UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Geração multi-parametrizada de *corpora* linguísticos

Nayara Rosy Felix da Silva

SÃO CRISTÓVÃO/ SE
2015

UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Nayara Rosy Felix da Silva

Geração multi-parametrizada de *corpora* linguísticos

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PROCC) da Universidade Federal de Sergipe (UFS) como parte de requisito para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Hendrik Teixeira Macedo
Co-orientador: Dr. Luciano de Andrade Barbosa

SÃO CRISTÓVÃO/ SE
2015

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

Silva, Nayara Rosy Felix da
S586g Geração multi-parametrizada de *corpora* linguísticos / Nayara Rosy Felix da Silva ; orientador Hendrik Teixeira Macedo. – São Cristóvão, 2015.
121 f. : il.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Sergipe, 2015.

1. Software - Desenvolvimento. 2. Linguística – Processamento de dados. 3. Princípios e parâmetros - Linguística. 4 . Sistemas de recuperação da informação. l. Macedo, Hendrik Teixeira, orient. ll. Título.

CDU 004.775

Nayara Rosy Felix da Silva

Geração multi-parametrizada de *corpora* linguísticos

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PROCC) da Universidade Federal de Sergipe (UFS) como parte de requisito para obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

Prof. Dr. Hendrik Teixeira Macedo, Presidente
Universidade Federal de Sergipe (UFS)

Dr. Luciano de Andrade Barbosa, Co-Orientador
IBM Research

Prof. Dr. João Batista da Rocha Junior, Membro
Universidade Estadual de Feira de Santana (UEFS)

Prof. Dr. Tarcísio da Rocha, Membro
Universidade Federal de Sergipe (UFS)

Geração multi-parametrizada de *corpora* linguísticos

Este exemplar corresponde à redação da Dissertação de Mestrado, sendo o Exame de Qualificação do mestrando **Nayara Rosy Felix da Silva** para ser aprovada pela Banca examinadora.

São Cristóvão - SE, 31 de Agosto de 2015

Prof. Dr. Hendrik Teixeira Macedo

Orientador

Dr. Luciano de Andrade Barbosa

Co-Orientador

Prof. Dr. João Batista da Rocha Junior

Membro

Prof. Dr. Tarcísio da Rocha

Membro

Agradecimentos

Primeiramente Deus q permitiu q tudo isso acontecesse, longo d minha vida, nã somente nestes anos de estudo, ms que m todos s momentos é o maior mestre q alguém pode conhecer.

Obrigado aos meus pais Antonio e Roseane, ao meus irmão Felipe e ao meu namorado Guilherme por todo apoio e amor incondicional, sem eles nem teria chegado a completar esta etapa da minha vida.

Ao meu orientador, Prof. Dr. Hendrik Teixeira Macedo, que considero um excelente professor e orientador, obrigado por suas orientações e ensino sem o qual está dissertação não seria possível. Ao meu co-orientador Dr. Luciano de Andrade Barbosa por todo ensinamento e que sempre se mostrou disposto a me auxiliar quando tive dúvidas, e todo o ensino foi muito importante para minha vida profissional.

Aos professores da PROCC-UFS que estiveram comigo, tirando dúvidas e sempre contribuindo com sugestões valiosas.

Agradeço à Flávio, Thiago e Carlos Júnior pelos dois anos de pesquisa que finalizou em minha dissertação. Todo esse projeto e trabalho não seria o mesmo sem a ajuda e acompanhamento de vocês.

Expressando de forma generalista obrigado a todos os meus amigos e colegas que estiveram comigo nesta caminhada, obrigado pelo apoio e incentivo para a realização deste trabalho.

Lista de Figuras

2.1 Arquitetura de alto nível do software de um sistema de RI. Fonte:Borges e Mourão, 2011. p. 7. 8
2.2 Arquitetura de alto nível de um *Web Crawler*. Fonte: Elaborada pelo autor . 18
2.3 Arquitetura de um *focused crawler* adaptada de Chakrabarti, Berg and Dom 1999 22

3.1 Arquitetura do *framework* ACHE Adaptada de Barbosa (2007) 32
3.2 Passos para a construção do *corpus* CETEMPúblico proposta por Rocha e Santos (2000) 39

4.1 Casos de uso disponibilizados pela interface a um utilizador humano quando este efetua a configuração do sistema 51
4.2 Arquitetura de alto nível do pré-processamento 52
4.3 Arquitetura de alto nível do pós-processamento 53

5.1 Perplexidade média de cada *corpus* para unigrama. 60
5.2 Perplexidade média de cada *corpus* para bigrama. 60
5.3 Perplexidade média de cada *corpus* para trigrama. 61
5.4 Perplexidade média de cada *corpus* para 4-grama. 61
5.5 Perplexidade média de cada *corpus* para 5-grama. 62
5.6 Fases de melhoria e expansão da geração corpus 72
5.7 Trecho do *corpus* Paramopama 73
5.8 Visão geral da Solução 82
5.9 Melhor quantidade de iterações 85
5.10 Número de termos selecionados pela fase de Aprendizado 86

5.11 Melhor limite de Perplexidade 86
5.12 Uso de emoticons 87

6.1 Tela para coleta de informações do futuro *corpus* 92
6.2 Tela para módulo de construção do modelo de linguagem 93
6.3 Tela para módulo de construção do modelo de entidades nomeadas 94
6.4 Tela para módulo de seleção do Pós-processamento do *Corpus* 95

Lista de Tabelas

3.1 Comparação de características entre os trabalhos com os trabalhos com *Focused Web Crawler* em Inglês 38
3.2 *Corpora* em Português 46

4.1 Descrição de casos de uso - crawler 52

5.1 Conjunto de Treinamento 56
5.2 Conjunto de teste 57
5.3 Perplexidade dos conjuntos de testes 1 a 4 58
5.4 Perplexidades para os conjuntos de testes 5 a 8 59
5.5 Valores de λ paca cada interpolação linear entre os dois *corpora* (conjunto de testes do 1 ao 4). 62
5.6 Perplexidade para os conjuntos de testes do 1 ao 4 (interpolação) 68
5.7 Valores de λ para cada interpolação linear entre os dois *corpora* (conjuntos de teste do 5 ao 8). 69
5.8 Perplexidade para os conjuntos de testes do 5 ao 8 (interpolação). 70
5.9 Perplexidade para os *corpora* Floresta, Poxim e Chave como conjunto de testes 71
5.10 Tamanho dos Corpora PTBR-NE 71
5.11 Tamanho dos Corpora PTBR-NE - (Número de entidades nomeadas) 71
5.12 Algumas características usadas pelo classificador NER 74
5.13 Resultados para PESSOA - Conjunto de testes 1 75
5.14 Resultados para LOCALIDADE(Conjunto de dados 1) 75
5.15 Resultados para ORGANIZAÇÃO (Conjunto de dados 1) 76
5.16 Resultados para TEMPO (Conjunto de dados 1) 77
5.17 Pontuação Geral (Conjunto de dados 1) 77

5.18 Resultados para PESSOA (Conjunto de testes 2) 78
5.19 Resultados para LOCALIDADE(Conjunto de dados 2) 78
5.20 Resultados para ORGANIZAÇÃO (Conjunto de dados 2) 78
5.21 Resultados para TEMPO (Conjunto de dados 2) 79
5.22 Pontuação Geral (Conjunto de dados 1) 79
5.23 Configurações de diferentes valores de parâmetros 85
5.24 Melhor Conjunto de características 88
5.25 Resultado final 88

Lista de Siglas

ACHE - *Adaptive Crawler for Hidden-Web Entries*

BFS - *Breadth-First Search*

CFC - *Context Focused Crawler*

CRF - *Conditional Random Fields*

DOM - *Document Objet Model*

DSFC - *Domain-Specific Form Classifier*

DFS - *Depth First Search*

DNS - *Domain Name Service*

FFC - *Form Focused Crawler*

FIFO - *First-In-First-Out*

HCLA - *Hypertext Content Analysis latent*

HITS - *Hyperlink Induced Topic Search*

HMM - *Hidden Markov Model*

HTML - *Hipertext Markup Language*

HTTP - *Hypertext Transfer Protocol*

IDF - *Inverse Document Frequency*

LSI - *Latent Semantic Indexing*

NER - *Named Entity Recognition*

NLTK - *Natural Language Toolkit*

RI - Recuperação de Informação

SFC - *Searchable Form Classifier*

SVD - *Singular Value Decomposition*

URL - *Universal Resource Locator*

VSM - *Vector Space Model*

WaCky - *Web-As-Corpus Kool Yinitiative*

WWW - *World Wide Web*

Resumo

O desenvolvimento de software de Processamento de Linguagem Natural (PLN) é altamente dependente da boa qualidade do que chamamos de *corpus* Linguístico. Um *corpus* é uma coleção de textos processáveis pelo computador, mas produzidos dentro de um ambiente comunicativo natural. Essa dependência advém do fato de que a maior parte do trabalho realizado com PLN hoje em dia está relacionado ao uso de técnicas de Aprendizado de Máquina para criação de modelos de linguagem. Para sistemas que permitem correção automática e previsão de palavras e sentenças, por exemplo, modelos linguísticos são gerados a partir de um *corpus* de texto a partir de transcrições usando N-Gram. Infelizmente, corpora linguísticos em português do Brasil não são abundantes e sua confecção não é uma tarefa trivial. Um caminho para a confecção de *corpus* linguísticos em Português é a confecção de *Crawlers*Web. *Focused Crawlers*, em particular, têm o propósito de coletar páginas da Web que sejam relevantes a um tópico ou interesse específico do usuário. *FocusedCrawlers* existentes ainda não atendem completamente a necessidades específicas e toda potencialidade de um sistema de PLN. Esta dissertação de mestrado se propõe a contribuir com o Estado da Arte ao propor uma ferramenta para a confecção automática de corpora bem representativos ao objetivo do usuário que possam ser balanceados em respeito a fatores tais como tipo de coleta, domínio, língua, formalidade do discurso e rotulação do texto. A ferramenta permite ainda que etapas de pós-processamento sejam realizadas, como por exemplo limpeza do *corpus*, construção de um modelo de linguagem e de um modelo de entidades nomeadas. Dois corpora foram criados em duas formas de coleta distintas: por dados da Web (*corpus* VazaBarris) ou por dados do Twitter (*corpus* Poxim). O *corpus* VazaBarris possui 86 milhões de palavras e o Poxim possui 3 milhões de palavras. Estes corpora foram avaliados por meio da criação de modelo de linguagem e comparação com dois outros corpora em Português. Os resultados mostram que Poxim alcançou o melhor valor de perplexidade. Poxim também traz maior contribuição quando interpolado com algum outro *corpus*. Além dos corpora, foi criado um método de coleta automática para streaming de dados, utilizando o algoritmo de *Relevance Feedback*. Segundo os resultados, utilizar *Relevance Feedback* para a coleta dos dados melhorou o valor de perplexidade com o *corpus* coletado inicialmente. Um terceiro *corpus* foi criado para rotulação de Entidades Nomeadas, o Paramopama. O Paramopama

é uma versão estendida PtBR do *corpus* WikiNer, com inclusão das entidades Pessoa, Localização, Organização e Tempo. Os resultados mostram que o Paramopama apresentou melhoria para as métricas de Precisão, Cobertura e Medida-F na comparação com outros três corpora do Estado da Arte.

Sumário

1 Introdução **1**

2 Fundamentação Teórica **6**

2.1 Recuperação de Informação 6

2.1.1 Arquitetura de Sistemas de Recuperação de Informação 7

2.1.2 Modelos de Recuperação de Informação 7

2.1.3 Busca da *Web* 13

2.2 Modelo de linguagem 15

2.3 *Web Crawler* 16

2.3.1 Arquitetura Básica de um Crawler 18

2.3.2 Métodos de busca e métricas 19

2.4 *Focused Web Crawler* 21

2.4.1 Arquitetura de um *Focused Web Crawler* 21

2.4.2 Abordagens baseadas em estrutura de Links 23

3 Trabalhos Relacionados **24**

3.1 Focused Web Crawler 24

3.1.1 *Best first* 24

3.1.2 Naive Bayes 30

3.1.3 PageRank 34

3.1.4 *Breadth-First* 35

3.1.5 Resumo Comparativo 36

3.2 Criação de *Corpora* para a língua Portuguesa 37

3.2.1 CETEMPúblico 38

3.2.2 Corpus CHAVE 40
3.2.3 Corpus HAREM 41
3.2.4 Corpus WaCky 41
3.2.5 Corpora Floresta Sintá(c)tica 43
3.2.6 Corpus Brasileiro 43
3.2.7 Corpus RELI 44
3.2.8 Resumo comparativo 45

4 Proposta de parametrização **47**
4.1 Pré-Processamento 47
4.2 Pós-processamento 49
4.3 Especificação 50

5 Construção de *Corpora* **54**
5.1 Construção dos *corpora* VazaBarris e Poxim 55
5.1.1 VazaBarris 55
5.1.2 Poxim 55
5.1.3 Experimentos e Resultados 56
5.2 Paramopama: um *corpus* brasileiro para reconhecimento de entidades nomeadas 63
5.2.1 Método de construção 64
5.2.2 Avaliação e Resultados 66
5.3 Método para coleta de dados de *streaming* para modelo de linguagem . . . 80
5.3.1 Solução Proposta 80
5.3.2 Relevance Feedback 81
5.3.3 Descrição da Solução 81
5.3.4 Avaliação Experimental 83
5.3.5 Número de tweets do corpus usados para a fase de Aprendizado . . 84
5.3.6 Número de termos selecionados pela fase de Aprendizado 84
5.3.7 Limite de perplexidade 86
5.3.8 Uso de Emoticons nos Dados para Aprendizado 87
5.3.9 Comparação com Baselines 87

5.4 Exemplo prático de aplicação 88

6 Ferramenta Web para geração multi-parametrizada de *corpora* linguísticos 90

6.1 Requisitos da ferramenta de gerenciamento 90

6.2 Demonstração da ferramenta de gerenciamento 91

6.2.1 Módulo de Coleta 91

6.2.2 Módulo de módelo de Linguagem 92

6.2.3 Módulo de Entidades Nomeadas 93

6.2.4 Módulo de Pós-Processamento 93

6.3 Implementação 94

6.4 Ambiente de experimentação 96

7 Conclusão 97

Referências 98

Capítulo 1

Introdução

O Processamento da Linguagem Natural (PLN) lida com problemas relacionados à automação da interpretação e da geração da língua humana em aplicações como Tradução Automática (MEDEIROS CASELI, 2007), Sumarização Automática de Textos (HEARST, 1999), Categorização Textual (BRA, 1994), Recuperação e Extração de Informação (BRA, 1994), entre muitas outras, além das tarefas relacionadas de criação e disponibilização de dicionários/léxicos e *corpus* eletrônicos (SARDINHA, 2000), desenvolvimento de taxonomias e ontologias (ALMEIDA CAMPOS; GOMES, 2010), investigações em linguística de *corpus* (SARDINHA, 2000). O PLN é visto como uma subárea da Inteligência Artificial devido ao fato das primeiras indagações sobre o processamento automático das línguas naturais terem sido motivadas por uma das preocupações da IA, a saber: a interação homem-máquina via "língua dos homens".

O desenvolvimento de software de PLN de qualidade hoje em dia é altamente dependente da boa qualidade do que chamamos de *Corpus* Linguístico. O estudo de *corpus* linguístico é definido como linguística de *corpus*. Linguística de *corpus* trata-se de uma área que coleta e explora conjuntos de dados linguísticos textuais com o propósito de servirem para a pesquisa em uma língua ou variedade linguística (SARDINHA, 2000) . Considerando estudos nesta área é possível afirmar que uma linguagem é padronizada quando é formada por sequências de palavras em forma de padrões, e esses padrões variam de acordo com as diferentes situações e contextos em que ocorrem.

Um *corpus* é uma coleção de textos processáveis pelo computador, mas produzidos dentro de um ambiente comunicativo natural. Essa dependência advém do fato de que a maior

parte do trabalho realizado com PLN hoje em dia está relacionado ao uso de técnicas de Aprendizado de Máquina para criação de modelos de linguagem. Para sistemas que permitem correção automática e previsão de palavras e sentenças, por exemplo, modelos linguísticos são gerados a partir de um corpus de texto a partir de transcrições usando N-Gram.

Apesar das pesquisas em PLN de português terem se iniciado muito antes da década de 1990, praticamente nada havia sido feito que visasse à criação de uma ferramenta robusta e de uso genérico, que requer recursos linguísticos e computacionais de grande monta.

Baseado nisso, pode-se utilizar a *World Wide Web* (WWW) (ou simplesmente *Web*) para o processo de construção do *corpus* linguístico. A *web* consiste num imenso repositório de informações que divergem em conteúdo, qualidade e tempo de vida. Estas informações trazem novos desafios relacionados à área de recuperação de informação. Dessa forma, surgiram sistemas de Recuperação de Informação (RI) para Web como uma alternativa para auxiliar o usuário em encontrar informações relevantes na web e, assim, conceber *corpora* linguísticos eficientemente. Esses sistemas de RI são chamados de motores de busca.

Web Crawlers também são utilizados pelos motores de busca na Web, como o Google [1] ou Yahoo[2], para coletar documentos. Para conseguir seu objetivo, um *crawler* atravessa pelo grafo *web* buscando páginas e armazenando as informações necessárias (texto, *hyperlinks*, figuras) para a sua tarefa. O início da etapa de rastreamento se dá pela coleta de um conjunto inicial de URLs sementes e seus *outlinks* são colocados em uma nova estrutura, a fronteira de URL. O processo continua de forma iterativa, visitando todas as páginas que estão na fronteira até que a fila esteja vazia ou quando determinar um critério de parada.

Ultimamente, contudo, motores de busca se intensificaram a fim de manter-se com a crescente quantidade de informações disponíveis na *web*. Porém, três problemas que são usualmente desafiadores aos crawlers acontecem atualmente: a grande aquisição de informação, a volatilidade da informação obtida e o conteúdo em páginas geradas dinamicamente ocasionando redundância de informação em URL distintas. Baseado nisso é necessário ter uma política de revisita para garantir a confiabilidade do processo.

Dessa forma, a tarefa de buscar informações específicas na *web* está se tornando uma tarefa difícil assim como o retorno de informações gerais pode não ser de interesse geral para

[1] http://www.google.com

[2] https://br.yahoo.com

todos os usuários. Dessa forma, um *web crawler* geral pode ser construído e aperfeiçoado de acordo com algum conhecimento de domínio. Dependendo da sua finalidade, um *web crawler* pode ser classificado em duas categorias: *standard crawler*, que visa coletar todos os tipos de páginas, e *focused crawlers* que visam reunir apenas as páginas pertencentes a um conjunto determinado de temas.

Utilizar *focused crawler* permite ao motor de busca operar com eficiência dentro de um espaço limitado da *web* (ZHANG; YIN; YUAN, 2007). Os *corpora* recolhidos por um *focused crawler* não são limitados a uma única língua, podendo apresentar *corpus* em português (LARANJEIRA, 2014).

O procedimento básico do funcionamento de um *focused crawler* é iniciado com diversas páginas sementes que são relevantes ao tópico. Sempre que o *crawler* busca uma nova página Web, as URLs não visitadas são extraídas a partir desta página e marcadas por uma relevância para o tema. O *crawler* busca a próxima URL que tenha a maior pontuação de relevância para iniciar o seu rastreamento.

Muitas pesquisas utilizam *focused web crawlers* (CHAKRABARTI; BERG; DOM, 1999b) (AGGARWAL; AL-GARAWI; YU, 2001) (BATSAKIS; PETRAKIS; MILIOS, 2009), porém dois fatores determinam a qualidade dos resultados da busca: cômputo de similaridade e ordenação das URLs na fronteira. O cálculo da similaridade é conhecido como "*web ranking*" (PRASATH; OZTüRK, 2011). O segundo relaciona ao problema de atribuir uma ordem adequada para as páginas não visitadas.

O *focused crawler* poderia atender a diversos objetivos específicos, como a língua, domínio, contexto, formalidade do discurso, tipos de sentenças, além do tipo de discurso: dados conversasionais, por exemplo, são bem menos abundantes em páginas Web convencionais do que em redes sociais, como o Facebook, ou sistemas de microblogging, como o Twitter (PRIYATAM, 2014).

Porém, ocorre uma ausência de *corpora* linguísticos na língua portuguesa do Brasil(*corpora* ptBR). O estudo e compilação de *corpora* ptBR está crescendo, porém predominam *corpora* compostos por textos de jornal. Dessa forma, a maioria dos *corpora* pbBR contêm uma carência de avaliações de qualidade do *corpus*, e em particular relacionadas ao tipo de discurso: seja ele de dados conversacionais ou dados formais, como em textos jornalísticos.

Estão sendo compilados textos criados especificamente para ambientes digitais, como o Twitter, e que pese a relativa facilidade na obtenção de dados de blogs e o interesse pela linguagem utilizada. Porém, ocorre uma carência de avaliações sobre esses dados avindos de blogs ou microblogging, em especial o Twitter.

