3. GRNews – Sistema de recomendação de segundo nível para suporte a produção de matérias jornalísticas

O **GRNews** é um sistema de recomendações de matérias de segundo nível que fornece aos jornalistas-editores 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 para conteúdo relacionado e executa estas consultas contra uma base de dados de matérias indexadas. 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 com diferentes critérios para compor a lista de matérias relacionadas.

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

3.1. Visão geral do sistema GRNews

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

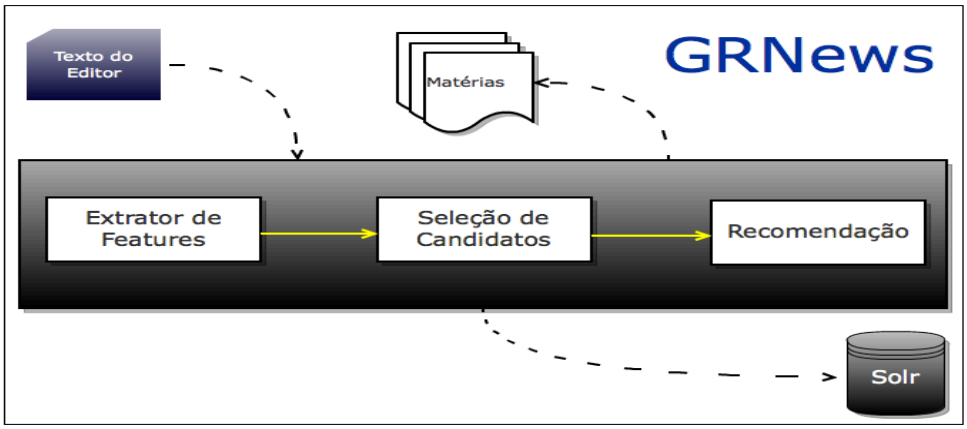

Figura 2 – GRNews arquitetura

Para acionar o **GRNews** é necessário que o editor informe os parâmetros de entrada. O ponto de partida é o formulário de inclusão de matéria, mostrado na Figura 4. 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.

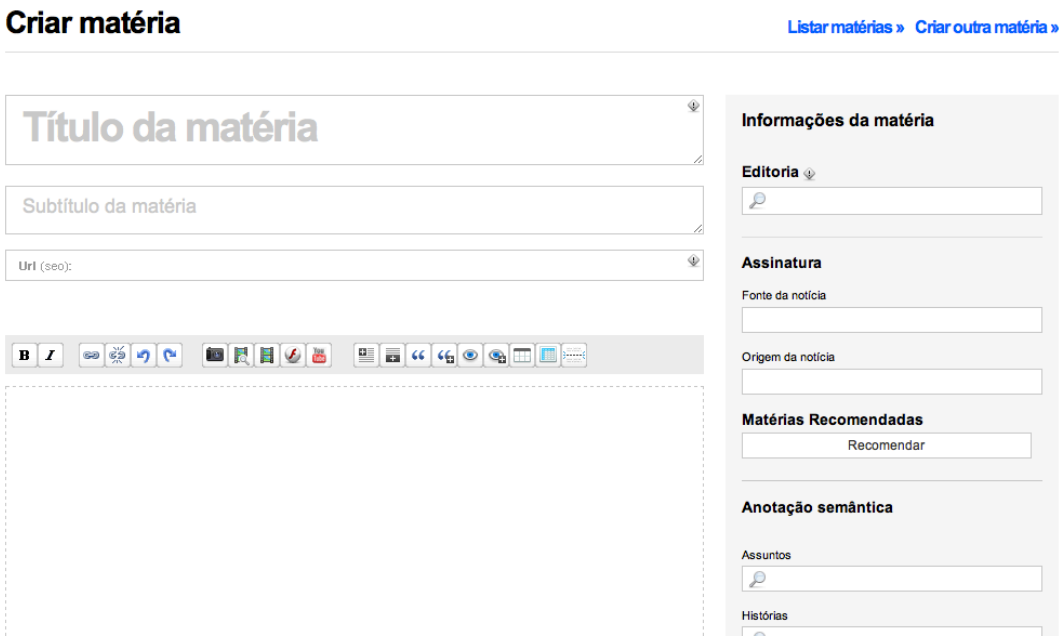

Figura 3 – Interface Web do formulário de matéria

Dentre as informações solicitadas pelo formulário, apresentamos na Tabela 3 apenas aquelas que serão usadas pelo sistema **GRNews,** por entendermos que são suficientes para realizarmos a função de recomendação proposta.

