha interessante é aplicação do algoritmo de **PageRank** para estabelecer um critério de popularidade ao **score** das matérias relacionadas. A ideia é utilizar os links existentes entre as matérias já relacionadas para estabelecer os pesos necessário para aplicação do algoritmo.

Por fim, faz sentido um estudo mais aprofundado sobre os aspectos de extração de entidades nomeadas para desenvolver um novo classificador que obtenha um melhor percentual de acurácia.

# Bibliografia

1. Yuanhua Lv, Taesup Moon, Pranam Kolari, Zhaohui Zheng, Xuanhui Wang, Yi Chang – (2011) **Learning to model relatedness for news recommendation**, Yahoo! Labs, Sunnyvale, CA, USA.
2. World Wide Web Size – (2011) **Daily Estimated Size**, disponivel em: <http://www.worldwidewebsize.com/>. Acesso em 08 Jul 2011.
3. Miniwatts Marketing Group – (2011) **Internet World Stats**, disponivel em: <http://www.internetworldstats.com/stats.htm>. Acesso em 19 Jan 2011.
4. James Davidson, Benjamin Liebald, Junning Liu – (2010) **The YouTube Video Recommendation System**, Google Inc.
5. Francesco Ricci, Lior Rokach, Bracha Shapira – (2010) **Introduction to Recommender Systems Handbook**.
6. Bracha Shapira, Lior Rokach – (2010) **Recommender Systems and Search Engines – Two sides of the same Coin!?**, Department of Information Systems Engineering, Ben-Gurion University.
7. Jacob Perkins – (2010) **Python Text Processing with NLTK 2.0 Cookbook**.
8. Bruno Magalhães Nogueira – (2009) **Avaliação de métodos não-supervisionados de seleção de atributos para Mineração de Textos**, Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, ICMC-USP.
9. Fekade Getahun, Joe Tekli, Chbeir Richard, Marco Viviani, Kokou Yetongnon – (2009) **Relating RSS News/Items**, Laboratoire Electronique, Informatique et Image (LE2I), UMR-CNRS Université de BourgogneSciences et Techniques
10. Yin Yang, Panagiotis Ipeirotis, Wisam Dakka, Dimitris Papadias, Nilesh Bansal, Nick Koudas – (2009) **Query by Document,** Computer Science University of Toronto, koudas@cs.toronto.edu.
11. Arni Darliani Asy’arie, Adi Wahyu Pribadi – (2009) **Automatic News Articles Classification in Indonesian Language by Using Naive Bayes Classifier Method**, Universitas Islam Negeri Syarif Hidayatullah Jakarta, Universitas Pancasila Jl. Srengseng Sawah, Jagakarsa Jakarta.
12. Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze – (2008) **Introduction to Information Retrieval**, Cambridge University Press.
13. Luiz Claudio Gomes Maia – (2008) **Uso de Sintagmas Nominais na Classificação Automática de Documentos Eletrônicos**, Universidade Federal de Minas Gerais, UFMG.
14. Alexander E. Richman, Department of Defense – (2008) **Mining Wiki Resources for Multilingual Named Entity Recognition,** Departamento de Defesa Americano.
15. Takashi Yoneya, Hiroshi Mamitsuka – (2007) **PURE: A Pubmed Article Rcommendation System Based on Content-Based Filtering,** Bioinformatics Center, Kyoto University, Gokasho Uji, Japan and Discovery Research Laboratories, Kirin Pharma, Miyahara, Takasaki, Japan
16. Abhinandan Das, Mayur Datar – (2007) **Google News Personalization: Scalable Online Collaborative Filtering**, Google Inc
17. Dipanjan Das, Andre F.T. Martins – (2007) **A Survey on Automatic Text Summarization,** Language Technologies Institute Carnegie Mellon University.
18. Hiroyuki Toda, Patrick Schone – (2005) **A Clustering Method for News Articles Retrieval System**, NTT Cyber Solutions Laboratories, NTT Corporation
19. Greg Linden, Brent Smith, Jeremy York – (2003) **Amazon.com Recommendations Item-to-Item Collaborative Filtering**, IEEE Internet Computing Magazine.
20. Ying Huang – (2001) **An Intelligent adaptative news filtering system**, University of Science & Technology of China.
21. Chin-Yew Lin, Eduard Hovy – (1997) **Identifying Topics by Position,** Information Sciences Institute of the University of Southern California
22. Apache Lucene – (2011) **Lucene Documentation**, disponivel em: <http://wiki.apache.org/lucene-java> . Acesso em 20 Set 2011.
23. Min-Ling Zhang, Jose M. Penã and Victor Robles – (2009) **Feature Selection for Multi-Label Naive Bayes Classification**, College of Computer and Infor- mation Engineering, Hohai University, Nanjing 210098, China.
24. Fábio Natanael Kepler, Marcelo Finger – (2006) **Comparing Two Markov Methods for Part-of-Speech Tagging of Portuguese**. Universidade de São Paulo.