Para a compilação de *corpora* linguísticos seria necessário uma ferramenta que permitisse ao interessado na construção do *corpus* facilidade para a geração do mesmo. Porém, esse tipo de ferramenta está em ausência na literatura atual.

Esta dissertação de mestrado tem as seguintes contribuições:

- Investigação aprofundada de parâmetros relevantes para a geração de *corpora* linguísticos, o objetivo de utilizar essa investigação é tornar de fato o interessado em criação de um *corpus*, o autor central do processo. Os elementos desta parametrização devem ser devidamente categorizados entre duas etapas: (1) Pré-processamento e (2) Pós-processamento. No primeiro caso, o usuário tem a possibilidade de definir tipo de coleta, domínio, língua, tipo de texto, tipo de discurso, entre outros. No segundo caso, ele pode definir como o *corpus* deve ser compilado e manipulado, se a manipulação será por limpeza e formatação dos dados, de maneira a preparar o *corpus* para o processamento computacional, ou será construído o modelo de linguagem ou modelo de entidades nomeadas do *corpus* resultante, enfim, definições sobre o objeto-alvo do problema que deseja atacar com a criação daquele *corpus*. A fim de ilustrar a efetividade dessa investigação de parâmetros, é realizada a geração e avaliação de três *corpora* linguísticos, onde os efeitos da parametrização de ambas etapas de pré-processamento e pós-processamento possam ser observados. Estes três *corpora* também são contribuições práticas para as pesquisas em Linguística Computacional para língua portuguesa e para o Brasil, em particular.

- Para uma avaliação dos dados advindos do microblogging Twitter, será realizada uma investigação utilizando o *Relevance Feedback* para refinamento de uma consulta inicial na compilação do *corpus* de dados do twitter, mostrando que dados do twitter são mais relevantes para modelos conversacionais.

- Desenvolvimento de um software de configuração de *crawler* para geração de *corpora* de acordo com os parâmetros investigados e que possa ser utilizado de forma online

por investigadores de linguística de *corpus*

Sendo assim, podemos enumerar como objetivos específicos dessa dissertação:

1. Propor elementos da parametrização para as etapas de pré-processamento e de pós-processamento na geração de *corpora* linguísticos;

2. Mostrar a efetividade da proposta de parametrização na concepção de três diferentes *corpora* da língua portuguesa: (i) *corpus* com dados provenientes de páginas Web, (ii) *corpus* com dados provenientes do microblogging Twitter e (iii) *corpus* anotado para reconhecimento de entidades nomeadas;

3. Avaliar os *corpora* gerados quanto à qualidade de seu modelo de linguagem;

4. Desenvolver uma ferramenta que reflita os elementos de parametrização propostos, automatize o processo de geração de *corpora* e possa ser utilizada de forma online por pesquisadores, estudantes ou interessados em Linguística de *Corpus* de uma maneira geral.

O restante do texto desta dissertação está organizado nos seguintes capítulos:

- O Capítulo 2 apresenta a fundamentação teórica pertinente a Recuperação de Informações na *Web*, assim como *Crawler* e *Focused Web Crawler* e Modelo de Linguagem;

- O Capítulo 3 apresenta os trabalhos correlatos para a solução do problema e apresenta os principais *corpora* em português;

- O Capítulo 4 apresenta o método do trabalho, incluindo os parâmetros necessários no pré-processamento e pós-processamento e apresenta as especificações do *focused crawler* desejado;

- O Capítulo 5 apresenta a construção de três *corpora* em Português e um método de coleta automática para streaming de dados;

- O Capítulo 6 apresenta a ferramenta como produto proposto;

- O Capítulo 7 apresenta as conclusões obtidas.

Capítulo 2

Fundamentação Teórica

Neste capítulo são abordados os conceitos principais relacionados à Recuperação de Informação (Seção 2.1), *Web Crawler* (Seção 2.3) , Modelo de Linguagem (Seção 2.2), *Focused web crawler* (Seção 2.4).

2.1 Recuperação de Informação

Um sistema de Recuperação de Informação (do inglês *Information Retrieval*) - RI - ajuda os usuários a encontrar e recuperar informações. Para alguns autores, trata da representação, armazenamento, organização e acesso a itens de informação (documentos ou páginas *web*), que juntos fornecem aos clientes facilidade de acesso às informações necessárias (BAEZA-YATES; RIBEIRO-NETO, 1999).

Recuperação de Informação foi originalmente desenvolvida para ajudar a gerenciar a enorme literatura científica. Iniciou com o objetivo de indexar textos e buscar por documentos específicos em uma coleção. Com o aumento da informação cresceu e incluíu as áreas de modelagem, classificação de textos, arquitetura de sistemas, interfaces de usuários, visualização de dados, filtragem e linguagem.

A busca de informações consiste em identificar quais os documentos de um *corpus* atendem a necessidade do usuário. A recuperação de informação lida com textos e envolvem a extração de informações semânticas e sintáticas do texto juntamente com toda problemática com relação ao PLN. O sistema deve interpretar e classificar os documentos encontrados de acordo com o grau de relevância na pesquisa.

2.1.1 Arquitetura de Sistemas de Recuperação de Informação

Antes do início do processo de busca em RI, é necessário obter uma coleção de documentos (própria ou coletada da Web). Esta coleção é armazenada em um repositório, onde os seus documentos são indexados para que as etapas de recuperação e ranqueamento sejam efetuadas mais rapidamente.

Com isso, o processo de busca inicia quando o usuário apresenta ao sistema uma consulta. O usuário traduz nesta a sua necessidade de informação na linguagem do sistema, geralmente apresenta um conjunto de palavras que transmitem a informação necessária. Pela consulta, é feita uma análise sintática e a consulta é expandida com as formas variantes das palavras que ela contém. Em seguida, a consulta é processada e recupera-se os documentos.

Nos documentos, geralmente, são criadas estruturas de dados, com o fim de acelerar o processo de recuperação. Por exemplo, a extração de atributos de um texto, selecionando somente os termos relevantes. Através de um processo de ranqueamento os documentos são retornados aos usuários.

O processo de ranqueamento tem a finalidade de identificar os documentos com uma maior probabilidade de serem considerados relevantes para o usuário. Essa é a etapa mais importante de um sistema de RI, a qualidade final do resultado depende do ranqueamento. Os documentos do topo da lista são formatados para apresentação.

O trabalho de Borges and Mourão (2013) apresenta uma arquitetura de um sistema de RI simples e genérica, acrescentando um módulo necessário aos sistemas de RI para a Web, como motores de busca e *Web Crawlers* conforme ilustra a Figura 2.1 .

A arquitetura proposta utiliza o conceito de índice invertido, que tem o objetivo de acelerar a tarefa de busca em coleções de documentos (BORGES; MOURÃO, 2013). Dessa forma, para cada palavra do vocabulário, o índice armazena identificadores para documentos que contém esta palavra.

2.1.2 Modelos de Recuperação de Informação

Os modelos de um sistema de RI tem o objetivo de produzir uma função de ranqueamento. Cada modelo considera um conjunto de palavras-chave, chamadas de termo de indexação, que aparece no texto de um documento da coleção.

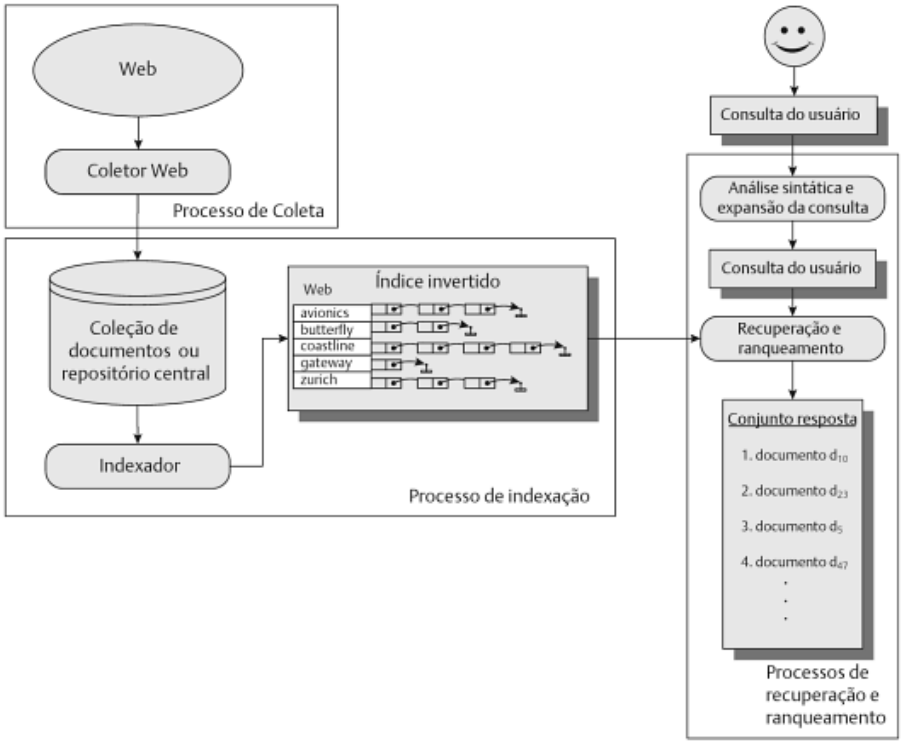

Figura 2.1: Arquitetura de alto nível do software de um sistema de RI. Fonte:Borges e Mourão, 2011. p. 7.

Porém, para recuperar as respostas de uma consulta, os termos de indexação associados a um documento não são igualmente úteis para descrever o conteúdo deste documento. É necessário decidir a importância de um termo para a descrição do conteúdo de um documento, porém não é uma tarefa fácil (FERNEDA, 2003). Para isso, é necessário implementar um algoritmo de ordenação dos documentos com a função de ranqueamento. Este algoritmo trabalha com premissas básicas sobre a relevância de um documento.

Para Baeza-Yates and Ribeiro-Neto (1999) um modelo de recuperação de informação é uma quádrupla $< D, Q, F, R(q_i, d_j) >$, onde:

- D é um conjunto de representações lógicas dos documentos;
- Q é um conjunto de termos para a consulta da necessidade do usuário;
- F é o arcabouço para as modelagens dos documentos, consultas e relações;
- $R(q_i, d_j)$ é a função de ranqueamento que associa à representação de uma consulta q_i

e é representação de um documento d_i um número real.

Para construir o modelo é necessário definir as representações lógicas de um documento e as necessidades do usuário. Em seguida, é feito um arcabouço para modelagem das representações. Os modelos clássicos, utilizados no processo de RI são os modelos booleano, vetorial e probabilístico. Além dos modelos clássicos, modelos mais avançados de recuperação de informação são propostos ao longo dos anos, dentre estes, destacam-se os modelos baseados em *links*.

Modelo Booleano

O modelo booleano é baseado na teoria dos conjuntos e na álgebra booleana (BORGES; MOURÃO, 2013). Aqui, os termos de indexação são considerados como pertencentes ou não nos documentos, fazendo com que as frequências de termos da matriz por documentos seja toda binária.

Dada um conjunto de termos de uma consulta Q e um conjunto de documento considerados relevantes para Q, esta consulta é constituída por termos de indexação ligados pelos conectivos booleanos: not, and e or. Portanto, os documentos recuperados contém os termos satisfeitos de uma expressao booleana convencional.

Os principais problemas do modelo Booleano é a ausência de um ranqueamento, pois todos os documentos são considerados somente como presentes ou ausente, o que pode interferir na recuperação dos documentos, tendo respostas muito grandes ou nulas. A vantagem do modelo é a sua simplicidade, utilizando pesos binários para os termos de indexação.

Modelo Vetorial

O modelo vetorial "baseia-se na comparação parcial entre a representação dos documentos e a da consulta do usuário" (KURAMOTO, 2002) atribuindo peso tanto aos termos da expressão de busca como aos termos de indexação que representam os documentos. Este modelo apresenta uma resposta mais eficiente em relação ao modelo booleano em relação à qualidade da resposta, pois os documentos são ordenados de forma decrescente de acordo com o grau de similaridade.

Dessa forma, as consultas (q) e documentos (dj) são representadas por vetores de pesos que especificam o tamanho e a direção de seu vetor de representação. Então, utiliza-se o

cosseno do ângulo formado por estes vetores para calcular o grau de similaridade do documento *d* com a consulta *q*. é dado pela Equação 2.1:

$$sim(dj, q) = \frac{\sum_{i=1}^{t} w_{i,j}.w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2}.\sqrt{\sum_{i=1}^{t} w_{i,q}^2}} \tag{2.1}$$

Os pesos $w_{i,j}$ e $w_{i,q}$ denotam a relevância de cada termo aos documentos ($w_{i,j}$) e às consultas ($w_{i,q}$) no espaço vetorial. A similaridade de $sim(dj, q)$ varia entre 0 e 1 pois $w_{i,j} \geq 0$ e $w_{i,q} \geq 0$. Dessa forma, diferente do modelo booleano, o modelo vetorial ordena pelo grau de similaridade dos documentos.

Para o cálculo dos pesos $w_{i,j}$ e $w_{i,q}$ utiliza-se uma técnica, *Inverse Document Frequency* (IDF) (SPARCK JONES, 1972), que faz o balanceamento entre o número de ocorrências do termo no documento e o número de documentos onde o termo aparece. Este cálculo é dado pela Equação 2.2 (SPARCK JONES, 1972).

$$w_{i,q} = (1 + log f_{i,q}).log\left(\frac{N}{n_i}\right) \tag{2.2}$$

Ou seja, o peso é proporcional ao produto da frequência do termo no documento e inversalmente proporcional á frequência na coleção. Dessa forma, o modelo vetorial é uma boa estratégia de ranqueamento para coleções genéricas, sendo simples e rápido.

Modelo Probabilístico

O modelo probabilístico, proposto em 1975 por (ROBERTSON; SPARCK JONES, 1988), apresenta os documentos como pesos binários que representam a presença de termos. Neste modelo, após realizada a consulta do usuário, existe um conjunto de documentos em que todos são considerados como relevantes. Este conjunto de documentos é considerado como conjunto ideal, e por meio desse conjunto, é possível recuperar os documentos relevantes para a consulta realizada. No entanto, essas características não são conhecidas quando a consulta é realizada, precisando de uma estimativa inicial para gerar uma descrição probabilística preliminar do conjunto ideal, e então recuperar um conjunto de documentos iniciais (BORGES; MOURÃO, 2013).

Assim, dada uma consulta q, o modelo probabilístico atribui a cada documento d_j uma razão P onde $\frac{^{d_j}\ \text{relevante a}\ ^{q}}{P(d_j\ \ \text{nao relevante a}\ \ q)}$, que computa a similaridade do documento d_j com

a consulta q. O teorema de Bayes é a ferramenta principal para o modelo probabilístico.

No modelo probabilístico, a consulta q é um subconjunto dos termos de indexação. O documento d_j é dado por um vetor de pesos binários, $d_j = (w_{1,j}, w_{2,j}, \ldots, W_{i,j})$ onde $w_{i,j} = 1$ se o termo está presente no documento e $w_{i,j} = 0$ se está ausente. R é dado como o conjunto ideal e $\bar{R}$ como complemento (documentos não relevantes). Dessa forma, a similaridade $sim(d_j, q)$ de um documento d_j para uma consulta q é dada pela Equação 2.3:

$$sim(d_j, q) = \frac{P(R|\vec{d_j}, q)}{P(\bar{R}|\vec{d_j}, q)} \tag{2.3}$$

Utilizando a regra de Bayes:

$$sim(d_j, q) = \frac{P(R|\vec{d_j}, q) \cdot P(R|q)}{P(\bar{R}|\vec{d_j}|q) \cdot P(\bar{R}|q)} \tag{2.4}$$

Onde, $P(d_j|R)$ é a probabilidade de selecionar aleatoriamente o documento d_j entre os documentos relevantes e $P(R|q)$ é a probabilidade de um documento selecionado aleatoriamente da coleção ser relevante. Assim como $P(\bar{R}|\vec{d_j}|q)$ e $P(\bar{R}|q)$ são semelhantes e complementares.

Como $P(R|q)$ e $P(\bar{R}|q)$ são iguais para todos os documentos, pode-se:

$$sim(d_j, q) \sim \frac{P(R|\vec{d_j}, q)}{P(\bar{R}|\vec{d_j}|q)} \tag{2.5}$$

A principal vantagem do modelo probabilístico é a ordem em que os documentos são ranqueados, pela ordem decrescente de sua probabilidade. Porém, alguns fatores afetam o desempenho, como a separação aleatória dos documentos, em relevantes e não relevantes e não utiliza uma frequência na qual o termo de indexação ocorre no documento.

Modelos baseados em *Links*

Na web, devido ao grande número de documentos (páginas *web*), é necessário considerar os *links* entre as páginas e não só o conteúdo do texto como os modelos anteriores. Assim, em modelos baseados em *links* são utilizados algoritmos baseados em *hyperlinks* para calcular a relevância de um documento de uma coleção. Particularmente são utilizados dois modelos, *PageRank* (BRIN; PAGE, 1998) e HITS (KLEINBERG, 1999).

PageRank O algoritmo PageRank, desenvolvido por (BRIN; PAGE, 1998) em 1996, utiliza a estrutura de links da web para calcular a importância de um documento em uma coleção. *PageRank* (PR) é a probabilidade de uma página web ser visitada por um usuário aleatoriamente por meio de uma página inicial que a referencia. Dessa forma, para cada página web, um valor, entre 0 e 1, é pré-calculado.

A fórmula para cálculo do *PageRank* é dada pela Equação 2.6:

$$PR(A) = (1-d) + d(\frac{PR(T_1)}{C(T_1)} + \frac{PR(T_2)}{C(T_2)} + \ldots + \frac{PR(T_n)}{C(T_n)}) \tag{2.6}$$

onde:

- A é um documento (página web);
- $T_{1,2,\ldots,n}$ são os documentos que contém em suas estruturas *links* para A;
- PR(A) é o *PageRank* do documento A e é calculado usando um algoritmo iterativo ;
- C(T) é o número de links que T apresenta;
- d é um fator de amortecimento, ajustado entre 0 e 1.

Como vantagem, o algoritmo PageRank apresenta o resultado de uma consulta com rapidez e consegue um desempenho melhor que outros modelos por manipular um conjunto adicional de informações para os documentos, através dos *links*.

HITS Segundo Kleinberg (KLEINBERG, 1999), páginas web podem ser classificadas em duas classes: autoridades e *hubs*(interligam as autoridades). O algoritmo *Hyperlink Induced Topic Search* (HITS) identifica, simultaneamente, os dois tipos de páginas. Para (KLEINBERG, 1999) o algoritmo HITS assume duas premissas:

- Se documento d_1 possui *hyperlink* para o documento d_2 então d_2 é considerado relevante para d_1;
- Se d_1 possui ligações para diversos documentos de alta relevância, d_1 torna-se valioso e se apontar para d_2 sugere que d_2 também tem relevância para o tema.

No algoritmo HITS, o ranqueamento da página *web* é decidido através da análise dos conteúdos textuais em uma determinada consulta, esse conjunto inicial se chama conjunto raiz. Após a coleta, o algoritmo expande o conjunto raiz adicionando os documentos que são referenciados ou referenciam os documentos que se encontram no conjunto raiz.

Após a formação dos dois conjuntos, o algoritmo HITS associa a cada documento d_x encontrado um peso de *hub* $h(d_x)$ e um peso de autoridade $a(d_x)$. E, de modo iterativo, atualiza os pesos de cada documento, utilizando as Equações 2.7 e 2.8:

$$a(d_i) = \sum_{d_j \to d_i} h(d_j) \tag{2.7}$$

$$h(d_i) = \sum_{d_i \to d_j} a(d_j) \tag{2.8}$$

A equação $d_i \to d_j$ denota que o documento d_i possui *hyperlink* para o documento d_j.

Como vantagem, o algoritmo HITS se destaca na pontuação de páginas *hubs* e autoridades relevantes de acordo com a string de consulta, além de que o ranqueamento pode utilizar outros rankings baseados em RI. Porém, o algoritmo calcula o *ranking* de cada página para a consulta, o que pode levar mais tempo para obter a resposta, assim como pode ocorrer de encontrar páginas de autoridades e de *hubs* irrelevantes.

2.1.3 Busca da *Web*

A *World Wide Web* (WWW) é uma coleção de documentos *hypertextos* ligados entre si, as páginas web. Uma página web possue as seguintes características:

- Endereçamento conhecido como *Unifom Resourse Locator* (URL);
- Um protocolo de transferência, o *Hypertext Transfer Protocol* (HTTP) que permite que um programa do usuário requisite uma página, por meio de sua URL, ao computador onde esta página está localizada;
- Uma linguagem padrão para especificar as estruturas da páginas web, como a *Hipertext Markup Language* (HTML);

A *web*, baseada em páginas HTML, pode apresentar um agrupamento conceitual de páginas web, por meio de *tags* que especificam URLs de outras páginas, sendo, dessa forma,

constituídos os *links*. Uma página P_1 pode apresentar um *link* para a página P_2 quando P_2 é relevante para o assunto de P_1. P_1 também pode apresentar uma referência a uma página P_3 se esta trata um tópico especial do assunto tratado por P_1. Ou seja, as URLs criam uma rede de citações elaboradas por assunto, autor, entre outros.