Tabela 3 – Campos do formulário de matéria

Título	O título da notícia é um dado textual limitado em 255 caracteres alfanuméricos.	Obrigatório
Subtítulo	O subtítulo da notícia é um dado textual limitado em 1.000 caracteres alfanuméricos.	Obrigatório
Corpo	O corpo da notícia é um dado textual sem limite de caracteres alfanuméricos. O Editor pode fazer uso de 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 Web site. Exemplos de editoria seriam: 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 3, 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 4 – Componente de matérias recomendadas

3.2. Extrator de *features*

3.2.1. Discussão geral sobre *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.

Nossas *features* são baseadas em informações textuais extraídas do corpo da matéria, definidas como *n-grams*. Um *n-gram* ou *n-grama*, em português, é o nome dado a um conjunto de palavras em sequência obtidas a partir de um texto. Os *n-grams* são classificados de acordo com o total de palavras que os compõem, conforme mostra a Tabela 4.

Tabela 4 – 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 número maior de palavras, iremos adotar para este trabalho apenas *n-grams* com até três palavras.

As seções a seguir descrevem algumas das *features* que usaremos no projeto **GRNews**.

3.2.2. Termos mais frequentes

A primeira *feature* a ser considerada consiste do conjunto dos termos do documento, ordenados por sua frequência de aparição no texto. Segundo Manning et al. [12], a frequência de um termo é o número de vezes que este termo aparece no documento.

A idéia 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 observado por 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 dos termos que apresentam pouco significado (para o texto). Uma dessas técnicas chamada *stopword list*, consiste na elaboração de uma lista de exclusão baseada 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 Manning et al. [12], *tokenização* é o processo de quebrar o documento em partes baseando-se em um padrão de corte. Durante nosso processo de tokenização, os termos são separados à 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 da *bag of word*, os *n-grams* são separados de acordo com o número de palavras nos seus respectivos vetores de *unigrams*, *bigrams* e *trigrams*. Em seguida, 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.

3.2.3. Tags HTML informativas

Tags HTML informativas são marcadores em HTML que enfatizam a importância de uma parte 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: , que indica textos em itálico; e , que indica textos 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. Os ngrams presentes em tags informativas possuem um peso maior durante a fase de seleção da matérias candidatas pois acreditamos que estas tags tendem a resumir o assunto que esta sendo tratado em toda a matéria. Por exemplo, no texto da Figura 5, serão selecionados os termos “Festival de Cinema e Cultura da Diversidade Sexual.” (em itálico, no começo do texto) e “Biblioteca Pública Dolor Barreira” (no final do texto, em negrito).

Figura 5 – Exemplo de matéria

3.2.4. Texto em títulos de vídeos e fotos

Os termos que ocorrem dentro de estruturas que apresentam 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, a fim 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 5, 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.

3.2.5. 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 de 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 Toda e Schone [18], a tarefa de extração de entidades 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 existem como, por exemplo, o trabalho de Maia [13], que utiliza sintagmas nominais para classificação automática de documentos. Ruy, Cicero e Júlio [30,31], apresentam estudos que utilização arvores de decisão DT combinadas aprendizado guiado por regras de transformações TBL para construir tarefas de processamento de linguagem natural em língua portuguesa.

Dada a dificuldade de encontrar uma ferramenta eficiente que pudesse ser usada e alterada para atuar na atividade de extração de entidades nomeadas (ver Seção 2.2.2), partimos para o desenvolvimento de uma solução própria, utilizando conceitos de aprendizado de máquina. A ferramenta desenvolvida possui um processo que separa as atividades em fases encadeadas, onde dado um texto de entrada, a ferramenta processa e gera como saída uma lista de possíveis entidades reconhecidas.

Conforme resume a Figura 6, a ferramenta de extração de entidades proposta possui 3 fases que são apresentadas a seguir.