# APÊNDICE A

Este apêndice apresenta todos os dados que foram coletados durante a etapa de experimentação. Os dados foram tabulados e organizados por editorias.

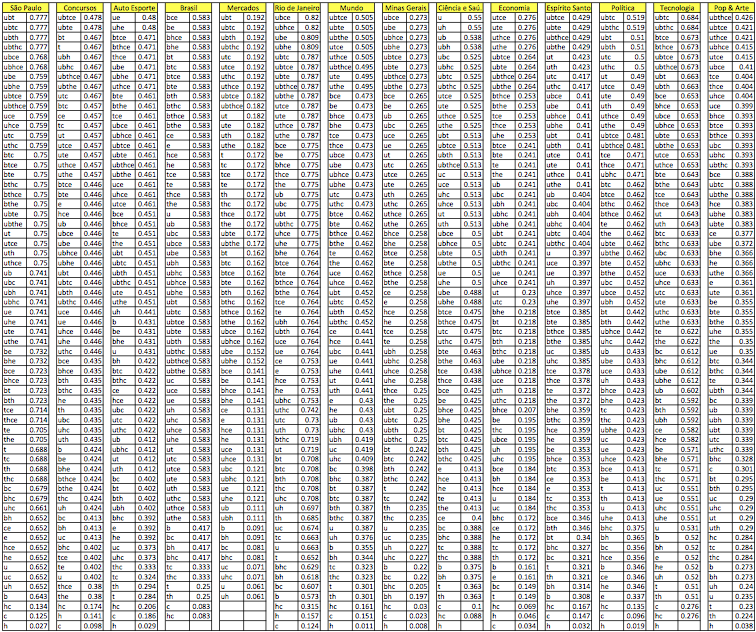


Figura 12 – Tabela de acurácia por editoria com 5 recomendações e sem o fator de similaridade

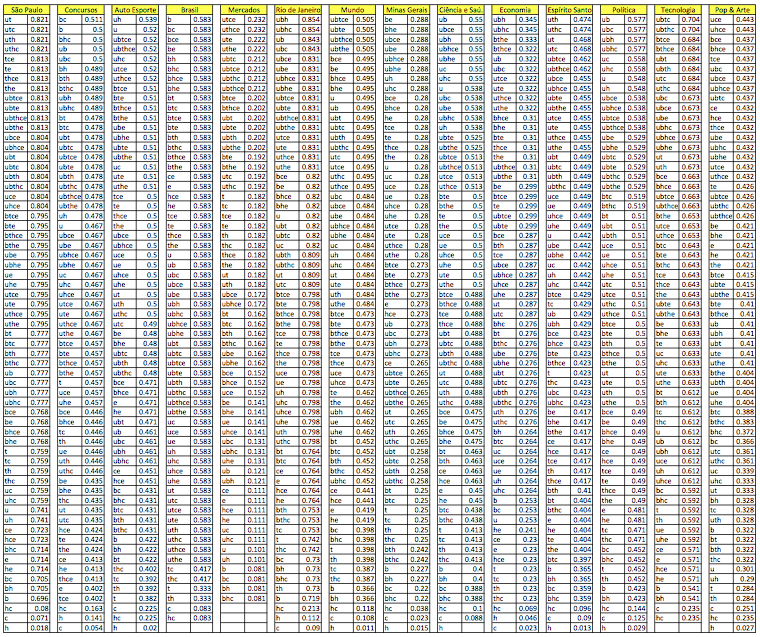


Figura 13 – Tabela de acurácia por editoria com 5 recomendações com o fator de similaridade