Mecanismos de Busca

Com o crescimento da Internet, recuperar informações da *web* tornou-se um problema devido à explosão de publicações. Dessa forma, para amenizar o problema, foram criadas ferramentas com o propósito de localizar recursos informacionais. Esses mecanismos, chamados de *search engines*, sites de busca ou portais, que, por meio de uma consulta de um usuário, recuperam uma lista de endereços de páginas *web* que são relevantes á consulta apresentada.

Os mecanismos de busca foram desenvolvidos especificamente para a *Web*, fornecendo serviços de recuperação de informação, como localização de páginas na *Web* que contenham informações específicas.

Algumas características são esperadas em um mecanismo de busca na Web para que estes respondam de forma efetiva ao desafio de localizar na Web a informação desejada pelo usuário. Para (HU, 2001) os requisitos para um mecanismo de busca identificados são:

- A eficácia do sistema para localizar e classificar os documentos;
- A eficiência de algoritmos de busca e classificação de documentos *web*;
- Acesso de sistemas não tendenciosos as páginas;
- Resultados úteis e expressivos;
- Atualização constante das informações *Web*.
- Cobertura do sistema;
- Adaptação do sistema às consultas dos usuários.

Diversas tecnologias têm sido propostas. são classificadas em seis categorias (HU, 2001):Exploração dos *hyperlinks*, recuperação de informação, metabuscadores, baseado em SQL, pesquisa por conteúdo multimídia e outras.

2.2 Modelo de linguagem

Modelo de linguagem foi originalmente desenvolvido para o problema de reconhecimento de voz definido como um modelo estatístico de sequência de palavras. Tornou-se um dos tópicos mais importantes na área de processamento de linguagem natural (MARTIN; JURAFSKY, 2000), tais como em sistemas de reconhecimento de voz modernas (MARTIN; JURAFSKY, 2000), de tradução automática(KOEHN, 2007), correção ortográfica (PIRINEN; LINDÉN et al., 2010) e previsão de palavras (TRNKA, 2006). Um modelo de linguagem consiste num conjunto finito X e uma função $p(x_1, x_2, ..., x_n)$, tais que utilizando a regra da cadeia da probabilidade para a frase inicial $X = x_1x_2x_3...x_n$, pode-se definir:

$$P(x_1, x_2, ..., x_n) = P(x_1)P(x_2|x_1)...P(x_n|x_1...n-1) \tag{2.9}$$

A Equação 2.9 pode ter um custo computacional alto ao gerar um modelo de linguagem quando lida com um grande número de possibilidades de sentenças.

Para gerar um modelo de linguagem com um custo computacional menor utiliza-se os modelos n-grama, onde somente as n palavras mais recentes da história são utilizadas para condicionar a probabilidade da próxima palavra. Pode-se utilizar o modelo bigrama, onde aproxima o contexto para uma única palavra anterior à palavra atual, ou o modelo trigrama que utiliza as duas últimas palavras anteriores á palavra para aproximar o contexto. O desenvolvimento a seguir refere-se ao caso particular de gramáticas bigrama, a probabilidade de uma sequência de palavras torna-se:

$$P(x_n|x_(n-1)) \tag{2.10}$$

Para estimar as probabilidades do bigrama ou n-grama, utiliza-se o modelo de Estimativa de Máxima Verossimilhança (MLE), as Equações 2.11 e 2.12 apresentam um exemplo de MLE para um n-grama:

$$P(w|c) = \frac{C(w)}{C(c)} \tag{2.11}$$

$$P(c_n|c(n-1)) = \frac{C(c_{n-1}, c_n)}{\sum_c C(c_{n-1}, n_i)} \tag{2.12}$$

Na comparação entre modelos de linguagem, é importante ser capaz de quantificar a dificuldade que estes impõem à uma determinada tarefa. Os modelos de linguagem tendem a minorar as incertezas do conteúdo das sentenças e facilitar o reconhecimento. Ou seja, se existem poucas palavras que podem seguir uma dada palavra em um modelo de linguagem, o sistema de reconhecimento terá menos opções para verificar se o desempenho será melhor do que se existir muitas palavras possíveis.

Dessa forma, antes de usar um modelo de linguagem, é necessário avaliá-lo em um sistema de reconhecimento e determinar qual deles fornece a menor taxa de erro. Uma métrica que geralmente é utilizada independente da tarefa é a Perplexidade. A Perplexidade é o inverso da probabilidade do conjunto de teste (segundo um modelo), normalizada pelo número de palavras. As Equações 2.13 e 2.14 apresentam a perplexidade de um modelo de linguagem P, dado um conjunto de teste.

$$Perplexidade(P, W) = P(W)^{(\frac{-1}{m})} \tag{2.13}$$

$$Perplexidade(P, W) = \sqrt[m]{\prod_{i=1}^{m} \frac{1}{P(w_i|w_{i-1}, ..., w_{i-n+1})}} \tag{2.14}$$

Quanto menor o valor da perplexidade, melhor é o modelo de linguagem para o determinado conjunto de testes. E minimizar a perplexidade é o mesmo que maximizar a probabilidade. Quando um corpus não é bom o suficiente para gerar um modelo de linguagem, combinam-se vários *corpora* utilizando interpolação linear.

2.3 *Web Crawler*

Buscas na *Web* tem um papel muito importante para os usuários, onde os mesmos tendem a procurar temas de seu interesse enquanto navega na *web*, o que faz com que aumente o número de motores de busca, por exemplo o *Google* e *Bing*. Esses motores de busca coletam uma grande coleção de documentos agregados e fornecem métodos para consultá-los. Dessa forma *Web Crawlers*, também chamados de *Robot* ou *Spider*, são sistemas de transferência de páginas *web* para serem usadas em motores de busca (OLSTON; NAJORK, 2010). O resultado de um *Crawler* é uma coleção de páginas *web*.

A *web* não é um repositório centralmente gerenciado, é composta de milhões de provedores e conteúdos independentes, onde cada um oferece os seus próprios serviços e competem uns com os outros. Fazendo com que os motores de busca tenham que se intensificar para manter-se com a crescente quantidade de informações disponíveis na rede (ELYASIR; ANBANANTHEN, 2012). A web pode ser vista como um repositório de informações, mantidas por um conjunto de protocolos e formatos de dados. Os agregadores de conteúdo tem duas opções:

- Um modelo que vasculhe a web para obter informações novas ou atualizadas;
- Estabelecer uma convenção e um conjunto de protocolos que permitam que os provedores de conteúdo enviem conteúdo de interesse para os agregadores;

Um algoritmo de um *web crawler* é baseado em um conjunto de URLs sementes em que o *crawler* baixa todas as páginas *web* indexadas por essas URLs. Após, é feita a extração de todos os links contidos nessas páginas e de forma interativa faz o *download* de todas as páginas abordadas por estes novos *links*.

Porém, sistemas de *web crawler* ultimamente têm que lidar com grandes desafios (OLSTON; NAJORK, 2010):

- Escala: a *web* tornou-se grande e está em constante evolução, um crawler de ampla cobertura deve alcançar um alto desempenho e tem complicações em sua implementação;
- Seleção de conteúdo: um *crawler* não rastrea pela web completa, o rastreamento ocorre seletivamente e em uma ordem controlada;
- Obrigações sociais: um *crawler* deve ter planejamento para não sobrecarregar os sites que o compõe;
- Adversários: alguns provedores apresentam conteúdo inútil ou enganoso no *corpus* pelo *crawler*.

Crawlers podem ser classificados de acordo com sua maneira de rastrear as páginas *web*, que está relacionado com a aplicação final que ele servirá. Em (ELYASIR; ANBANANTHEN, 2012) são classificados de acordo com a sua funcionalidade e podem ser *crawler standard*

e *focused crawler*. *Crawler standard* tem um comportamento aleatório para a coleta das páginas *web*, enquanto *focused crawler* tem uma maneira orientada para fazer o rastreamento das páginas.

2.3.1 Arquitetura Básica de um Crawler

O funcionamento básico de um *web crawler* inicia com um conjunto de URLs chamadas "sementes". Para cada URL, o *crawler* analisa a página, extraindo o seu texto e todos os links existentes e coloca eles numa fila, a fronteira de URLs, que consiste em todos os links de páginas que ainda vão ser analisadas pelo rastreador. Esse processo se repete até ser imposto um critério de parada.

Este processo é apresentado na Figura 2.2, que mostra uma arquitetura de um *Web Crawler* genérico.

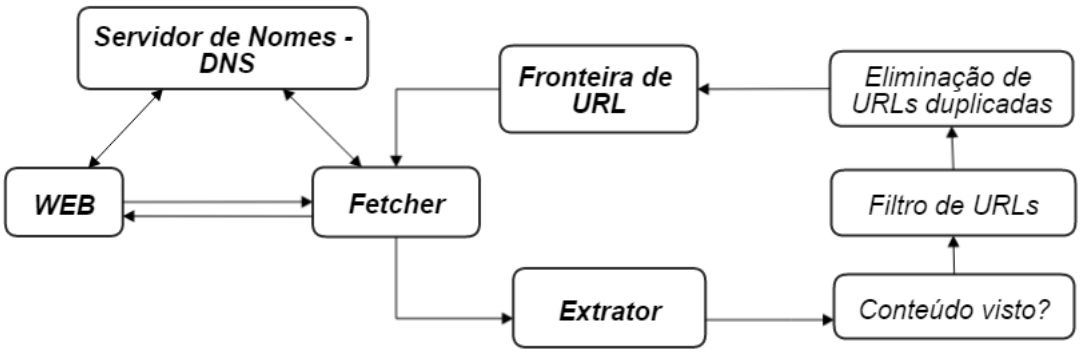

Figura 2.2: Arquitetura de alto nível de um *Web Crawler*. Fonte: Elaborada pelo autor

Inicialmente, as informações disponíveis em páginas *web* são coletadas e indexadas e, em seguida, escolhe uma URL da fronteira. A fronteira URL é uma estrutura de dados que guarda e gerencia URLs que ainda vão ser obtidas no rastreamento atual.

Nesta etapa, geralmente é implementado uma *Fist In Fist Out* (FIFO), que resulta em um percurso em largura de um gráfico *web*. Porém , deve-se considerar duas desvantagens: as páginas de alta qualidade mudam frequentemente e devem ser priorizadas por cada iteração e a maioria dos *links* na *web* ligam outras URLs que estão no mesmo *host*. Usar uma fila

de prioridade simples pode resultar um diversas solicitações em um mesmo *host*, trazendo problemas de polidez (MANNING; RAGHAVAN; SCHüTZE, 2008).

Web Crawlers também podem priorizar a ordem em que se encontram as URLs na fronteira. Segundo (OLSTON; NAJORK, 2010) pode ser necessário priorizar as páginas por meio do seu *PageRank*, pelo tráfego que recebem, reputação do *site* ou até mesmo a atualização da página durante o rastreamento.

O *crawler*, em seguida, escolhe o *fetcher*. O *fetcher* primeiro chama a módulo servidor de nomes. Cada URL contém um componente de host. O mecanismo que consiste em encontrar o endereço IP que corresponda ao nome deste host é chamado de *Domain Name Service*(DNS). O módulo de servidor de nomes - DNS atende requisições DNS dos coletores, determinando qual servidor *web* busca uma página especificada por uma URL. Este módulo mantém um cache dos identificadores DNS resolvidos e evita que se façam requisições de DNS remotas.

Em seguida, o *fetcher* utiliza o protocolo HTTP para recuperar a página *web* a partir de uma URL. Se o *download* for bem sucedido, a página pode ou não ser armazenada em um repositório de páginas web. Nos dois casos, a página é passada pelo módulo de extrator de links. Este módulo analisa o conteúdo HTML e extrai o texto e os *hyperlinks* contidos nele.

As URL passam pelo módulo do filtro de URL e pela Eliminação de URLs duplicadas. Nesta etapa são excluidos as URLs que ja foram visitadas e coloca na Fronteira de URL as que ainda não foram visitadas.

2.3.2 Métodos de busca e métricas

Métodos e métricas de buscas são utilizados para atender a necessidade de saber quais páginas o *crawler* deve indexar e quais não para a fronteira de URLs.

A métrica *First-In-First-Out* (FIFO) produz uma coleta de páginas URLs mais abrangente, ela visa coletar uma grande quantidade de sites (Algoritmo 1).

Algoritmo 1: Crawler tradicional

início

Seja I uma lista de URLs sementes;

Seja F a Fronteira;

para *cada* $URLi \in I$ **faça**

$Enqueue(i, F)$;

fim

repita

$u \leftarrow Dequeue(F)$;

$p \leftarrow Get(u)$;

Store p;

Extrair os hyperlinks de p; Seja U o conjunto de URLs citadas em cada hyperlinks; **para** *cada* $URLu \in U$ **faça**

Enqueue(u,F);

fim

até *not Empty(F)*;

fim

As operações *enqueue*, *empty* e *dequeue* são mais frequentes na FIFO. O procedimento *enqueue* acrescenta uma nova URL para a fila, se esta não tiver a URL encontrada. O *dequeue* não remove a URL da fila permanentemente, ele apenas coloca uma marca na URL para comprovar que ela foi removida mas deve permanecer para futuras consultas. Finalmente, o procedimento *empty* só é considerado verdade quando todas as URLs forem removidas (Assis, 2008).

Outra métrica de ordenação é a *breadth-first search* (BFS) (NAJORK; WIENER, 2001), que visita um número maior de locais produzindo um rastreamento mais eficiente. O algoritmo inica no nó raiz e percorre todos os nós vizinhos no mesmo nível. Se o objetivo inicial for alcançado, então termina a busca e retorna sucesso. Caso não seja alcançado, continua o processo de busca para o próximo nível até que alcance o objetivo.

Depth First Search(DFS) é uma métrica utilizada que usa a estrutura de pilha para a ordenação de URLs não visitadas. Esse algoritmo inicia no nó raiz e atravessa profundamente a árvore para os seus filhos. Se tiver mais de um filho, DFS dá prioridade ao filho mais a esquerda. Em seguida, recua para o próximo nó não visitado até finalizar o objetivo inicial.

No entanto, para (ESTER; KRIEGEL; SCHUBERT, 2004) é necessário um ordenamento da fila de URLs não visitadas, em que priorize as URLs mais importantes. Com isso, resultados mostram que o algoritmo *PageRank* (Seção 2.1.2) é uma métrica de ordenação de URLs que apresenta resultado excelentes com páginas que tenham *PageRank* elevado.

2.4 *Focused Web Crawler*

Existem diversas formas para a construção de um Web crawler, onde cada uma delas é dependente do uso ou das informações que serão obtidas. Uma delas é denominada por *Focused Crawler*, uma técnica apresentada por (CHAKRABARTI; BERG; DOM, 1999b), onde descreve um *crawler* que busca, adquire, indexa e mantém páginas em um conjunto específico de temas em um subconjunto restrito da *web*. Com o fim de economizar recursos de *hardware* e de rede, o *focused crawler* analisa as páginas rastreadas para encontrar *links* que são indicados para ser mais relevantes para o rastreamento e ignorar aquelas que são irrevelantes para a *web*.

A diferença do *focused crawler* está na forma como as URLs sementes são apresentadas pelo usuário. O *crawler* percorre o mesmo caminho padrão, porém só recolhe páginas semelhantes entre si baseadas em um domínio. Para executar uma instância específica, a entrada humana pode ser fornecida em duas formas: a primeira é selecionar e refinar nós dos tópicos específicos na taxonomia, e a segunda é fornecer exemplos de URLs adicionais que sirvam como pontos de partida para o rastreamento.

Assim, o conteúdo da *web* pode ser gerenciado por uma equipe de *focused crawler* distribuídos, onde cada um pode ser especializado em um ou mais tópicos de interesse. Cada *focused crawler* tem chance maior de detectar alterações nas páginas dentro do seu tema em toda a *web*.

2.4.1 Arquitetura de um *Focused Web Crawler*

A Figura 2.3 apresenta uma arquitetura proposta por (CHAKRABARTI; BERG; DOM, 1999a), onde um *focused web crawler* tem três componentes principais: um classificador que realiza as classificações de relevância nas páginas que foram rastreadas para decidir sobre a expansão de um *link*, um destilador que determina uma medida de prioridade entre as páginas

indexadas para o *crawler* conhecer a ordem de visitação e um *crawler* reconfigurável que governa o destilador e o classificador.

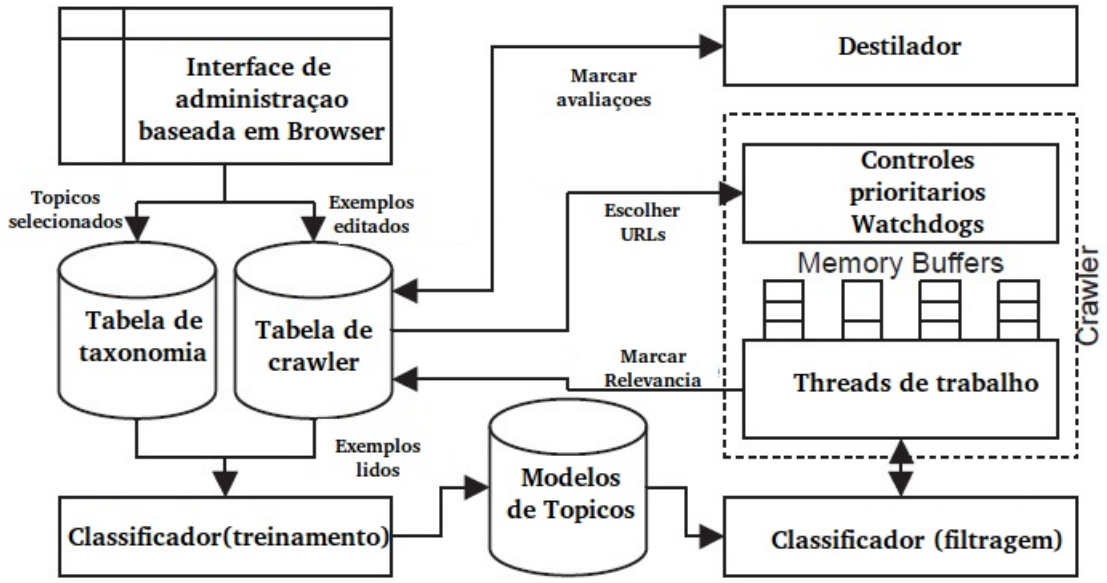

Figura 2.3: Arquitetura de um *focused crawler* adaptada de Chakrabarti, Berg and Dom 1999

Dessa forma, é dado um grafo dirigido G cujos nós são distribuídos fisicamente e G é a *web*, onde um custo é dado para cada visita a um vértice de G. Também é dado um diretório C com hierarquia de tópicos em forma de árvore, onde cada nó de C refere-se a algumas páginas de G relacionadas a determinado tópico. Os exemplos relacionados ao tópico c são dados como $D(c)$. Estas páginas podem ser pré-processadas e caracterizadas por um subconjunto de temas $C^* \subset C$ que são marcados como **bom**. Então, dada uma página *web* q, o sistema especifica como calcular a relevância de q com respeito a C^*.

O rastreamento inicia visitando todas as páginas em $D(C^*)$. Em cada passo, o sistema pode visualizar o conjunto de páginas visitadas V e, em seguida, visitar uma página não visitada que esteja na fronteira de URLs. Ou seja, o *focused crawler* tem o objetivo de maximizar a relevância média das páginas *web*.

Para (CHAKRABARTI; BERG; DOM, 1999b) a relevância de uma página é um sinal de boa relevância para as páginas vizinhas, o que faz com que o *focused crawler* utilize o módulo classificador. Em seguida, várias citações de um único documento podem significar documentos semanticamente relacionados, o que leva a utilizar o modulo destilador para

identificar páginas com um grande número de links até às páginas relevantes.

2.4.2 Abordagens baseadas em estrutura de Links

O desempenho de um *focused crawler* depende da seleção das URLs sementes. Geralmente, os usuários fornecem um conjunto de URLs como entrada para o *crawler* ou estas são selecionadas entre as respostas fornecidas por um motor de busca na *web* usando palavras-chaves como consulta. Deve-se atentar que boas páginas sementes podem ser tanto as páginas relevantes para o tema como páginas que levem à URLs relevantes.

A abordagem *Fish-Search* (BRA, 1994), baseada na *depth first*, atribui às páginas candidatas valores binários (1 se for relevante e 0 para não relevante). A relevância de um documento é dada baseada na ocorrência de palavras-chaves em seu conteúdo. Ou seja, todas as páginas relevantes tem a mesma prioridade.

O método *Shark-search* (HERSOVICI, 1998) é uma melhoria da abordagem *Fish-Search*. Enquanto que a abordagem *fish-search* concentra a sua busca nas regiões em que as páginas são consideradas relevantes, o método *shark-search* utiliza uma função com valor contínuo para medir a relevância de uma página. Esta função utiliza *Vector Space Model*(VSM) para atribuir valores não binários para as páginas candidatas. Para cômputo dos valores são priorizados o conteúdo da página *web*, o texto âncora, o texto em torno das ligações e valor de prioridade de páginas pai.