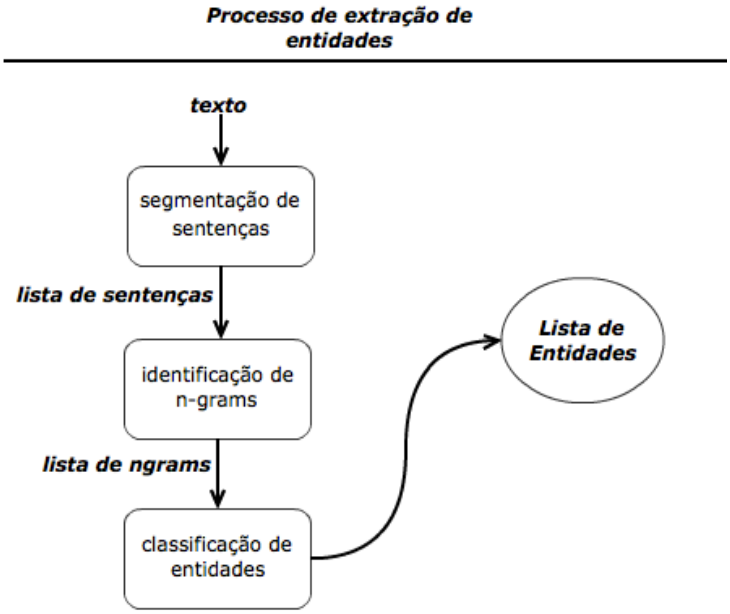

Figura 6 – Processo de extração de entidades

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 fase, 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*, 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, explicado mais adiante, para remover 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*.

   Por exemplo, 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 como conclusão a lista de *n-grams* mostrada na Tabela 5.

Tabela 5 – 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, respondesse '*sim'*, se reconhecesse que este *n-gram* representava uma entidade, ou '*não*', caso contrário. Diante desta necessidade, ficou clara a idéia 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 idéia foi treinar o classificador com base na classe gramatical dos termos presentes no *n-gram*. A fim 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 6) para treinar o classificador.

Tabela 6 – 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 um *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, divididas 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. Obtivemos uma acurácia de aproximadamente 90%, como mostra a Figura 7.

```
# 51397 tagged sents, training on 41118
# training AffixTagger with affix -3 and backoff <DefaultTagger: tag=-None->
# training <class 'nltk.tag.sequential.UnigramTagger'> _tagger with backoff <AffixTagger:
size=6218>
# training <class 'nltk.tag.sequential.BigramTagger'> _tagger with backoff
<UnigramTagger: size=21175>
# training <class 'nltk.tag.sequential.TrigramTagger'> _tagger with backoff
<BigramTagger: size=11323>
# evaluating TrigramTagger
# accuracy: 0.898660
```

Figura 7 – 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 Figura 8.

```
accuracy:  0.924668206905
Most Informative Features
             firstletter = 'LC'              nao : sim    =   1910.0 : 1.0
                  classe = u'KC+N'           nao : sim    =    107.0 : 1.0
                  classe = u'ART'            nao : sim    =     57.0 : 1.0
                  classe = '-None-+-None-'    sim : nao    =     53.4 : 1.0
                  classe = u'NPROP+NPROP+NPROP'    sim : nao    =     51.6 : 1.0
                  classe = u'NPROP+N+NPROP'    sim : nao    =     50.6 : 1.0
                  classe = u'NPROP+KC'       nao : sim    =     35.0 : 1.0
                  classe = u'PRO-KS-REL'     nao : sim    =     34.7 : 1.0
                  classe = u'PDEN'           nao : sim    =     33.8 : 1.0
                  classe = u'NPROP+NPROP+N'    sim : nao    =     30.1 : 1.0
                  classe = u'N+PREP+N'       nao : sim    =     29.8 : 1.0
                  classe = u'-None-+NPROP+NPROP'    sim : nao    =     29.6 : 1.0
                  classe = u'NPROP+NPROP+ADJ'    sim : nao    =     27.3 : 1.0
                  classe = u'N+KS'           nao : sim    =     25.7 : 1.0
                  classe = u'NPROP+N+N'      sim : nao    =     22.7 : 1.0
```