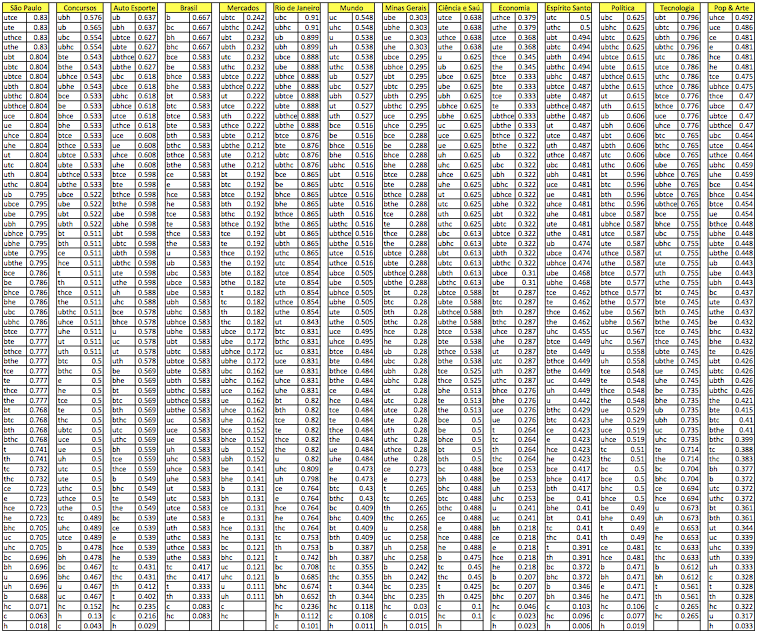


Figura 14 – Tabela de acurácia por editoria com 10 recomendações e sem o fator de similaridade

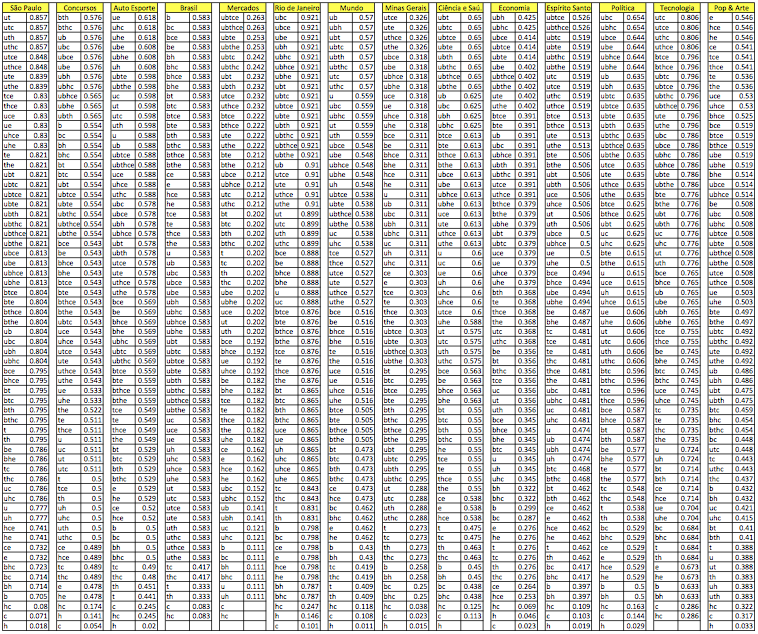


Figura 15 – Tabela de acurácia por editoria com 10 recomendações e com o fator de similaridade