Outras abordagens adicionais ao *focused crawler* é o *InfoSpider* e o *Best-First crawler*. O *InfoSpider* (MENCZER, 1997) utiliza redes neurais para decidir qual *link* deve seguir. Enquanto que o *Best-First* utiliza vetores de *term frequency* (tf) para computar a relevância de um tema.

Capítulo 3

Trabalhos Relacionados

Este capítulo apresenta os trabalhos relacionados à área de construção do *Focused Web Crawler*, bem como também sobre classificação de texto e da web e especificamente as propostas de algoritmos para resolver o problema de recuperação de informação com o rápido crescimento de dados na web. Aqui são apresentados trabalhos recentes sobre construção de *focused web crawler* (Seção 3.1) e criação de *Corpora* na língua Portuguesa (Seção 3.2).

3.1 Focused Web Crawler

Esta seção apresenta os trabalhos relacionados com a construção de motores de busca baseados em um gênero específico, os denominados *Focused Web Crawlers*.

3.1.1 *Best first*

Como apresentado na Seção 2.4 o desempenho de um focused crawler depende da seleção das URLs sementes. Geralmente, os usuários fornecem um conjunto de URLs como entrada para o crawler ou estas são selecionadas entre as respostas fornecidas por um motor de busca na web usando palavras-chaves como consulta.

Uma das técnicas é a *best first* onde a ideia básica é que, dada a fronteira de URLs, a melhor URL dada por um determinado critério é selecionada para o rastreamento. Os principais trabalhos que utilizam esta técnica são propostos por (AGGARWAL; AL-GARAWI; YU, 2001), (BATSAKIS; PETRAKIS; MILIOS, 2009), (DILIGENTI, 2000) e (CHAKRABARTI; BERG;

DOM, 1999b).

O modelo proposto por Aggarwal, Al-Garawi e Yu (2001), chamado de *intelligent crawling*, procura por meio de características específicas na web, páginas que apontem para determinado tema de estudo. Para isto, são consideradas diversas características para ordenação do rastreamento de URLs, que incluem a importância do conteúdo da página web na URL candidata, se as URLs contém *tokens* sobre o tema e se contém informações de grande relevância, se os links de uma página candidata satisfazem ao predicado proposto e se os irmãos de uma página web já tenham sido classificados como pertencentes ao tema.

A entrada do modelo consiste em um conjunto de características e uma estrutura de links da web que já foram rastreados e a saída corresponde a uma lista de prioridade que determina a ordem de visita das URLs. O algoritmo funciona através do rastreamento de páginas web de acordo com uma ordem de prioridade específica baseada nas informações de autoaprendizagem das páginas web. A cada iteração do algoritmo uma tabela é mantida contendo as informações correspondentes a cada candidato e as prioridades são reajustadas. Só finaliza quando a lista de prioridade estiver vazia.

Uma contribuição importante do trabalho de (AGGARWAL; AL-GARAWI; YU, 2001) foi a informação **K** de auto-aprendizagem das páginas web que se refere a:

- número de URLs rastreadas;
- número de URLs rastreadas que satisfazem o predicado;
- O número de páginas rastreadas em que uma determinada palavra i ocorre;
- O número de páginas rastreadas em que uma determinada palavra i ocorre e que satisfaz ao predicado;
- O número de páginas rastreadas em que o símbolo i ocorre em sua URL;
- O número de páginas rastreadas em que o símbolo i ocorre e que satisfaz ao predicado;
- O número de ligações rastreadas;

Quando estas informações são coletadas, os valores dos principais parâmetros são estimados e ocorre o rastreamento das páginas web.

Aggarwal, Al-Garawi e Yu (2001) propõem uma técnica mais geral que um *focused crawler*, onde rastreia páginas que satisfaçam aos predicados definidos pelo usuário e que possa reutilizar os conhecimentos adquiridos anteriormente para fornecer rastreamentos mais eficientes, por isso o nome *intelligent crawling*. O modelo usa uma alternativa de *focused crawler* em que o crawler possa aprender gradualmente a estrutura de links estatisticamente enquanto rastreia.

Apesar de mostrar eficácia no rastreamento, pesquisas por características no texto são dependentes de linguagem e pode tornar o rastreamento difícil de aplicar em situações em que a coleção imposta seja dada em vários idiomas.

O trabalho de Batsakis, Petrakis e Milios (2009) apresenta questões relacionadas com a concepção e implementação de um *focused crawler*. Muitos trabalhos abordam um *focused crawler* estimando a relevância de uma página web através do seu conteúdo e da análise de seus links. Porém, este trabalho apresenta uma nova abordagem de um *focused crawler* de aprendizagem utilizando uma proposta de *Crawlers* baseados em *Hidden Markov Model* (HMM) (LIU; JANSSEN; MILIOS, 2006) para ordenação de páginas relevantes.

O *Crawler* baseado em HMM funciona através de uma relação entre o conteúdo de uma página com o caminho que leva até as páginas relevantes. O conjunto de treinamento se dá com um conjunto de páginas sementes rotuladas como relevantes ou não. As páginas relevantes formam um *cluster* (C0) e as não-rotuladas são agrupadas por K-*means*, formando um grupo C1 para Ck (k definido pelo usuário). Um HMM é criado baseado neste agrupamento, onde cada página tem dois estados: (a) é o estado correspondente ao *cluster* que a página pertence e (b) é o estado que corresponde é distância a partir de uma página. As páginas web são caracterizadas por seus níveis ou estados escondidos L_i, (onde i é o nível) e pelo cluster C_j a que pertencem. O conjunto de estados escondidos das páginas e observações é modelado por um HMM. Dessa forma, a prioridade de uma URL é dada pela probabilidade em que um crawler irá conduzir a URL é uma página de destino.

É proposto que a pontuação da prioridade de uma página web seja definida como a média das prioridades dada pelo algoritmo HMM e pela similaridade do vetor de páginas relevantes com a representação de um vetor de termo da página. São propostas duas variações, usando somente o conteúdo da página ou usando o conteúdo da página e o texto âncora. Todos os *crawlers* apresentam um melhor desempenho quando uma combinação de conteúdo da

página e texto âncora são descritos na URL candidata.

O trabalho de (BATSAKIS; PETRAKIS; MILIOS, 2009) segue a estratégia de um *crawler* em que baixe somente as páginas que são relevantes para tópicos específicos. A partir dos resultados, notou-se um melhor desempenho quando o *focused crawler* foi melhorado por meio de ontologias específicas relacionadas aos tema pesquisado e não relacionados ao uso geral, como o *Wordnet*. Porém, deve-se atentar aos *crawlers* de aprendizagem, pois estes aprendem os padrões de rastreamento para levar até as páginas relevantes por meio de páginas não relevantes, o que aumenta a probabilidade de falha.

O trabalho de Diligenti (2000) propõe o método *Context Focused Crawler* (CFC) que modela os links e conteúdos dos documentos que estão ligados à página de destino para melhorar a eficiência em encontrar os documentos relevantes ao tópico escolhido.

O CFC usa o texto de uma página u para estimar a distância de link (número de percursos entre os links necessário para se deslocar de uma página a outra) para u. Esta representação é utilizada para treinar classificadores, que são otimizados para detectar e atribuir documentos em diferentes categorias.

O método consiste em derivar *back-links* para cada URL semente e estes constroem um gráfico de múltiplas camadas. Estes gráficos são usados para extrair os caminhos que levam até os nós relevantes, que são as URLs de destino. Os conteúdos mais relacionados aparecem no centro do gráfico, enquanto que nas camadas exteriores se encontram as URLs mais gerais. Quando o *crawler* descobre uma página com um conteúdo que esteja na superfície da hierarquia, usa o conhecimento sobre a estrutura do gráfico e realiza a busca das páginas relevantes.

Os classificadores utilizam indexação de palavras-chaves em cada documento usando uma modificação do TF-IDF para representar as características dos documentos. O TF-IDF de uma página w é computado pela Função 3.1:

$$v(w) = \frac{f^d(w)}{f^d_{max}} log \frac{N}{f(w)} \tag{3.1}$$

Onde $f^d(w)$ é o número de ocorrências de w em um documento d, f^d_{max} é o numero máximo de ocorrências de uma frase em um documento d, N é o número de documentos e $f(w)$ é o número de documentos onde a frase w ocorre ao menos uma vez. O algoritmo TF-IDF é modificado concatenando todos as URLs sementes em um único documento. Assim, são

removidas todas as *stop-words* e feita uma lematização das palavras em todo o documento. O TF-IDF é computado usando um *corpus* de referência de um *web crawler* geral.

Este trabalho apresenta um bom desempenho comparado a técnicas estudadas, como algoritmo BFS e um *Focused Crawler* que não utilize contextos. Utilizar recursos de níveis em URLs torna o trabalho importante para fóruns da web oculta pela necessidade de recolher conteúdos multimídias e estes, muitas vezes são armazenados em locais com alguns níveis de distância da URL de origem. Dessa forma, para se chegar a estes níveis é necessário uma abordagem baseada em regras.

O trabalho de Chakrabarti, Berg e Dom (1999b) descreve um *focused web crawler* com três componentes, um classificador para avaliar a relevância da página web com o tema escolhido, um destilador para identificar os nós relevantes e um *crawler* que governa o classificador e o destilador.

O *Focused Crawler* apresentado é o primeiro a utilizar um classificador, neste caso o Classificador Naive Bayes, para orientar o rastreamento. A ideia do *focused crawler* é classificar as páginas rastreadas dentro das categorias de uma mesma taxonomia com ajuda de documentos exemplos. Estes documentos são usados para criar o classificador Bayesiano, este é capaz de determinar a probabilidade $P(c|d)$ de um documento web d pertencer a uma categoria c da taxonomia, ou seja, se eles são classificados como páginas relevantes ou não relevantes.

Dessa forma, o usuário seleciona um conjunto de tópicos da taxonomia para ocorrer o rastreamento. O *crawler* segue, preferencialmente, os links das páginas que o classificador considera como mais relevantes para o tópico. Os links que pertencem a temas similares ao pesquisado também são seguidos. além disso, um módulo destilador tenta identificar as páginas *hub* utilizando o algoritmo de análise de links HITS modificado. Os links de páginas *hub* têm mais prioridade do que outros links.

De acordo com os resultados o *focused crawler* torna-se um paradigma efetivo pois, ao contrário de um *web crawler* geral, explora uma maior população de documentos de um determinado tema ao longo do tempo. Porém o trabalho faz um rastreamento por meio de documentos URLs exemplos e não utilizando palavras-chaves. Dessa forma, o *crawler* não se atenta em aplicar uma devida atenção à estratégias de ordenação de páginas para o processo de seleção de URLs sementes.

Coleções de documentos que tenham diversos grafos web, como páginas web com links e artigos científicos com gráficos de citação, são melhor analisadas quando tanto os conteúdos textuais como as ligações entre as páginas são utilizadas de uma forma unificada (ALMPANIDIS; KOTROPOULOS; PITAS, 2007).

Dessa forma, o trabalho de Almpanidis, Kotropoulos e Pitas (2007) apresenta um classificador que combina a análise de links com o conteúdo de textos, com o objetivo de recuperar e indexar os documentos da web de um determinado domínio. O algoritmo fornecido trabalha com a descoberta de recursos de informações no contexto de um motor de busca em tópicos da web quando não há um conhecimento prévio disponível da estrutura dos *links*, com o objetivo de superar as limitações impostas pela necessidade de ter que fornecer dados iniciais para o treinamento.

Foi avaliado um novo algoritmo chamado *Hypertext Content Analysis latent* (HCLA) que é um classificador *Latent Semantic Indexing*(LSI) (DEERWESTER, 1990). LSI assume que há uma estrutura básica ou latente no uso da palavra que está parcialmente escondida pela variabilidade na escolha de palavras. No trabalho de (ALMPANIDIS; KOTROPOULOS; PITAS, 2007) LSI combina a análise textual com a análise dos links utilizando um paradigma *Vector Space Model*(VSM). A tradicional *bag-of-words* do VSM é estendida para que cada documento seja representado por todos os termos que contém e todos os documentos de texto e hipertextos semelhantes.

LSI é uma técnica que analisa as relações entre os termos e conceitos que ocorrem em um texto. Utiliza uma técnica matemática chamada *Singular Value Decomposition* (SVD)(DEERWESTER, 1990) e tem capacidade para extrair o conteúdo conceitual do texto através da construção de associações entre os termos que tenham contextos semelhantes.

O trabalho de Almpanidis, Kotropoulos e Pitas (2007) propõe um algoritmo de um *focused crawler* que tem como objetivo superar as limitações impostas pela necessidade de fornecer uma grande quantidade de dados e mantém uma alta taxa de *precision*/*recall*. Através dos experimentos realizados por Almpanidis, Kotropoulos e Pitas (2007), HCLA supera os algoritmos *Breadth-First*, *BackLink*, *PageRank*, e duas extensões do *Shark-search*.

No entanto, o algoritmo requer alto processamento de energia e de recursos de memória. O desempenho do LSI é sensível ao tamanho do *corpus* treinado o que faz com que possa sofrer quando os dados disponíveis forem escassos. além disso, o classificador LSI não é

capaz de identificar as palavras-chaves.

3.1.2 Naive Bayes

O uso de *Naive Bayes* para o rastreamento utiliza a probablidade de que uma página com um determinado conjunto de características está no conjunto de páginas web relevantes para decidir qual a melhor página web para ser visitada a seguir. Isto é feito por meio de estatísticas de frequências das palavras nas páginas web relevantes e de frequências em que estas palavras aparecem nas páginas web gerais. Trabalhos como de (MCCALLUMZY, 1999), (BARBOSA; FREIRE, 2007), (DILIGENTI, 2000) e Chakrabarti, Punera e Subramanyam (2002).

A expansão da Internet e do número de seus utilizadores tem levantado muitos problemas novos em recuperação de informação e inteligência artificial. Coleta de informações da web é uma tarefa difícil para um usuário iniciante, mesmo que ele usa um motor de busca. Uma solução dada por McCallum (1999) é construir um motor de busca de domínio específico.

O trabalho descreve o projeto Ra que automatiza muitos aspectos para criação e manutenção de técnicas de aprendizagem.

Essas técnicas são utilizadas na criação de motores de busca direcionados a tópicos específicos, o sistema CORA, um motor de busca disponível para trabalhos de pesquisa na área de ciência da computação. O *spider* CORA inicia pelas *home page* de departamentos de informática e laboratórios reunindo links para documentos *PostScript*, Após é feito uma análise e pós-processamento dos documentos.

Em CORA os dados são organizados em hierarquias de tópicos, que é uma maneira eficiente de organizar e exibir grandes quantidades de informações. é criada uma estrutura de 70 folhas e são selecionadas apenas algumas palavras-chave associadas a cada nó.

A proposta alivia a carga sobre o classificador utilizando os dados não marcados, como palavras-chaves e hierarquias de classes. O construtor fornece algumas palavras-chave para cada categoria. Utilizando Expectation-Maximization (EM) para estimar rótulos de documentos não marcados e re-estimar os rótulos dos documentos marcados por palavras-chave.

CORA utiliza o classificador de texto Naive Bayes Multinomial. Para determinar a probabilidade de que um documento d_i pertença a classe c_j é dada a Equação 3.2, onde $w_{d_{ik}}$ é a palavra w_t que ocorre na K posição de um documento d_i.

$$P(c_j|d_i) \propto P(c_j) \prod_{k=1}^{|d_i|} P(w_{d_{ik}}|c_j) \tag{3.2}$$

Com isso, a formação de um classificador Naive Bayes padrão requer um conjunto de documentos, D, e seus rótulos de classe. A estimativa de frequência de palavras é dada pela Equação 3.3:

$$P(w_t|c_j) = \frac{1 + \sum_{d_i \in D} N(w_t, d_i) P(c_j|d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{d_i \in D} N(w_s, d_i) P(c_j|d_i)} \tag{3.3}$$

Onde, $N(w_t, d_i)$ é a quantidade de vezes em que a palavra w_t ocorre no documento d_i. $P(c_j|d_j)$ é um indicador de que o documento d pertence a classe c e v é o número de palavras do vocabulário.

Quando as classes são organizadas hierarquicamente, as estimativas dos parâmetros de Naive Bayes podem ser melhoradas utilizando a técnica de estatística *Shrinkage* (MCCALLUMZY, 1999) , onde novas estimativas dos parâmetros de frequência de palavras são calculados pela média ponderada das estimativas locais das classes e as estimativas de seus antepassados na hierarquia.

O trabalho mostra que as técnicas de aprendizado de máquina podem ajudar significativamente a criação e manutenção de motores de busca de domínio específico. E apresentou novas pesquisas de motores de busca utilizando aprendizado por reforço, classificação de texto e extração de informações.

Alguns problemas durante a visitação de páginas web ocorrem durante a etapa de recuperação de informação. Muitos estão relacionados a não obtenção do conteúdo "oculto" entre as páginas. Este conteúdo oculto é relacionado a páginas cujo conteúdo é gerado em tempo real, onde o usuário preenche on-line ou executa código *Active Server Pages* (ASP), *Java Server Pages* (JSP) ou *Hypertext Preprocessor*(PHP) no servidor. Essas páginas são classificadas como formulários (forms) e retornam conteúdo diferenciado em relação ao conteúdo preenchido pelo usuário.

Dessa forma, Barbosa e Freire (2007) apresentam um *framework* para que um *crawler* adaptativo aprenda a localizar automaticamente os bancos de dados da web oculta, apresentando um crawler focado a um determinado assunto. O *framework Adaptive Crawler for Hidden-Web Entries* (ACHE) é uma extensão do *framework Form Focused Crawler* (FFC)

abordando as limitações promissoras do FFC e apresentando uma adptação ao foco com a progressão do rastreamento.

Ou seja, dado um conjunto de formulários web, o *framework* ACHE tem o objetivo de identificar de forma eficiente e automaticamente outros fomulários no mesmo domínio. A figura 3.1 apresenta a arquitetura de alto nível do sistema ACHE.

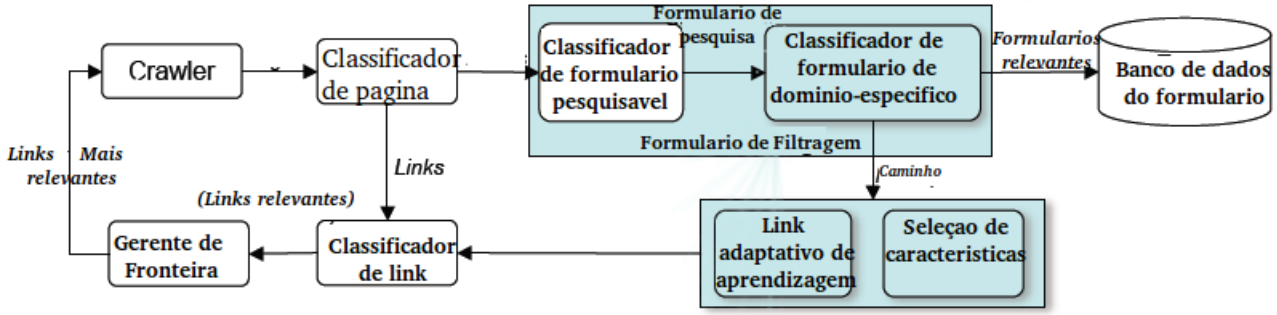

Figura 3.1: Arquitetura do *framework* ACHE Adaptada de Barbosa (2007)

ACHE utiliza um *focused crawler* para localizar as bases de dados on-line e também usa o conteúdo da página web para centralizar a sua pesquisa para um determinado assunto. Para lidar com a escassez de bases de dados *online* na Web, prioriza os links que são mais propensos a levar a formulários ao domínio do banco de dados procurado.

Além da arquitetura apresentada pelo *framework* FFC, ACHE é composto por mais dois classificadores: o *Searchable Form Classifier* (SFC) e o *Domain-Specific Form Classifier* (DSFC). Os formulários são processados pelos dois classificadores de forma sequencial. Cada formulário recuperado é classificado inicialmente pelo SFC como pesquisável ou não pesquisável. E, em seguida, o DSFC examina se os formulários pesquisáveis pertencem ao domínio.

ACHE também apresenta um componente, *link learner*, como elemento de aprendizagem. *Link learner* aprende de forma dinâmica as características extraídas automaticamente a partir de caminhos de sucesso pelos componentes de seleção de recursos e atualiza o *link classifier*.

O sistema ACHE proporciona uma solução escalável para o problema de construção automática de coleções de banco de dados. Ele pode ser personalizado para diferentes domínios.

Porém, o desempenho de algoritmos de aprendizagem automática é dependente dos exemplos de treinamentos utilizados para a construção do sistema. E, no caso do ACHE, construir dados de treinamento para formulários online tem uma dificuldade devido à enorme variação de conteúdo e estrutura, podendo ser necessário utilizar técnicas de consulta intuitiva para ajudar os clientes.