Figura 8 – 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 uma acurácia de aproximadamente 94%, como pode ser visto na Figura 9.

```
# accuracy:  0.940168934772Most Informative Features
#                   classe = u'PREP+N'         não : sim    =    151.7 : 1.0
#                     pre1 = u'VAUX'           não : sim    =     94.7 : 1.0
#                   classe = u'-None-+NPROP'    sim : não   =     58.0 : 1.0
#                   classe = u'PCP'            não : sim    =     42.9 : 1.0
#                   classe = '-None-'           sim : não   =     42.2 : 1.0
#                   classe = u'PREP+N+PREP'    não : sim    =     40.9 : 1.0
#                     pos1 = u'ADJ'            não : sim    =     32.0 : 1.0
#                   classe = u'KC'             não : sim    =     27.7 : 1.0
#                   classe = u'N+NPROP+NPROP'    sim : não   =     27.7 : 1.0
#                   classe = u'ADJ+PREP'       não : sim    =     26.3 : 1.0
#                     pos1 = u'NUM'            não : sim    =     22.8 : 1.0
#                   classe = u'NPROP+PREP'     não : sim    =     20.6 : 1.0
#                   classe = u'NPROP+NPROP'     sim : não   =     19.8 : 1.0
#                   classe = u'NPROP+PREP+N'    sim : não   =     19.2 : 1.0
#                   classe = u'NPROP+ADJ+NPROP'    sim : não   =     17.7 : 1.0
```

Figure 9 – 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. 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 7 temos um comparativo das entidades que foram reconhecidas pelos serviços estudados (ver Seção 2.2.2), comparados ao nosso extrator. Podemos notar que o nosso classificador foi capaz de reconhecer, para o texto do exemplo abaixo, 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 7 – 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 (novamente ver Seção 2.2.2) 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 pares, onde o primeiro valor era a própria sentença e o segundo valor era o nome da entidade nomeada, como ilustra 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 à Tabela 8 que aponta o aproveitamento em percentual de acertos por serviço.

Tabela 8 – Comparativo entre extratores

Ltasks	63%
Zemanta	32%
Nltk	42%
Yahoo	33%
Nosso Extrator	87%

Como podemos observar na Tabela 8, 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. Todavia é importante ressaltar que as demais ferramentas

não tiveram acesso ao mesmo corpus de entidades utílizado pelo nosso classificador o que não torna nossa estratégia melhor apenas no permite dizer que dadas as circunstâncias de um corpus proprietário, optamos por usar o nosso classificador por que este mostrou-se mais aderente ao nosso problema.

Outra observação importante é o fato de o nosso classificador ter sido treinado para reconhecer apenas a existência ou não de uma entidade. Esta é uma tarefa bem mais simple que reconhecer o tipo da entidade (Lugar, Pessoa e etc). As demais ferramentas apresentam a preocupação no reconhecimentos dos tipos das entidade e esse é um dos motivos pelo qual a acurácia é menor em relação a nossa estratégia.

3.3. Seleção de candidatos

A fase de seleção de candidatos consiste em realizar consultas em uma base de notícias 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** [16] faz uso de uma fase de separação de artigos candidatos, utilizando 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 executar buscas sobre uma base de notícias.

3.3.1. 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 da biblioteca do Lucene [22]:

$$score(q,d) = coord(q,d) * queryNorm(q) * \sum_{t\ in\ q} (tf(t\ in\ d) * idf(t)^2 * t.getBoost() * norm(t.d))$$

Função 5 – 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*.

3.3.2. 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**", vista na Figura 10, quando combinamos as *features unigrams* mais frequentes e entidades nomeadas, obtivemos o resultado da Tabela 9.

Tabela 9 – 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

Podemos perceber, na Tabela 9, que devido a fragilidade do extrator a palavra "prefeitura" encontrada no quarto parágrafo do texto, visto na Figura 10, não foi selecionada como entidade embora esteja referindo-se a prefeitura de Goiânia.

06/10/2011 12h27 - Atualizado em 06/10/2011 13h48

Reabertura do Zoológico de Goiânia é adiada para 2012

Previsão era que o parque voltasse a funcionar no dia 12 de outubro. Essa é a terceira vez que a prefeitura adia a reinauguração.

Do G1 GO, com informações da TV Anhanguera imprimir

A abertura do Zoológico de Goiânia, que estava **prevista para o dia 12 de outubro**, Dia das Crianças, foi adiada para 2012. O mês da reinauguração não foi confirmado. Essa é a terceira vez que a reabertura do parque foi adiada. A prefeitura já havia prometido a reinauguração para julho de 2010 e para janeiro de 2011.