Porém, buscas por meio de palavras-chaves e de links são os modos dominantes do acesso a textos na web. Pesquisas de *crawlers* e motores de busca baseados em palavras-chaves são limitados é interface básica fornecida pelos navegadores. Ao se realizar a busca por meio de uma informação específica, o usuário busca encontrar novas páginas relevantes ao tema escolhido. Dessa forma, *focused crawlers* cobrem informações de um tópico específicos com profundidade melhor e mantém o rastreamento mais rápido.

O trabalho de Chakrabarti, Punera e Subramanyam (2002) baseia-se na ideia de que são utilizados os textos em diferentes partes de uma página e as suas distâncias do *hyperlink* para prever a relevância de uma página de destino daquele *hyperlink*. O *crawler* proposto utiliza dois novos classificadores: O primeiro é chamado *apprentice*, que ordena a fila de prioridades de URLs não visitadas e tem treinamento online. As suas características são derivadas de uma árvore *Document Objet Model*(DOM). Sendo usada para calcular distâncias de palavras posicionadas em diferentes folhas da árvore DOM de um *hyperlink* dentro da página. O segundo classificador é o *Baseline* que funciona como um treinador para o *apprentice*.

O processamento de uma árvore DOM é feito, inicialmente, analisando uma página u e formando a árvore DOM para u. Com a árvore construída, é preciso enumerar todas os nós folhas consecutivamente. Uma referência a um link v é um nó interno a_v, raiz de uma subárvore que contém o texto âncora do link $< u, v >$.

Para coletar as características derivadas da árvore DOM utilizou-se uma extensão do modelo Clevel para a coleta de características. O modelo Clevel utiliza a relação entre a relevância da URL v de uma referência (u, v) e a proximidade dos termos de (u, v) aprendidas de forma automática quando um *token* t tem distância x do HREF (u, v). Ou seja, a proximidade de termos para (u, v) é aprendida automaticamente.

O trabalho apresenta uma melhoria simples de um *focused crawler* que melhora a lista de prioridades para as URLs não visitadas na fronteira de rastreamento. O resultado apresenta uma maior taxa de páginas relevantes encontradas para um determinado tópico e uma

diminuição de falsos positivos, que são eliminados quando encontrados.

3.1.3 PageRank

Como apresentado na Seção 2.4, o algoritmo PageRank utiliza a estrutura de links da web para calcular a importância de um documento em uma coleção. *PageRank* (PR) é a probabilidade de uma página web ser visitada por um usuário aleatoriamente por meio de uma página inicial que a referencia. Dessa forma, para cada página web, um valor, entre 0 e 1, é pré-calculado.

Como vantagem, o algoritmo PageRank apresenta o resultado de uma consulta com rapidez e consegue um desempenho melhor que outros modelos por manipular um conjunto adicional de informações para os documentos, através dos *links*. Vários trabalhos utilizam as técnicas de PageRank na construção de um *focused crawler* como (CHO; GARCIA-MOLINA; PAGE, 1998).

O trabalho de Cho, Garcia-Molina e Page (1998) foi um dos pioneiros na área e introduziu uma estratégia de busca *best-first* baseada em critérios simples, como a ocorrência de palavras-chaves e texto âncora.

O objetivo principal do trabalho foi projetar um *crawler* que visitasse as páginas com uma maior similaridade com o objetivo da pesquisa, e por último as com valores mais baixos. Com isso o *crawler* tem que se basear em algumas métricas para adivinhar qual a próxima página a ser rastreada. Dessa forma, uma métrica de ordenação O é utilizada neste trabalho para selecionar uma URL u tal que O(u) tenha o maior valor entre todas as URLs da fila. Foram propostas as seguintes métricas de ordenação:

- **BackLink-Based Crawlers**: o *crawler* administra três estruturas de dados principais: uma fila que contém todas as URLs que precisam ser visitadas. Quando essa página é visitada ela é armazenada em outra fila e os links mantém pares de links (u1, u2), onde u2 foi visitado a partir de u1. Nessa primeira métrica é implementada uma função para comparação que usa três métricas de ordenação: *breadth-first*, *backlink count* e *PageRank*.

- **Similarity-Based Crawlers**: A métrica baseada em similaridade, IS(P), mede a relevância de cada página de um tópico ou uma consulta em um *crawler*.

O trabalho mostrou o uso de métricas de similaridade baseada em links, utilizando o PageRank para mostrar a importância das páginas que contenham muitos links para o *crawler*. O que é uma medida razoável para levar a uma boa coleção de páginas rastreadas.

Muitos trabalhos testaram estas métricas propostas, (FETTERLY; CRASWELL; VINAY, 2009), (ZAÏANE; STRILETS, 2002), (ZAÏANE; STRILETS, 2002). Porém, mostram que estas listas tornam a pesquisa muito lenta, fazendo com que o usuário fique à mercê do PageRank de um motor de busca e tenha dificuldade em articular a uma consulta inicial específica que leve a resultados satisfatórios.

3.1.4 *Breadth-First*

Como apresentado na Seção 2.4, uma métrica de ordenção importante é a a *breadth-first search* (BFS) (NAJORK; WIENER, 2001), que visita um número maior de locais produzindo um rastreamento mais eficiente. O algoritmo inica no nó raiz e percorre todos os nós vizinhos no mesmo nível. Se o objetivo inicial for alcançado, então termina a busca e retorna sucesso. Caso não seja alcançado, continua o processo de busca para o próximo nível até que alcance o objetivo. Alguns trabalhos utilizam esta técnica, como Rennie e McCallum (1999) e Cho, Garcia-Molina e Page (1998).

Em Rennie e McCallum (1999) é proposto um *web spider* dirigido a tópico que é solucionado por uma aprendizagem por reforço.

O aprendizado por reforço refere-se a um *framework* para aprendizagem de decisão ótima de recompensas ou punições. Nele uma tarefa é dada por um conjunto de estados, $s \in S$, um conjunto de ações, $a \in A$, uma função de transição de estado-ação, $T : S \times A \rightarrow S$, e uma função de recompensa, $R : S \times A \rightarrow R$. Em cada passo, o agente seleciona uma ação e tem como resultado uma recompensa e uma nova ação. O objetivo do aprendizado por reforço é aprender um mapeamento dos estados para ações, $\pi : S \rightarrow A$, que maximiza a soma das recompensas ao longo do tempo.

Na tarefa de rastreamento, os documentos dentro de um tópico são as recompensas imediatas, as ações estão seguindo um *hyperlink* e o estado é o vetor de bits indicando quais os documentos dentro do tópico ainda estão sendo consumidos cujas ações foram descobertas. Dessa forma, as principais características específicas do *framework* adequado para aprendizagem em reforço são:

- O desempenho é para ser medido em termos de recompensa ao longo do tempo;
- Ambiente apresenta situações com recompensa retardada;

Dessa forma, utiliza-se essa proposta para resolver o problema enfrentado pelos *focused crawler* em que frequentemente é difícil de um *crawler* aprender que algumas URLs sementes que não satisfazem ao tópico desejado podem levar, muitas vezes, o rastreamento até documentos altamente relevantes. A aprendizagem por reforço é utilizada para treinar um *web crawler* com exemplos específicos contendo os documentos de destino. Dessa maneira, um site ou servidor onde o documento aparece repetidamente é rastreado para que o *crawler* aprenda a construir caminhos otimizados até os documentos de destino.

Porém, esta abordagem apresenta uma sobrecarga em que os usuários especificam sites representativos. A inicialização pode ser lenta e poderia retornar um rastreamento de uma parte do host. E pode enfrentar dificuldades quando a hierarquia de links é distribuídas por uma série de sites.

3.1.5 Resumo Comparativo

A Tabela 3.1 apresenta um resumo das características encontradas pelos trabalhos selecionados sobre *Focused Crawler*.

A maioria dos trabalhos se concentram em buscar documentos na superfície da web, tendo poucos trabalhos em que o *focused crawler* busca por meio da web oculta (BARBOSA; FREIRE, 2007). Muitos utilizam URLs para construir o conjunto de URLs sementes. Poucos utilizaram pesquisas por meio de palavras-chaves em motores de busca para a construção dos mesmos. Destes a maioria utiliza Google, Yahoo ou Bing como motores de busca.

Além disso, a maioria dos trabalhos anteriores ignoraram a dimensão multilingue, todos fizeram apenas a coleta de conteúdo em um único idioma.

Com relação a domínio, alguns trabalhos apresentaram classificação de domínios. Muitos utilizaram diretórios web atualizados, como o *Open Directory Project*(ODP)[1], chamado de DMOZ. O ODP é o mais amplo e abrangente diretório da web editado por humanos. Ele é construído e mantido por uma vasta comunidade global de editores voluntários. Os trabalhos

[1] http://dmoz.org.

que utilizaram o DMOZ foram: (BARBOSA; FREIRE, 2007), (BATSAKIS; PETRAKIS; MILIOS, 2009), (CHAKRABARTI; PUNERA; SUBRAMANYAM, 2002) e (PRIYATAM, 2014).

As tabelas 3.1 apresenta um resumo comparativo das técnicas utilizadas pelos trabalhos apresentados na Seção 3.1. As principais caracteŕisticas são:

1. *Best-First*
2. Pontuação de similaridade
3. Análise de Links
4. Modelo Booleano
5. *Vector Space Model*
6. Modelo *Naive Bayes*
7. *Document Frequency*
8. HMM
9. Árvore DOM
10. HITS
11. Aprendizagem por Reforço
12. Breadth-First
13. PageRank

3.2 Criação de *Corpora* para a língua Portuguesa

Esta seção apresenta os trabalhos relacionados com a construção de *corpora* linguístico para o Português do Brasil e, ao final, é feita uma tabela comparativa com as principais características de cada *corpus*.

Trabalho	1	2	3	4	5	6	7	8	9	10	11	12	13
(AGGARWAL; AL-GARAWI; YU, 2001)	X	X	X										
(ALMPANIDIS; KOTROPOULOS; PITAS, 2007)			X	X	X								
(BARBOSA; FREIRE, 2007)						X	X						
(BATSAKIS; PETRAKIS; MILIOS, 2009)	X							X					
(CHAKRABARTI; PUNERA; SUBRAMANYAM, 2002)						X			X				
(CHAKRABARTI; BERG; DOM, 1999b)			X							X			
(RENNIE; MCCALLUM, 1999)											X	X	
(CHO; GARCIA-MOLINA; PAGE, 1998)			X										X
(MCCALLUM, 1999)						X							
(DILIGENTI, 2000)	X				X	X							X

Tabela 3.1: Comparação de características entre os trabalhos com os trabalhos com *Focused Web Crawler* em Inglês

3.2.1 CETEMPúblico

O trabalho de Rocha e Santos (2000) apresenta a versão 1.0 do *corpus* CETEMPúblico (Corpus de Extractos de Textos Electrónicos MCT/Público), criado em 25 de Julho de 2000, contém mais de 180 milhões de palavras distribuídas por 1.567.625 extractos, correspondentes a cerca de 1.500 edições diárias em português europeu.

Além dos suplementos locais de cada edição, o jornal Público[2] inclui as seguintes secções: Destaque, Política, Internacional, Sociedade, Ciência, Educação, Desporto, Média, Cultura, Economia e Última Página. Semanalmente são publicados cinco suplementos (Computadores, Economia, Artes, Sons e Leituras) e uma revista dominical (a Pública). Apresenta cerca de 0,2% de autores brasileiros e foi o primeiro jornal português com uma edição completa online.

Para criação do *corpus* inicialmente receberam seis CDs contendo cada um aproximadamente seis meses de artigos referentes aos anos de 1996 a 1998. Cada CD estava dividido em vários diretórios, cada um contendo os artigos de uma edição (diária) do jornal. Após o processamento foi notado que o *corpus* resultante abrangia apenas 57 milhões de palavras,

[2]http://www.publico.pt/

então o jornal Público enviou uma segunda versão de CDs, cobrindo os anos de 1991 a 1995.

A Figura apresenta os passos para a construção do *corpus* CETEMPublico final.

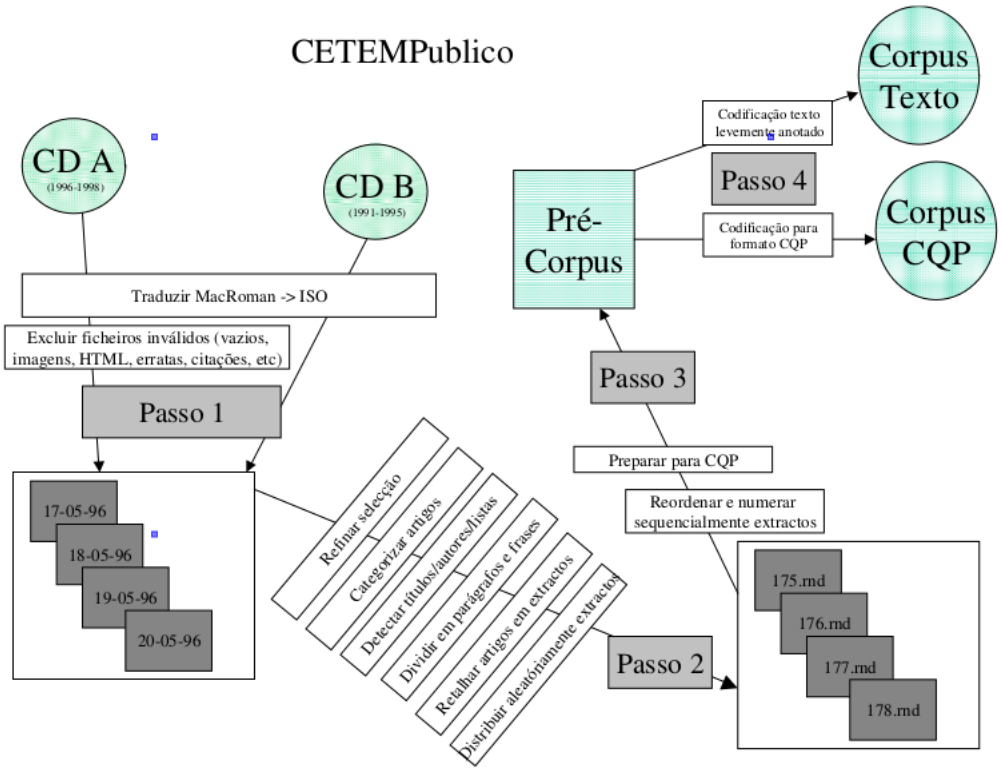

Figura 3.2: Passos para a construção do *corpus* CETEMPúblico proposta por Rocha e Santos (2000)

O passo 1 é iniciado por uma rotina que lê cada documento do CD e cria uma versão modificada no disco. Foi criado um ficheiro para cada edição diária, totalizando mil ficheiros. No segundo passo são realizadas algumas operações: repartição em artigos, seleção dos artigos e o seu refinamento, classificação dos artigos, identificação e anotação dos títulos e autores, separação de frases e divisão dos artigos.

Após, o passo 3 cria um ficheiro compacto, lendo seqfuencialmente cada um dos ficheiros criados no passo anterior. Eliminou-se as anotações de identificação do ficheiro e os extratos são reordenados aleatoriamente e gravados em formato de *tokens* por linha. No final, cria-se um único ficheiro com 180 milhões de *tokens*.

No final, o corpus ficou disponível em dois formatos: uma versão texto com codificação tipo SGML destinada aos investigadores que quiserem usar o corpus como entrada para

os seus sistemas e uma versão CQP5 destinada aos investigadores que queiram sobretudo consultar o corpus.

3.2.2 Corpus CHAVE

O trabalho de Santos e Rocha (2005) criou um *corpus* em português, chamado de CHAVE, usando os mesmos textos (restrito aos anos 1995-1995) que foram usados para a construção do corpus CETEMPúblico (ROCHA; SANTOS, 2000) (textos completos do jornal diário português PÚBLICO). Porém, no *corpus* CETEMPúblico, os documentos foram divididos em parágrafos e este trabalho adaptou os textos em um novo formato. O *corpus* tornou-se uma coleção de 106.821 documentos (348 MB), onde cada documento contém um artigo no jornal. Porém, alguns destes artigos reuniram várias notícias curtas com diferentes assuntos, esse problema pode prejudicar o desempenho de alguns sistemas de RI. O *corpus* é dividido em cinquenta tópicos em português. Os documentos são marcamos com data e tipo de seção, os títulos e autores não foram guardados no texto, porém é fornecido uma lista dos prováveis autores.

O corpus CHAVE foi o primeiro criado no âmbito da participação da língua portuguesa no *Cross-Language Evaluation Forum* - (CLEF). O CLEF é uma série de avaliações conjuntas pretendendo promover a pesquisa e desenvolvimento na área de recolha de informação entre várias línguas.

Porém , além dos textos usados em 2004, ou seja os textos dos anos de 1994 e 1995 do diário português PÚBLICO (note-se, aliás, que no CLEF 2004, na pista de RI apenas foram empregues textos de 1995), a partir de 2005 foram utilizados também textos dos mesmos anos do diário brasileiro Folha de São Paulo quer para RI (adhoc), RAP, ou RIG. Em 2007 deixaram de ser usados na pista de RI, passando a ser usados na pista robusta do CLEF.

A estrutura dos ficheiros disponibilizados é a seguinte:

- Textos: divididos em CHAVEPublico e CHAVEFolha: os textos utilizados nas edições do CLEF: textos completos dos jornais PÚBLICO e Folha de São Paulo dos anos de 1994 e 1995;
- 2004 - Conjunto de recursos obtidos durante a edição do CLEF2004;
- 2005 - Conjunto de recursos obtidos durante a edição do CLEF2005;

- 2006 - Conjunto de recursos obtidos durante a edição do CLEF2006;
- 2007 - Conjunto de recursos obtidos durante a edição do CLEF2007;
- 2008 - Conjunto de recursos obtidos durante a edição do CLEF2008;
- 200x/Monte - Avaliação dos documentos relativos a cada tópico;
- 200x/PerguntasRespostas - Perguntas e respostas compilados pelos organizadores do CLEF200x;
- 200x/Topicos - Tópicos em português compilados pelos organizadores do CLEF200x.

3.2.3 Corpus HAREM

O trabalho de Santos e Cardoso (2006) apresentou uma coleção de texto criada na competição HAREM. HAREM foi a primeira competição de avaliação conjunta de sistemas de reconhecimento de entidades mencionadas e ocorreu o desenvolvimento de uma nova metodologia para avaliação de sistemas de entidades nomeadas.

HAREM foi organizado pela Linguateca e teve 10 participantes de 6 países diferentes, que apresentaram 15 corridas (mais 6 os não-oficiais, ou seja, enviadas após o prazo). Os participantes tinham 48 horas para marcar um grande e variada coleção (a coleção HAREM com 1.202 documentos (mais de 466 mil palavras) a partir de 8 gêneros diferentes e diversas variedades de Português), dos quais uma parte menor foi codificada manualmente pela organização.

O *corpus* resultante do HAREM é uma coleção de textos de várias origens e gêneros, em que as entidades nomeadas foram identificadas, semanticamente classificadas e morfologicamente marcadas no contexto, e foram revisadas de forma independente, de acordo com um grande conjunto das diretivas aprovadas por todos os participantes.

3.2.4 Corpus WaCky

O próximo trabalho utilizou a metodologia *Web-As-Corpus Kool Yinitiative* - (WaCky) para construção de um corpora em português. WaCky foi utilizado na construção de muitos *corpora* para línguas como o Inglês, Italiano e Alemão. Dado isso, foi adotado o método WaCky

para criação do grande corpus para o Português do Brasil, o brWacky (BOOS, 2014). Foram rastreados mais de 1,3 milhões de documentos resultando em um total de 3 bilhões de palavras e deste aproximadamente 8% foi realizado o Pos-tagging.

Para criação do *brWacky* foram feitos alguns passos:

- *Crawling*: construiu-se um conjunto de palavras de conteúdo para usar como sementes para o crawler, esse conjunto foi feito por meio de uma lista de palavras disponível para o corpora em português na linguateca. Essa lista foram removidos os stopwords e aplicou-se um limite de baixa frequência para obter palavras com uma frequência média. A partir deste conjunto, buscou-se em um motor de busca (Bing Api) 1000 pares aleatórios de palavras. Foi gerado um conjunto de URLs (10 para cada consulta realizada) e o *crawler* passa pelos links em cada página e armazenando as páginas visitadas;

- *Post-Crawl Cleaning***:** Esta etapa incluiu a remoção de texto constante, aplicando uma metodologia baseada na biblioteca *boilerpipe*, que usa métricas de densidade e características do texto superficial. Ou seja, as características do texto que são usadas são relacionadas à palavra e ao comprimento médio da sentença;

- *Duplicate Removal***:** Antes da remoção de textos duplicados, foram excluídos alguns textos com tamanhos fora do limite estipulado. Foram conservados apenas documentos entre 5kb e 200kb. Adaptou-se o algoritmo *shingling* consistindo em obter 20 amostras com 5-grams palavras de cada texto e comparar com os outros textos. Se ocorrer mais de dois textos idênticos à amostra, assume-se que os textos são duplicados.

Uma vez que o corpus foi criado, usou-se o TreeTagger treinado para o Português para a tokenização, segmentação de sentenças e *Part-of-Speech(P0S)-tagger*.

Dessa forma, este trabalho apresentou o *corpus* em Português *brWac* com 1,3 milhões de documentos e cerca de 156 milhões de sentenças e 3 bilhões de *tokens*. A coleta do corpus durou aproximadamente 24 horas.

3.2.5 Corpora Floresta Sintá(c)tica

O *corpora* Floresta Sintá(c)tica (BICK, 2007) compõe-se de um conjunto de textos – frases – sintaticamente analisados, em forma de árvore e previamente revistos.

Para a construção da Floresta Sintá(c)tica foi necessário duas fases distintas: uma fase de pré-processamento e a fase de revisão da análise automática propriamente dita (tanto no formato CG como no formato de árvores gerado a partir deste).

A fase de pré-processamento reviu aspectos quanto à parte lexicográfica, que consistiu no enriquecimento do léxico português do sistema português PALAVRAS (BICK, 2000) com a inserção das palavras mais frequentes do corpus (cerca de 8.000 a 9.000 novos lexemas), esta revelou-se de extrema utilidade, evitando erros de análise automática devido a falhas no dicionário do PALAVRAS. Esta fase englobou também uma revisão manual da separação frásica automática.

Na fase de revisão da análise automática foi feita à criação automática e revisão manual do corpus em formato *Constraint Grammar* - (CG), seguido pela geração automática e revisão manual de árvores deitadas. Como o sistema PALAVRAS tinha um grande número de categorias e distinções, foi feito uma revisão morfossintática euxastiva no *corpus* construído, a nível da função e forma sintática e informação morfológica. Em seguida, o trabalho de revisão foi dividido em duas etapas: a revisão de etiquetas CG de forma e função, e constituintes sintáticos tipo estrutura sintagmática, evitando o surgimento de erros gerados pela geração automática de árvores.

Como resultado, foi apresentado na sua primeira fase o Bosque: 1.427 árvores (correspondendo a 251 extractos, 1.405 frases distintas, 36.408 unidades, aprox. 34.256 palavras) revistas e a Floresta Virgem: 41.406 árvores, ou seja, o primeiro milhão de palavras do CETEMPúblico analisado e arborizado automaticamente, sem revisão (7.913 extractos, 41.406 frases, 1.072.857 unidades).

3.2.6 Corpus Brasileiro

O *corpus* Brasileiro foi um trabalho desenvolvido por Berber Sardinha, Moreira Filho e Alambert (2008) do grupo GELC, que está sediado no Centro de Pesquisas, Recursos e Informação de Linguagem (CEPRIL), Programa de Pós-Graduação em Linguística Aplicada

(LAEL) da PUCSP, feito a partir de uma ampla variedade de registros escritos e falados e contém aproximadamente um bilhão de palavras de português brasileiro.

A construção do corpus se deu mediante três passos: (i) a obtenção de textos e transcrições de conversas, textos jornalísticos, fontes off-line com entrevistas sociolinguísticas e reuniões; (ii) estruturação do material em banco de dados *Structured Query Language* (SQL), e (iii) disponibilização do *corpus* para pesquisa on-line, através de um motor de busca PHP disponível em [3]. A estrutura do corpus segue a arquitetura proposta por Davies (2005 inter alia), que consiste na importação de dados textuais em bancos de dados relacionais e, em seguida, consultar os bancos de dados via PHP. Dessa forma, o usuário terá acesso a informações sobre freqüência de ocorrência dos termos de sua busca além de linhas de concordância onde os termos ocorrem e ele não terá acesso ao texto integral, pois isso violaria leis de direitos autorais. Foi necessário atingir o nível de um bilhão de palavras pois um corpus geral deve ter uma amostra de uma população imensa, quanto maior e mais variada essa amostra, mais representativa ela será. O impacto social do Corpus Brasileiro foi significativo, pois colocou ao dispor dos cidadãos do país e do exterior uma vasta quantidade de informação sobre a língua portuguesa. Os usuários do corpus incluíram lingüistas, pesquisadores da linguagem, professores de língua materna, estrangeira, de redação, jornalistas, escritores, roteiristas, publicitários, alunos de diversos níveis, dicionaristas, gramáticos e uma ampla gama de profissionais que lidam com a língua em uso.

3.2.7 Corpus RELI

O trabalho de Freitas (2012) descreve a construção do ReLi – um corpus de Resenhas de Livros manualmente anotado quanto à expressão de opinião. O *corpus* é composto por 1600 resenhas de 13 livros (7 autores), totalizando cerca de 260 mil palavras e 12 mil frases. Para cada livro coletou-se cerca de 200 resenhas e, quando esse número não pôde ser atingido, completou com outras obras do mesmo autor até chegar a um número próximo a 200.

A opinião nas resenhas foi anotada no nível da frase e do sintagma, e acontece em 3 fases: (i) identificação e anotação da polaridade das frases que contêm opinião; (ii) identificação do alvo da opinião; e (iii) identificação e anotação da polaridade do trecho que contém opinião.

O *corpus* foi anotado por 3 anotadores – A, B e C – , embora tenha terminado com

[3] http://www.sketchengine.co.uk/

apenas 1. Todos os anotadores eram alunos de graduação do curso de Letras sem experiência com a tarefa de anotação. Todos passaram por um processo de treino até que estivessem familiarizados com a tarefa. Durante o processo de treino e também durante todo o processo de anotação, os anotadores foram encorajados a perguntar e discutir suas opções e, à medida que surgiam casos não previstos, as soluções eram discutidas e incorporadas ao manual. O resultado passou por uma revisão depois dos primeiros textos anotados, com refinamento das explicações, incorporação de dúvidas, exemplos e casos difíceis. E, finalmente, terminada a anotação, o corpus passou por um processo de revisão, em que buscou por inconsistências.

3.2.8 Resumo comparativo

Os trabalhos desta seção mostram como se deu a construção dos principais *corpora* em português, que estão disponíveis no site da Linguateca[4]. Porém os seus tamanhos podem variar de 23 mil para quase 800 milhões de *tokens*.

A Tabela 5.1 apresenta os corpora em português que estão disponíveis na linguateca. O *corpus* NILC/São Carlos[5] foi desenvolvido no Núcleo Interinstitucional de Lingüística Computacional, sediado no Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo em São Carlos e contém textos brasileiros de registros jornalístico, didáticos, epistolar e redações de alunos.

O *corpus* ANCIB[6] foi criado a partir de uma série de ficheiros contendo as mensagens em português enviadas para a lista homônima da Associação Nacional de Pesquisa e Pós-Graduação em Ciência da Informação.

O *corpus* ECI-EBR foi criado pela *European Corpus Initiative* (ECI), baseado no corpo Borba-Ramsey. É uma seleção de excertos de obras brasileiras, contendo pelo menos discurso literário, didáticos e discursos políticos.

A maioria dos trabalhos se concentra em buscar documentos em jornais ou na literatura, muitos destes trabalhos que buscaram documentos em jornais usou o Jornal Folha de São Paulo, com notícias da década de 90. Dois corpora disponíveis com uma grande quantidade de palavras refere-se ao português de Portugal.

[4]http://www.linguateca.pt/

[5]http://www.linguateca.pt/acesso/corpus.php?corpus=SAOCARLOS

[6]http://www.linguateca.pt/acesso/corpus.php?corpus=ANCIB

Tabela 3.2: *Corpora* em Português

Corpus	Tokens	Sentenças	Língua	Tipo de Dados	Ano
CETEMPúblico	191.300.000	1.567.625	Português Europeu	Edições do Jornal Público	1991-1999
Chave	124.095.306	4.682.363	Português Europeu (Publico) e Brasileiro (Folha)	Edições do Jornal Público	1994-1995
Harém	80.000	5.000	Português brasileiro	coleções douradas do HAREM	2006
brWac	3.000.000.000	156.000.000	Português brasileiro	Dados da Web	2014
corpora Floresta	7.252.306	327.050	Português brasileiro	diversos	
Corpus Brasileiro	792,765,372	29,919,794	Português brasileiro	transcrições de textos	-
ReLi	189,577	8,752	Português brasileiro	Resenha de Livros	-
NILC/São Carlos	42,912,644	1,988,621	Português brasileiro	texto jornalístico, didático e epistolar e redações	1996
ANCIB	1,707,731	83,509	Português brasileiro	Ficheiros de mensagens	2003
ECI-EBR	922,378	44,381	Português brasileiro	Discurso literário, didáticos e políticos	1994

Capítulo 4

Proposta de parametrização

A geração automática de um *corpus* linguístico pode ser enviesada a partir da especificação de alguns parâmetros de interesse do usuário. Neste capítulo, são propostos e descritos parâmetros que atendem à fase de pré-processamento de um *corpus* (Seção 4.1) e fase de pós-processamento de um *corpus* (Seção 4.2). A Seção 4.3 traz a arquitetura do focused crawler proposto.

4.1 Pré-Processamento

Para a construção de um *corpus* é necessário a seleção de textos pertinentes e relevantes para a pesquisa. Nesta etapa, é definido o tipo de corpus que está se compilando e outras decisões que dizem respeito ao seu tamanho e à sua composição em termos dos textos existentes bem como dos gêneros aos quais eles pertencem. Nesta etapa, para a obtenção dos textos pode ser necessário o uso de ferramentas que pré-processem os resultados para as buscas. Neste trabalho, esta fase é denominada de pré-processamento. A parametrização proposta para esta fase, inclui:

- Tipo de coleta. O usuário pode escolher como se dará a coleta do *crawler*, que pode ser por dados da web em geral ou pode ser por dados retirados de *tweets* que estão sendo inseridos na base do Twitter no momento.
- Língua. Atualmente, apesar da enorme quantidade de *focused web crawlers* na literatura a maioria deles apresenta o resultado das buscas na língua inglesa. Dessa forma,

deseja-se um *crawler* que, dependendo das configurações definidas pelo usuário, utilize processamento de documentos para criação de coleções independentes da língua utilizada, mas que tenha preferência pelo português.

- Palavras-chaves. O *crawler* proposto deve calcular a relevância dos documentos de acordo com as palavras-chaves dadas para um determinado domínio. Uma palavra-chave é definida como uma expressão regular descrevendo o domínio investigado. Com isso, as palavras-chaves utilizam um motor de busca sobre os tópicos investigados. Este trabalho executa as palavras-chaves utilizando Bing como motor de busca.

- Domínio. Para a construção de um *focused web crawler*, o domínio é uma característica importante ao conjunto de *corpus* resultante. Em muitos algoritmos, para ocorrer a avaliação é necessário tópicos e alguns alvos relevantes correspondentes. Eles poderiam gerar possíveis temas e conjuntos de prováveis alvos para o domínio especificado. Porém, seria uma desvantagem para quando o *crawler* precisasse de muitos temas ou conjuntos.

  Por isso, muitos *crawlers* utilizam diretórios *web* atualizados, como o *Open Directory Project*(ODP)[1], chamado de DMOZ. O ODP é o mais amplo e abrangente diretório da web editado por humanos. Ele é construído e mantido por uma vasta comunidade global de editores voluntários.

  Para o português o DMOZ apresenta 16 tópicos principais. Cada tópico é representado por três tipos de informações: as palavras da hierarquia (tópicos) do ODP são consideradas palavras-chaves do domínio; as ligações externas constituem metas do alvo e os nós concatenam as descrições de texto e texto âncora das URLs de destino que formam a descrição de um domínio. As palavras-chaves do tópico são dadas aos indexadores como modelos de consulta.

- Tamanho do Corpus. Com relação ao tamanho de um corpus, eles são classificados em pequeno, pequeno-médio, médio, médio-grande e grande, de acordo com o tamanho do vocabulário. O usuário especifica o tamanho de memória disponível para que o *crawler* possa rastrear a quantidade desejada.

[1] http://dmoz.org.

- Sinônimo. A avaliação da qualidade de relevância de um *crawler* é proporcional ao tamanho do dicionário de palavras-chaves relevantes. Dessa forma, é necessário que o *crawler* lide com um grande dicionário de palavras-chaves de relevância para determinados domínios. Por isso o uso de sinônimos é importante ao *focused crawler*, por apresentar novas opções de palavras-chaves.

  Sinônimos de palavras-chaves específicas devem ser definidos e colocados no processamento dos *corpora*. Uma ferramenta útil para atualização de sinônimos para o *crawler* é o dicionário *WordNet* desenvolvido no laboratório de ciência cognitiva da Universidade de Princeton. Trabalhos sugerem que essa ferramenta deve ser executada em fase de inicialização ou sempre que o dicionário de palavras relevantes é alterado.

4.2 Pós-processamento

Após a construção do *corpus* pode ser necessário selecionar textos que farão parte do corpus. Nesta etapa, é feita a limpeza do texto, na qual algumas informações são removidas para melhorar o desempenho de ferramentas. Esta fase é destinada a melhorar a qualidade do texto e é realizada no final do processo de rastreamento, o *focused crawler* poderá fazer o processamento do *Corpus*, podendo utilizar as seguintes especificações:

- Limpeza do texto. Ao final do rastreamento, inicia-se a etapa de extração de informações. Nela é criada a representação dos documentos através do modelo espaço vetorial, que a possibilidade de representar de forma lógica um documento em um espaço vetorial.

  Nessa etapa, o usuário pode usar três tipos de limpeza: limpar caracteres especiais, eliminar pontuação e eliminar *StopWords*.

  Antes da representação espaço vetorial de um documento, faz-se necessária a limpeza do mesmo, buscando eliminar as palavras que não influenciam no significado do documento, as "*StopWords*". Estas palavras, geralmente, são preposições, conjunções e não tem influência no valor semântico do documento.

- N-gram (n=1 a 5). Outra operação que pode ser realizada nesta etapa é a geração de N-gram.

- Entidades nomeadas. Entidades nomeadas são termos que apresentam um ou mais designadores rígidos, num determinado texto. Os mais comuns tipos de entidades são substantivos próprios (nomes de pessoas, organizações, entidades locais), substantivos temporais (datas, tempo, dias, anos e meses) e entidades numéricas (medições).

  O reconhecimento de entidades nomeadas define-se como uma tarefa que visa identificar as entidades nomeadas por sua posterior classificação, atribuindo categorias para as entidades. Neste caso, a entrada vai ser o *corpus* limpo e a saída é um conjunto de anotações. As prováveis saídas para este trabalho serão:

  - Pessoa;
  - Localidade;
  - Organização;
  - Tempo;

4.3 Especificação

Para a especificação do *focused crawler* foram utilizados diagramas *Unified Modeling Language* (UML) através do diagrama de casos de uso, de sequência e pela arquitetura do sistema e de seu funcionamento.

A Figura 4.1 apresenta o diagrama de caso de uso do sistema para inserir uma consulta.

Na Tabela 4.1 é apresentada uma descrição do caso de uso.

A Figura 4.2 mostra a arquitetura de alto nível do trabalho proposto na etapa de pré-processamento. O *focused crawler* foi projetado com o objetivo de calcular a relevância de documentos Web de acordo com uma consulta do usuário.

Quando o usuário envia a consulta ele especifica a linguagem que deseja para seu *corpora*. Após, o usuário indica o tipo de dados para coleta: se por páginas web ou por dados do twitter. A consulta por dados da web pode ser de dois tipos: por meio de palavras-chaves ou por um conjunto de URLs . Essas URLs sementes podem ser pré-processadas pelo crawler, dadas pelo usuário ou por domínios específicos do DMOZ.

Em seguida, após a criação do *corpus* é feito um pós-processamento. A Figura 4.3 apresenta a arquitetura de alto nível do trabalho proposto na etapa de pós-processamento. Através

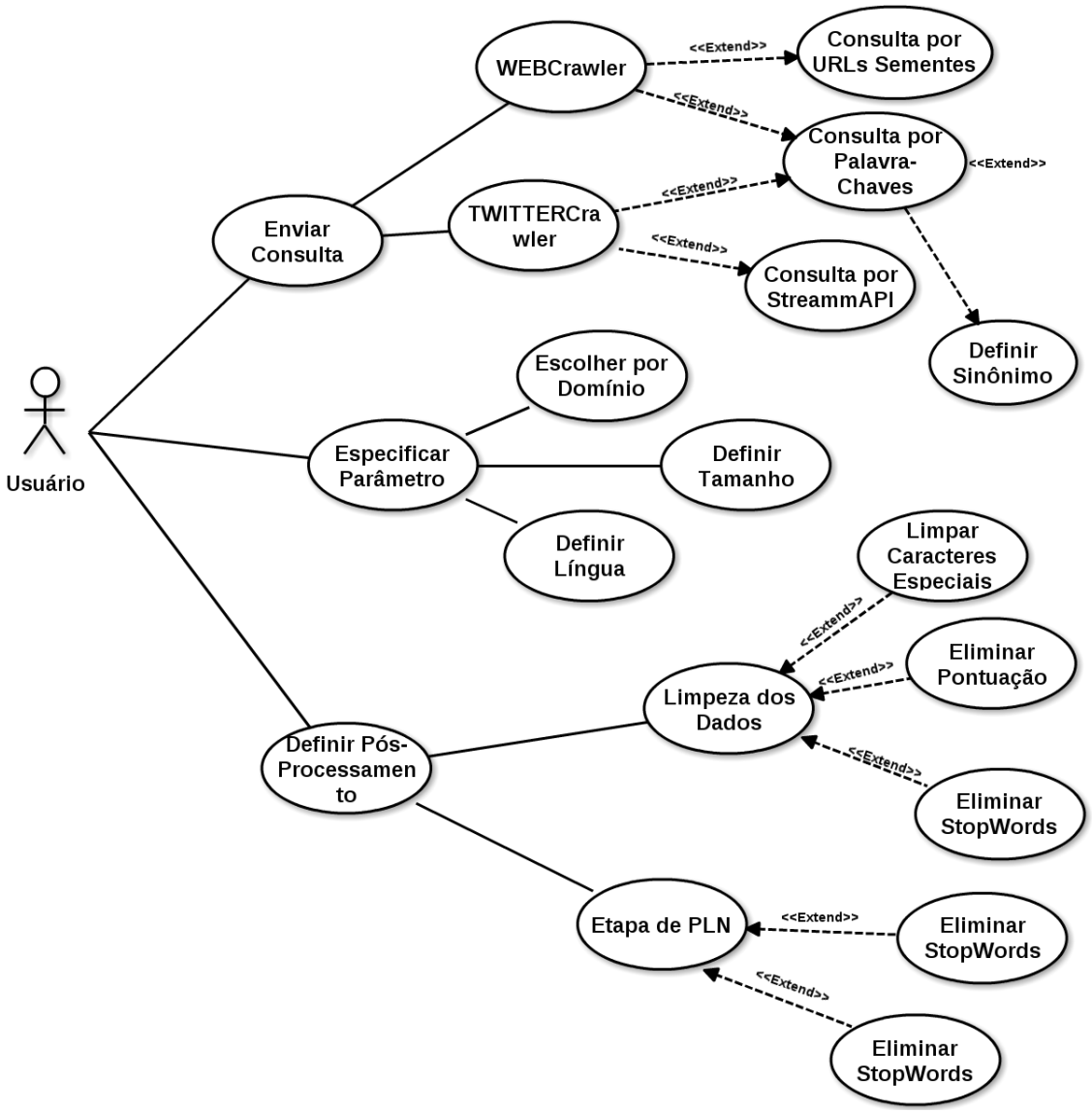

Figura 4.1: Casos de uso disponibilizados pela interface a um utilizador humano quando este efetua a configuração do sistema

desta etapa pode ocorrer um processamento dos dados por meio de algoritmos de aprendizagem de máquina que são: limpeza de texto, *N-gram* e entidades nomeadas.