O parque foi interditado pelo Instituto Brasileiro do Meio Ambiente e dos Recursos Naturais Renováveis (Ibama) em julho de 2009, depois da morte de vários animais. **Somente no mês de setembro de 2011, dez animais morreram.** Segundo a direção do zoológico, existem atualmente 480 animais no parque.

Na terça-feira (4), o Ibama e o Ministério Público Federal (MPF) estiveram no local fazendo uma vistoria. "As obras estão bastante avançadas, comparado com a penúltima vistoria realizada, que aconteceu no mês de fevereiro. Nós percebemos que tem sido disponibilizado pessoas e que estão sendo empenhados esforços para o cumprimento do Termo de Ajustamento de Conduta," ressalta a coordenadora de fauna do Ibama, Cristiane Borges Miguel. Segundo ela, nenhuma solicitação de reabertura foi apresentada ao órgão.

O último prazo dado à prefeitura pelo Ibama para a reabertura do zoológico vence em fevereiro de 2012. Caso o parque não seja aberto dentro do prazo, a Amma - que é a responsável pelo local - poderá ser multada.

Reforma

Hoje 130 homens trabalham nas obras do zoo. A maioria está concentrada na reforma dos recintos dos animais. Entre as maiores reformas estão a da rede de esgoto, que será ampliada, e a rede elétrica, que será substituída por uma subterrânea.

saiba mais

Obras do Zoológico de Goiânia estão 70% concluídas, diz direção do parque

Conselho Municipal de Turismo aponta 11 pontos críticos em Goiânia

Figura 10 – Exemplo de matéria publicada

3.3.3. 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.

Para cada combinação de *feature*, uma consulta é montada e é submetidas à base de notícias 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*.

3.4. Recomendação

A fase de recomendação consiste em sugerir ao editor as cinco melhores matérias consideradas como melhor relacionadas com a matéria que está sendo criada. Para alcançar uma recomendação desejável, o sistema permite que o editor escolha o fator de ordenação que será usado para corrigir o *score* da matéria. Os fatores possíveis são - Fator de Similaridade e Fator de Popularidade – discutidos a seguir.

3.4.1. Fator de Similaridade

Na similaridade de textos, 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 à 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 não é tão eficiente no que diz respeito ao grau de relacionamento dos textos. Segundo Yuanhua et al. [1], a medida de similaridade

de texto sozinha não é suficiente para capturar o relacionamento entre duas matérias. Um exemplo apresentado em [1] 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 texto. Ainda no trabalho de Yuanhua et al. [1], os autores definem em seu trabalho 3 critérios a fim de modelar uma função capaz de produzir 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, Getahun et al. [9] entendem que o fator de relacionamento semântico entre duas notícias pode ser entendido por 3 idéias simples: *Inclusão*, quando uma notícia está inserida em outra; *Interseção*, quando duas notícias apresentam conceitos em comum; e *Oposição* quando as notícias apresentam ideias contrárias sobre o mesmo assunto. Além destes trabalhos, é possível encontrar outros que tratam do tema de relacionamento de conteúdo, mostrando-nos a dificuldade que o tema implica.

3.4.2. Fator de popularidade

Para cada matéria cadastrada na base de dados, calculamos um fator de popularidade com base nos links criados entre as matérias pelo elemento de matérias relacionadas. Para gerar o fator de popularidade das matérias, utilizamos o algoritmo de *pagerank* do Google.

Após a geração dos *ranks* das matérias, os mesmos foram armazenados na base de dados. Uma vez obtido o *rank* da matéria, ele é multiplicada ao *score* dado à matéria pelo servidor de busca e a listagem de candidatas é então reordenada.

3.5. Comentários do projeto

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

***PYTHON*:** Linguagem de programação de alto nível que permite a escrita

de código conciso e poderoso. Possui uma estrutura de código simples, de modo que a maioria dos programadores consegue facilmente ler e entender um programa nela escrito.

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 ensino mas, com o passar dos anos, tem sido adotado pela indústria e por pesquisadores.

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

Durante nosso trabalho, fizemos uso do corpus **MAC-MORPHO** encontrado no **NLTK**.

***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.