Caso de Uso	**Ator**	**Descrição**
Envia Consulta	Usuario	usuário do sistema envia uma consulta
Especifica Parâmetros	usuário	usuário do sistema especifica parâmetros necessários para o *corpus*
Consulta Palavras-Chaves	usuário	usuário escolhe enviar consultas por palavras-chaves
Consulta URLs sementes	usuário	usuário escolhe enviar consultas por URLs sementes
Escolher Domínio	usuário	usuário escolhe qual domínio vai ser especificado no *corpus*
Definir Língua	usuário	usuário define qual língua vai ser especificada no *corpus*
Definir Sinônimos	usuário	usuário define se são utilizados sinônimos na consulta
Definir Tamanho do *Corpus*	usuário	usuário define o tamanho do *corpus* desejado

Tabela 4.1: Descrição de casos de uso - crawler

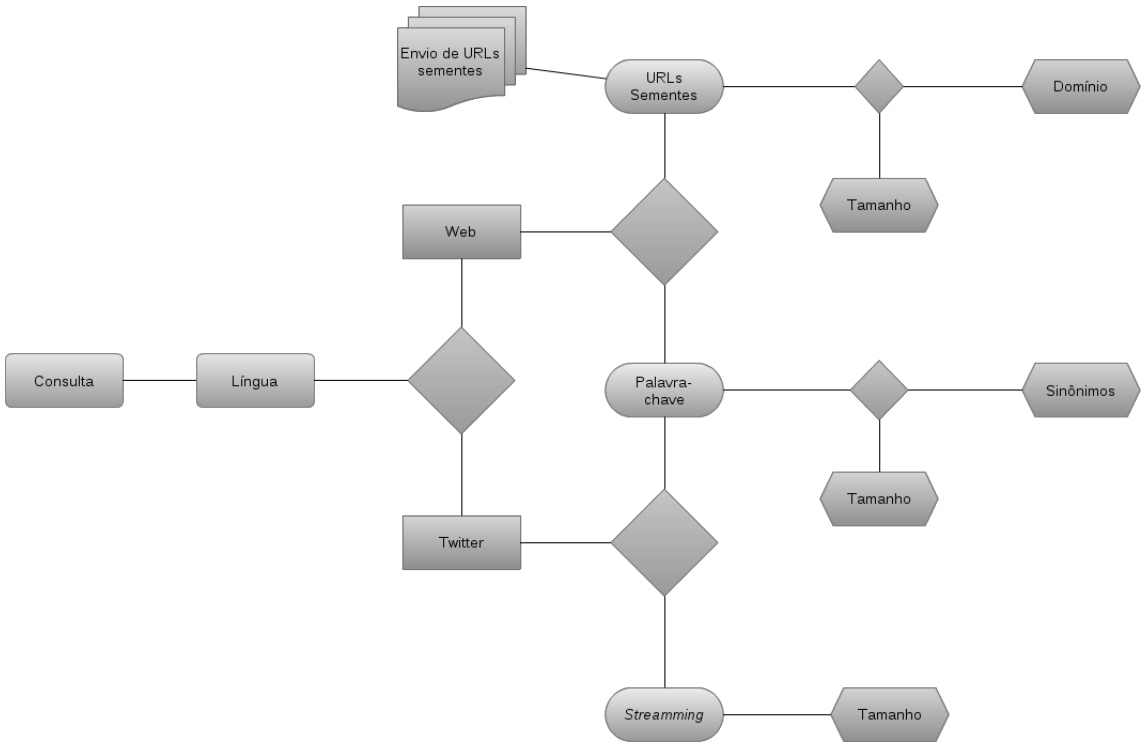

Figura 4.2: Arquitetura de alto nível do pré-processamento

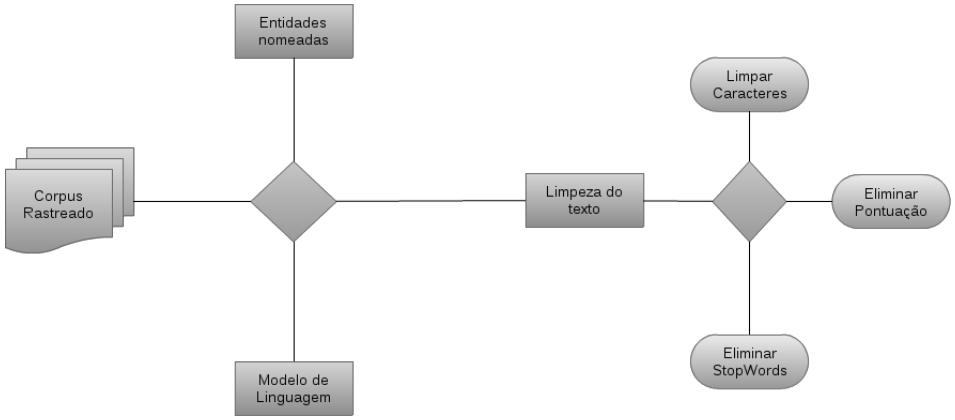

Figura 4.3: Arquitetura de alto nível do pós-processamento

Capítulo 5

Construção de *Corpora*

Este capítulo apresenta a criação de 3 (três) diferentes *corpora* linguísticos de modo a mostrar a aplicabilidade da proposta de definir parametrização para *focused crawlers* ao tempo que contribuem para as pesquisas em Linguística Computacional para a língua portuguesa. Os *corpora* foram denominados de VazaBarris, Poxim e Paramopama, uma homenagem a grandes rios do Estado de Sergipe, Brasil.

O *corpus* VazaBarris consiste de 86 milhões de palavras obtidas a partir de 150 mil páginas da Web de notícias. O *corpus* Poxim consiste de 3 milhões de palavras a partir de mais de 300 mil tweets rastreados. O paramopama consiste de 310 mil palavras anotadas a partir de 12.500 sentenças.

Os três *corpora* estão disponível para downloand: VazaBarris[1], Poxim[2] e Paramopama[3]

Os *corpora* VazaBarris e Poxim estão descritos e devidamente avaliados na seção 5.1. O *corpus* Paramopama é apresentado e avaliado na seção 5.2.

A geração de corpus a partir de streaming de tweets foi submetida a uma investigação mais aprofundada. Um método para esta geração foi proposto e devidamente avaliado na seção 5.3.

[1]VazaBarris: https://www.dropbox.com/s/bx1pz1pp4foyci3/VazaBarris.rar?dl=0

[2]Poxim: https://www.dropbox.com/s/fc1ckak8rkvgkgz/Poxim.rar?dl=0

[3]Paramopama:https://goo.gl/9e3O1O

5.1 Construção dos *corpora* VazaBarris e Poxim

5.1.1 VazaBarris

O método de construção para VazaBarris consistiu em duas etapas principais: (1) coleta e manipulação de páginas da Web e (ii) manipulação do *corpus*.

Para realizar o rastreamento, foi utilizado o *crawler* MEMEX (BARBOSA; FREIRE, 2007). Foram selecionados 158 sites Web de notícias brasileiras que está disponível em [4] e realizou o rastreamento somente dentro desses sites. O resultado foi um conjunto de 150 mil páginas HTML.

Na segunda etapa, ocorreu o pós-processamento do texto coletado. Ocorreu primeiro a limpeza dos textos, cada página web foi limpa. O processo de limpeza consistiu na remoção de caracteres especiais, que neste caso foram links URL, tags HTML, imagens, tabelas e números de página.

5.1.2 Poxim

As redes sociais são importantes recursos para analisar os padrões de comunicação atuais das pessoas. Poxim é um *corpus* construído pelo rastreamento de tweets. Seu processo de construção consistiu em duas etapas principais: (1) rastreamento de tweets e (ii)manipulação de *corpus*.

Foi utilizada a StreamingAPI do twitter para coletas dados do twitter. O crawler é implementado usando o Twitter4j [5], uma API interface Java para publicar Twitter. A biblioteca language-Detection (SHUYO, 2010) foi usada para identificar automaticamente *tweets* em Português. O resultado deste primeiro passo foi um conjunto de cerca de 300 mil mensagens de texto(tweets).

O segundo passo, foi realizada a etapa de pós-processamento. Limpou-se as mensagens coletadas, incluindo a remoção de caracteres especiais, que neste caso é a remoção de links de URL (por exemplo http://url.com), nomes de usuário (por exemplo @Nome) e hashtags (por exemplo hashtag). Enfim, foi realizada a fase de Stemming.

[4] Lista de site de notícias brasileiros: https://www.dropbox.com/s/m2hmj7ftn3tq64l/sites%20de%20not%C3%ADcias%20bra

[5] http://twitter4j.org/en/index.html

5.1.3 Experimentos e Resultados

Foi criado um modelo de linguagem para os corpora (VazaBarris e Poxim) e, para comparação, para dois outros corpora em Português – Floresta (BICK, 2007) e Chave (SANTOS; ROCHA, 2005). O corpora Floresta foi construído como resultado de um projeto de pesquisa Floresta Sintética (BICK, 2007), uma colaboração entre Linguateca e projeto VISL. Neste trabalho, foram usadas as 4 partes do Floresta.

O outro corpus usado, Chave, é o resultado do envolvimento de Linguateca no Cross-Language Evaluation Forum (CLEF) (ROCHA; SANTOS, 2006). O texto do *corpus* Chave é originado do jornal *Folha de São Paulo* e de textos públicos. A Tabela 5.1 mostra o tamanho de cada *corpus* em termos de número de palavras. Para criar o modelo de linguagem, foi utilizado a ferramenta SRILM (STOLCKE, 2002).

A perplexidade dada pelas Equações 2.13 e 2.14 foi computada sobre 8 diferentes conjuntos de testes (ver tabela 5.2). Estes conjuntos de teste são diálogos em redes sociais, e-mails e blogs, que foram gentilmente cedidas por voluntários, estão disponíveis em [6]. Vale ressaltar que, para conjuntos de teste com valores de Out of Vocabulary (OOV) distintos, a comparação dos valores de perplexidade não faz sentido(DYBKJAER; HEMSEN; MINKER, 2007). Então, para cada conjunto de treinamento foi gerado um modelo de linguagem para cada um dos conjuntos de testes, dado que $OOV = 0$.

Tabela 5.1: Conjunto de Treinamento

Nome	Tamanho (número de palavras)
VazaBarris	86,993,260
Poxim	3,006,240
corpora Floresta	6,046,541
Chave	97,884,763

As Tabelas 5.3 e 5.4 mostram os resultados para a perplexidade de cada *corpus*. O valor N nas colunas $Corpus-N$ está relacionado ao $N-gram$ usado na avaliação. Consideramos $N \in [1, 5]$.

[6]Conjunto de testes para cálculo de perplexidade: https://www.dropbox.com/sh/middx8dysqldf0j/AAAqX4gs9T63YBiG79uI

Tabela 5.2: Conjunto de teste

Nome	Tamanho (número de palavras)
Set1	113,211
Set2	33,353
Set3	6,084
Set4	3,766
Set5	17,426
Set6	11,718
Set7	13,514
Set8	16,793

Os resultados mostram que Poxim alcançou o melhor valor de perplexidade. Isso é esperado, uma vez que o tipo de texto (estilo de escrita) do Twitter é semelhante ao dos conjuntos de teste, tomados a partir de redes sociais de diálogo textos, e-mails e blogs. Com relação especificamente ao unigram (1-gram), os resultados são bastante semelhantes entre todos os *corpora*, que salienta o fato de que, embora o domínio possa variar, o vocabulário é bastante semelhante, excepto para o *corpus* Chave.

Com relação às figuras 5.1-5.5, fica claro que o *corpus* Poxim apresenta os melhores resultados se comparado com os *corpora* Floresta e VazaBarris que mostram pequenas variações entre eles. Também é possível observar estas pequenas variações na linha 3 das Tabelas 5.5 e 5.7.

Também foi calculada a interpolação linear de combinações de dois diferentes corpora. Cada *corpus* dentro de uma interpolação tem um λ correspondente, o que representa a taxa de contribuição de cada *corpus* para o modelo de linguagem. O par de valores de λ pode variar do conjunto de teste para conjunto de teste, como mostram as Tabelas 5.5 e 5.7. A fim de encontrar os melhores valores de λ, foi utilizada uma implementação do SRILM para Expectation Maximization (EM).

A partir das Tabelas 5.5 e 5.7, pode-se ver que o *corpus* Poxim traz a maior contribuição quando interpolado com qualquer outro corpus. A razão para isso se dá pois Poxim tem um estilo de escrita semelhante (conversional) ao conjunto de testes. Floresta tem uma alta con-

Tabela 5.3: Perplexidade dos conjuntos de testes 1 a 4

corpus-N	Set1	Set2	Set3	Set4
corpora Floresta-1	412.5	218.3	131.0	17.9
corpora Floresta-2	545.8	290.9	218.3	19.8
corpora Floresta-3	550.6	295.5	222.3	20.2
corpora Floresta-4	555.4	298.2	224.8	20.3
corpora Floresta-5	557.6	299.9	226.8	20.4
Poxim-1	193.1	103.2	59.7	12.9
Poxim–2	178.9	98.4	73.4	12.5
Poxim–3	179.0	96.9	75.1	12.6
Poxim–4	183.2	98.9	77.1	12.7
Poxim–5	185.2	100.4	78.1	12.8
VazaBarris-1	420.9	235.6	79.5	8.7
VazaBarris–2	480.1	302.0	95.4	8.6
VazaBarris–3	517.6	322.8	103.5	8.6
VazaBarris–4	552.2	342.8	107.2	8.8
VazaBarris–5	568.3	355.4	110.5	8.9
Chave-1	619.0	400.2	193.1	21.1
Chave–2	1519.8	1062.8	584.0	24.7
Chave–3	1537.5	1100.0	602.9	24.9
Chave–4	1550.0	1113.2	611.6	25.1
Chave–5	1557.5	1118.7	617.1	25.4

tribuição quando interpolada com VazaBarris. Isso ocorre porque Floresta é composto por documentos com opiniões, debates em que o estilo é semelhante ao dos textos nos conjuntos de teste. VazaBarris, por outro lado, apenas contém os textos de jornal.

As Tabelas 5.6 e 5.8 apresentam a perplexidade para os pares de interpolação para cada N-gram.

A partir dos resultados, pode-se ver que o melhor é a interpolação entre os *corpora*

Tabela 5.4: Perplexidades para os conjuntos de testes 5 a 8

corpus-N	Set5	Set6	Set7	Set8
corpora Floresta-1	305.6	309.9	355.1	314.4
corpora Floresta-2	473.8	579.5	626.6	546.1
corpora Floresta-3	482.1	579.9	635.0	549.8
corpora Floresta-4	484.4	583.5	635.9	554.2
corpora Floresta-5	486.8	584.4	636.8	556.1
Poxim-1	146.4	121.0	118.4	141.1
Poxim–2	161.2	140.9	122.0	164.0
Poxim–3	162.7	142.0	121.7	161.6
Poxim–4	166.5	144.7	123.5	163.5
Poxim–5	167.7	145.9	124.4	165.1
VazaBarris-1	347.5	276.7	317.6	310.0
VazaBarris–2	500.2	481.3	511.2	512.1
VazaBarris–3	550.7	542.5	566.4	564.3
VazaBarris–4	579.1	575.1	589.6	598.7
VazaBarris–5	595.7	595.7	604.0	615.8
Chave-1	518.3	515.5	333.7	524.9
Chave-2	1547.3	2145.7	1084.0	1742.4
Chave–3	1557.3	2174.7	1103.7	1757.6
Chave–4	1561.4	2188.7	1106.6	1780.9
Chave–5	1569.0	2194.3	1107.8	1789.7

VazaBarris e Poxim: os valores de perplexidade para tal interpolação linear são os mais baixos. Isso indica que VazaBarris e Poxim são dois corpora complementares para esta tarefa.

Foi realizado um experimento final usando VazaBarris como o conjunto de treinamento e os conjuntos de testes são os *corpora* Floresta, Poxim e Chave. Os valores de perplexidade são mostrados na Tabela 5.9. Pode-se notar que o *corpora* Floresta e o *corpus* Chave mostrou

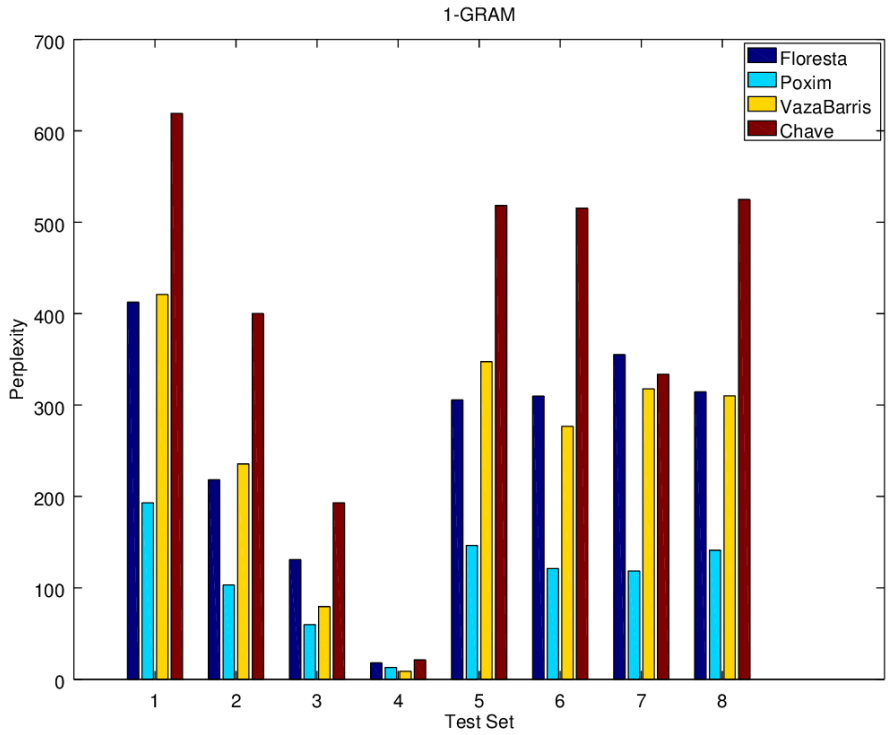

Figura 5.1: Perplexidade média de cada *corpus* para unigrama.

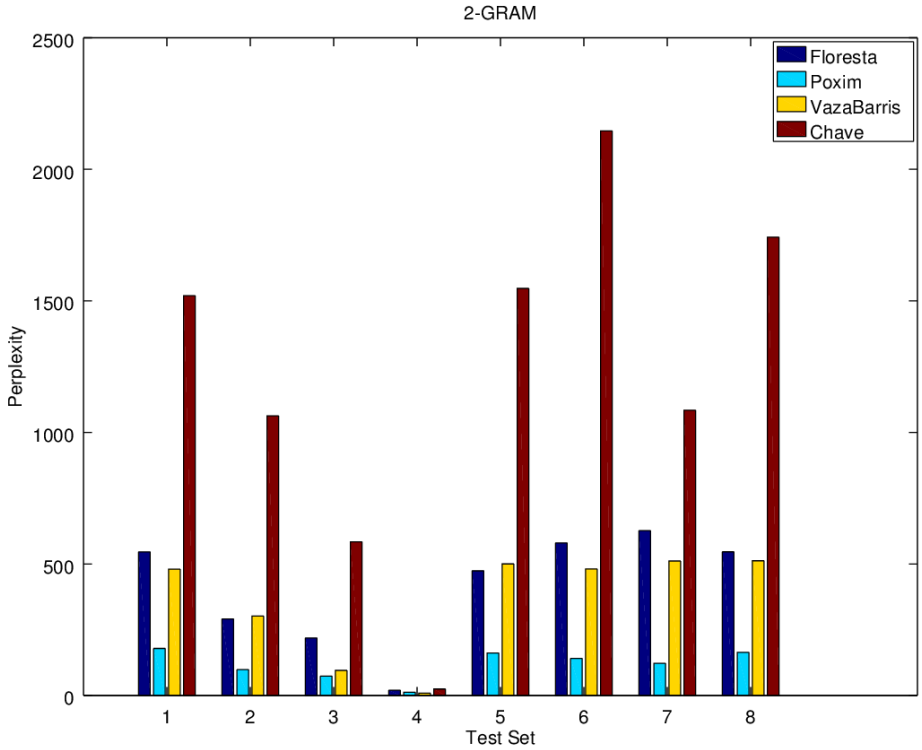

Figura 5.2: Perplexidade média de cada *corpus* para bigrama.

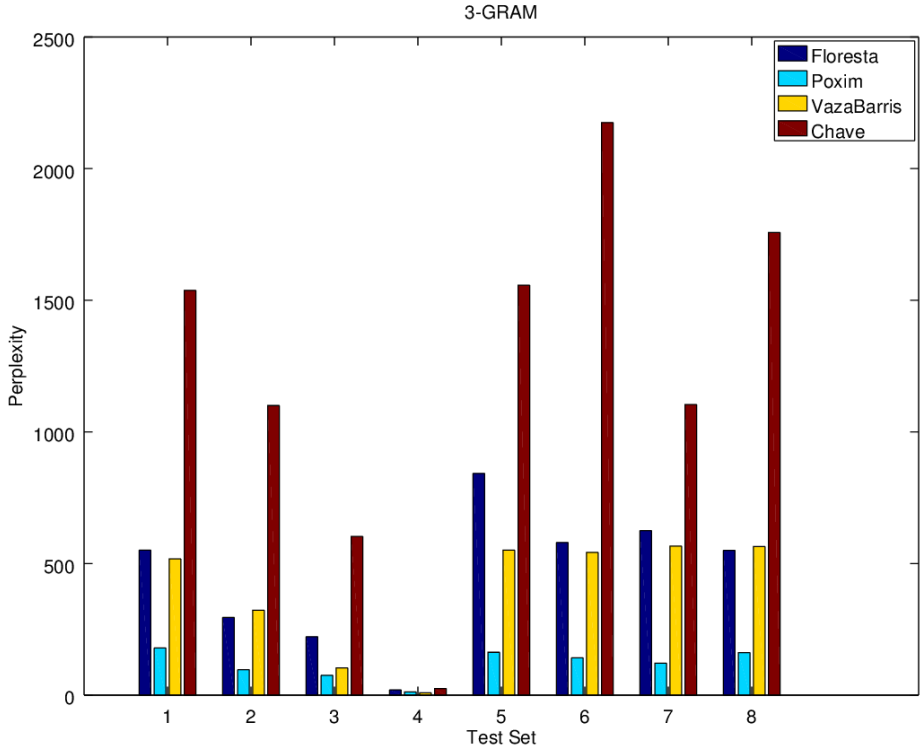

Figura 5.3: Perplexidade média de cada *corpus* para trigrama.

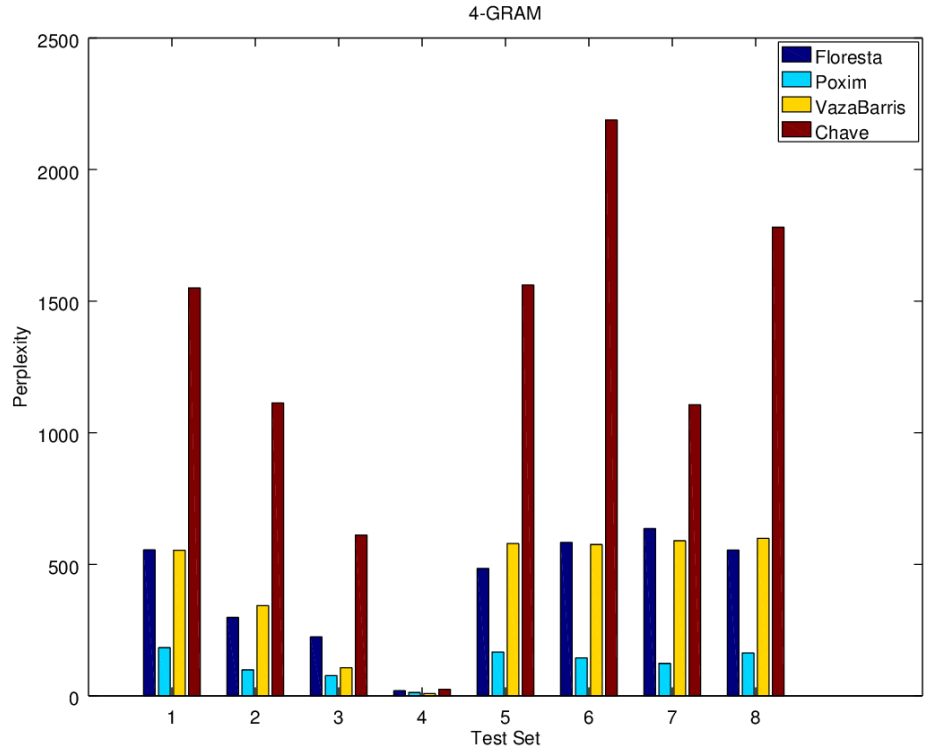

Figura 5.4: Perplexidade média de cada *corpus* para 4-grama.

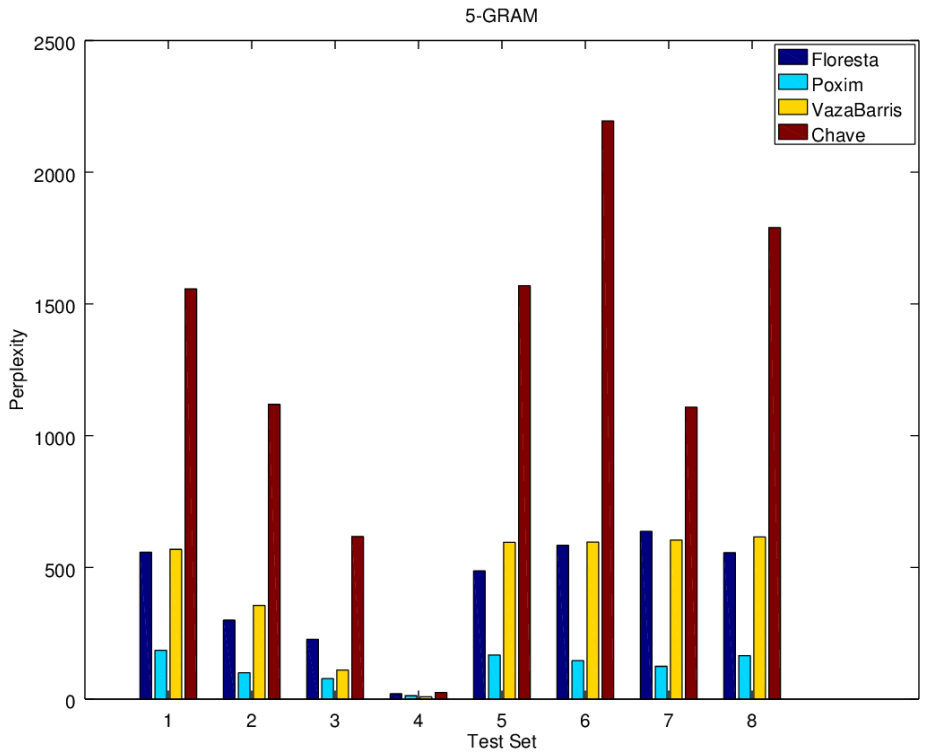

Figura 5.5: Perplexidade média de cada *corpus* para 5-grama.

Tabela 5.5: Valores de λ paca cada interpolação linear entre os dois *corpora* (conjunto de testes do 1 ao 4).

VazaBarris-*Corpus*	Set1	Set2	Set3	Set4
VazaBarris-Poxim	0.19;0.81	0.15;0.85	0.12;0.88	0.15;0.85
VazaBarris-Chave	0.70;0.30	0.72;0.28	0.69;0.31	0.39;0.61
VazaBarris-Floresta	0.39;0.61	0.36;0.64	0.37;0.63	0.25;0.75
Poxim-Chave	0.89;0.11	0.91;0.09	0.96;0.04	0.81;0.19
Poxim-Floresta	0.79;0.21	0.80 0.20	0.84;0.16	0.72;0.28
Floresta-Chave	0.86;0.14	0.92;0.08	0.89;0.11	0.75;0.25

resultados muito melhores que Poxim.

5.2 Paramopama: um *corpus* brasileiro para reconhecimento de entidades nomeadas

Entidades nomeadas (do inglês *Named Entities* - (NE)) são basicamente qualquer coisa que pode ser referido com um nome próprio pertencentes a categorias pré-definidas, tais como os nomes das pessoas, organizações, locais, expressões de tempo, quantidades, valores monetários, porcentagens, entre outros. NER desempenha um papel central para extrair informações relevantes de um texto. Em (FINKEL; GRENAGER; MANNING, 2005), por exemplo, o reconhecimento de NER é utilizado para fornecer o reconhecimento de dados biomédicos, tais como as proteínas, os dados de ADN e de ARN e tipo de células a partir de artigos médicos.

Como uma tarefa típica de classificação automática,a detecção de entidades nomeadas precisa de um bom conjunto de treinamento. Em um conjunto de treinamento, todos os termos que representam entidades desejadas devem estar devidamente rotulados. Normalmente, o processo de rotulagem é uma tarefa propensa a erros. Felizmente, há uma grande quantidade de corpora de entidades nomeadas disponíveis em diferentes idiomas: Inglês (TJONG KIM SANG; DE MEULDER, 2003), (WEISCHEDEL; BRUNSTEIN, 2005), (CHINCHOR, 1998), (DODDINGTON, 2004), espanhol (TJONG KIM SANG; DE MEULDER, 2003), Alemão (TJONG KIM SANG; DE MEULDER, 2003), Holandês (TJONG KIM SANG; DE MEULDER, 2003), chinês (DODDINGTON, 2004), Árabe (DODDINGTON, 2004).

Para Português, no entanto, há uma falta de corpora com reconhecimento de entidades nomeadas - a partir de agora, vamos nos referir a *corpora* deste tipo como corpora PTBR NE. Os corpora mais relevantes, certamente os mais citados, são o corpora HAREM (SANTOS; CARDOSO, 2006). Combinados, eles somam 10 mil sentenças com 200.000 palavras com tag em 9 categorias diferentes. Os melhores resultados alcançados pelo HAREM foi um F-medida de 70%. Para comparação, os corpora em inglês atualmente atingiram uma medida-F de 88% (TJONG KIM SANG; DE MEULDER, 2003). Aqui, serão utilizadas apenas 4 entidades do corpus HAREM (Pessoa, Localização, Organização e Tempo) para uma melhor comparação entre os corpora.

Este trabalho estendeu a versão PTBR de WikiNER corpus, revisando tags designadas incorretamente, a fim de melhorar a qualidade do corpus. Foi também estendido o tamanho do *corpus* e fornecida uma avaliação adequada. O novo corpus foi chamado de Paramopama.

5.2.1 Método de construção

O processo para construção do *corpus* foi realizado em duas etapas: melhoria e expansão, conforme apresentado na figura 5.6

- Melhoria: Nesta fase, um classificador NE foi treinado com o *corpus* HAREM para rotular automaticamente o corpus WikiNER. O resultado desse processo de rotulagem automática foi então revisto e comparou-se os dois rótulos de cada palavra em cada frase e removidas manualmente as entidade nomeadas erradamente. Além disso, também foi acrescentada a tag TEMPO ao *corpus* para identificar as expressões de tempo. No final deste processo, foi produzido, como um todo, um conjunto de 10.000 frases do corpus WikiNER que tiveram suas marcações corrigidas e revisadas.

- Expansão. Nesta fase, foi utilizado o conjunto produzido na fase de melhoria para treinar um classificador NER e rotular em torno de 2500 sentenças mais recentes obtidas a partir de sites de notícias de vários domínios diferentes (por exemplo, economia, política, esportes, tecnologia). Em particular, foi influenciado o processo de rastreamento para presença de instâncias da entidade de ORGANIZAÇÃO. Isto foi importante, a fim de melhorar a capacidade de aprendizagem do classificador para esta entidade, reduzindo assim o número de falsos negativos da entidade ORGANIZAÇÃO e diminuir o número de falsos positivos de outras entidades. Em seguida, o classificador NE resultante foi usado para marcar as frases mais recentes. Finalmente, as palavras de marcação foram avaliadas e as frases foram adicionados ao *corpus* final.

O resultado deste processo duplo é um *corpus* totalmente marcado, Paramopama, composto por sentenças revistas do *corpus* WikiNER e um conjunto de sentenças de notícias, marcadas e avaliadas. As Tabelas

O *corpus* Paramopama foi anotado com as entidades: Pessoa, Localização, Organização e Tempo. Outras palavras foram marcados como “Outros”. A figura 5.7 mostra trechos de Paramopama.

Para proceder à classificação das entidades, foi utilizada a ferramenta Stanford NER por ser um classificador de ordem arbitrária em sequência que implementa o *Conditional Random Fields*(CRF) (LAFFERTY; MCCALLUM; PEREIRA, 2001). A justificativa para a escolha do

CRF é que ele tem a capacidade de atenuar a forte suposição do HMM e de gramáticas estocásticas, por exemplo. Também evita uma questão importante do *Maximum Entropy Markov Models* (Memm): um comportamento preconceituoso de estados que são precedidos por alguns estados (LAFFERTY; MCCALLUM; PEREIRA, 2001). Além disso, trabalhos anteriores mostram bons resultados do CRF para NER em *corpora* em Inglês(MCCALLUM; LI, 2003), (SETTLES, 2004).

Além de um *corpus* devidamente identificado e a escolha de um modelo de aprendizagem adequado, a definição de um conjunto de características relevantes é crucial. Características devem ser preditoras plausíveis da classe e deve ser facilmente e confiavelmente extraída do *corpus*. Algumas destas características são relacionadas com o modelo de aprendizagem automática, tais como a janela deslizante do algoritmo de Viterbi, em que uma palavra posicionada para a frente pode influenciar a marcação de uma palavra anterior. Outras características estão relacionadas com o *token* para serem identificadas e para o texto circundante. Por exemplo, palavras com letras maiúsculas e minúsculas ou pontuação, como eBay e Yahoo!, são indicadores fortes de que a palavra deve ser marcada como OROGANIZAÇÃO. Uma data, por outro lado, pode ser mais fácil para identificação com a utilização de expressões regulares, de modo que elas são marcadas na classe TEMPO. Características que definem cidades, países ou monumentos podem ser usadas para marcar as palavras na classe LOCALIDADE. Da mesma forma, dicionários de nomes de pessoas podem ser usados para marcar as palavras na classe PESSOA.

Varias características foram definidas no classificador. A maioria delas refere-se a itens lexicais tais como a forma de texto, letras de palavras em n-gramas, caso palavra, presença de dígitos, pontuação ou hífen, lemmas e stems, muito importante na detecção de expressões de tempo, por exemplo. Dicionários também foram definidos para nome de pessoas, cidades, países e organizações e foram recolhidos a partir de sites e bancos de dados disponíveis na Internet. Esses dicionários são essenciais para a detecção de todas as classes, especialmente os nomes de pessoas. Outras características referem-se ao contexto da palavra a ser anotada, tal como a palavra anterior, a palavra seguinte, a palavra anterior da etiqueta, entre outros. A ordem CRF é também uma característica importante, pois define a janela de contexto em que todos os recursos são avaliados. A Tabela 5.12 enumera algumas das características que definimos para representar as instâncias no conjunto de treinamento.

5.2.2 Avaliação e Resultados

O objetivo de nossa avaliação experimental é avaliar o corpus Paramopama, comparando os resultados com outros três corpora PTBR NE-marcado e combinações entre eles: Primeiro HAREM, Segundo HAREM, Primeiro + Segundo HAREM (HAREM completo), WikiNER, Paramopama e Paramopama + harém. Precisão, Cobertura e medida-F (F1-Score) são as medidas de avaliação.

Dois conjuntos diferentes de teste foram considerados. O primeiro conjunto de teste consistiu nos últimos 10% do *corpus* Segundo HAREM, sendo responsável por 383 sentenças (este conjunto de teste está disponível em [7]). O segundo conjunto de teste consistiu em 5.855 sentenças do *corpus* WikiNER que foram melhoradas como resultado da fase de Melhoria, este conjunto de teste está disponível em [8].

Os testes foram realizados a partir da ferramenta StanfordNER (**?**), onde foram computados os TP(*true positives*), FP (*false positives*) e FN (*false negatives*) a partir do classificador de entidades nomeadas treinado com o corpus Paramopama, sobre os dois conjuntos de testes.

Em ambos os conjuntos de teste, as entidades nomeadas utilizadas na avaliação são PESSOA, LOCALIZAÇÃO, ORGANIZAÇÃO e TEMPO. Na verdade, o *corpus* WikiNER não é avaliado em relação à entidade TEMPO desde que a tag não estava presente no *corpus* inicial.

As Tabelas 5.13, 5.14, 5.15 e 5.16 mostram os valores de precisão, cobertura e medida-F para o primeiro conjunto de teste e as Tabelas 5.18, 5.19, 5.20 e 5.21 mostram os valores de precisão, cobertura e medida-F1 para o segundo conjunto de teste (grupo 2), para as entidades nomeadas PESSOA, LOCALIZAÇÃO, ORGANIZAÇÃO e TEMPO, respectivamente. . Tabelas 5.17 e 5.22 mostram o resultado médio geral para o Conjunto 1 e do Conjunto 2, respectivamente.

Os resultados mostram que, em geral, o *corpus* Paramopama tem um resultado melhor do que ambos *corpus* HAREM e WikiNER. A pontuação geral para definir um teste, apresentados na Tabela 5.17, indica uma medida-F em torno de 80% para Paramopama enquanto que o *corpus* HAREM atingiu, no máximo, cerca de 77% e o *corpus* WikiNER com menos do

[7] Conjunto de teste 1: https://www.dropbox.com/s/o5yr1kyy2tbuvgv/10Harem.txt?dl=0

[8] Conjunto de teste 2: https://www.dropbox.com/s/yr5xny3eoww65hx/WikNER$_test.txt?dl = 0$

que 60%. Um desempenho comparativo semelhante foi alcançado para o segundo conjunto de teste, no Tabela 5.22. Como o resultado da Precisão, cobertura e medida-F nos testes foram significantes para a entidade TEMPO, foi aplicado o classificador de entidade TEMPO sobre o *corpus* do HAREM e combinado com o *corpus* WIKINER melhorado, formando assim, o *corpus* Paramopama. A combinação do Paramopama com o HAREM alcançou os melhores resultados, com um excelente desempenho para a classe TEMPO com uma média de medida-F de 92% (Tabelas 5.16 e 5.21).

Tabela 5.6: Perplexidade para os conjuntos de testes do 1 ao 4 (interpolação)

VazaBarris-*Corpus*-N	Set1	Set2	Set3	Set4
VazaBarris-Poxim-1	189.3	102.4	60.1	13.0
VazaBarris-Poxim–2	163.1	93.3	71.8	12.3
VazaBarris-Poxim–3	161.2	91.3	73.5	12.3
VazaBarris-Poxim–4	163.4	92.6	74.8	12.4
VazaBarris-Poxim–5	164.7	93.7	75.6	12.5
VazaBarris-Chave-1	379.7	216.9	127.8	18.2
VazaBarris-Chave-2	434.0	273.1	197.5	18.8
VazaBarris-Chave-3	444.2	281.8	208.9	19.1
VazaBarris-Chave-4	459.1	291.9	213.2	19.2
VazaBarris-Chave-5	466.1	298.5	216.5	19.4
VazaBarris-Floresta-1	316.2	184.7	108.0	16.6
VazaBarris-Floresta–2	318.6	206.3	143.8	16.7
VazaBarris-Floresta–3	319.4	208.0	148.0	16.9
VazaBarris-Floresta–4	323.7	210.3	149.0	17.0
VazaBarris-Floresta–5	325.6	212.4	150.4	17.1
Poxim-Chave-1	521.4	220.5	148.1	17.4
Poxim-Chave-2	461.0	202.7	187.9	16.1
Poxim-Chave-3	449.7	196.5	192.1	16.1
Poxim-Chave-4	456.5	199.4	197.8	16.2
Poxim-Chave-5	460.2	202.0	200.3	16.3
Poxim-Floresta-1	521.8	221.4	150.7	17.4
Poxim-Floresta–2	456.0	198.4	184.5	16.0
Poxim-Floresta–3	444.0	191.2	186.8	15.9
Poxim-Floresta–4	448.7	193.0	190.8	16.0
Poxim-Floresta–5	451.8	194.8	192.6	16.1
Floresta-Chave-1	1242.5	525.8	383.5	24.8
Floresta-Chave-2	1629.5	711.6	672.1	26.0
Floresta-Chave-3	1606.8	717.6	682.0	26.2
Floresta-Chave-4	1612.0	722.8	688.2	26.3
Floresta-Chave-5	1614.7	726.4	694.2	26.3

Tabela 5.7: Valores de λ para cada interpolação linear entre os dois *corpora* (conjuntos de teste do 5 ao 8).

VazaBarris-*Corpus*	Set5	Set6	Set7	Set8
VazaBarris-Poxim	0.16;0.84	0.16;0.84	0.11;0.89	0.15;0.85
VazaBarris-Chave	0.71;0.29	0.75;0.25	0.87;0.13	0.73;0.26
VazaBarris-Floresta	0.39;0.61	0.43;0.57	0.45;0.55	0.43;0.57
Poxim-Chave	0.91;0.09	0.95;0.05	0.97;0.03	0.96;0.04
Poxim-Floresta	0.79;0.21	0.86;0.14	0.91;0.09	0.87;0.13
Floresta-Chave	0.86;0.14	0.92;0.08	0.96;0.04	0.90;0.10

Tabela 5.8: Perplexidade para os conjuntos de testes do 5 ao 8 (interpolação).

VazaBarris-*Corpus*-N	Set5	Set6	Set7	Set8
VazaBarris-Poxim-1	145.8	119.7	118.0	140.5
VazaBarris-Poxim–2	152.0	132.5	118.3	154.5
VazaBarris-Poxim–3	153.5	133.7	117.8	152.4
VazaBarris-Poxim–4	155.1	135.1	118.6	153.4
VazaBarris-Poxim–5	156.1	136.1	119.4	154.7
VazaBarris-Chave-1	323.4	270.6	310.3	287.1
VazaBarris-Chave-2	456.4	471.6	481.8	437.7
VazaBarris-Chave-3	479.1	502.0	516.2	460.3
VazaBarris-Chave-4	491.1	519.4	531.2	476.1
VazaBarris-Chave-5	498.8	528.7	540.7	484.0
VazaBarris-Floresta-1	242.0	233.3	266.5	245.0
VazaBarris-Floresta–2	290.3	329.8	351.3	316.7
VazaBarris-Floresta–3	297.2	338.3	361.5	324.1
VazaBarris-Floresta–4	298.9	342.0	363.8	328.4
VazaBarris-Floresta–5	300.7	344.0	366.0	330.5
Poxim-Chave-1	420.9	491.1	355.4	435.5
Poxim-Chave-2	459.8	592.0	365.8	518.7
Poxim-Chave-3	457.4	590.6	362.0	506.1
Poxim-Chave-4	465.5	602.2	366.9	511.6
Poxim-Chave-5	468.0	606.8	370.1	516.9
Poxim-Floresta-1	370.9	497.1	359.4	439.6
Poxim-Floresta–2	389.6	584.5	363.1	511.7
Poxim-Floresta–3	384.7	577.4	357.8	495.7
Poxim-Floresta–4	387.7	584.2	360.7	498.5
Poxim-Floresta–5	389.1	587.5	363.1	502.4
Floresta-Chave-1	990.6	1623.7	1370.3	1154.2
Floresta-Chave-2	1590.2	3518.5	2696.9	2185.0
Floresta-Chave-3	1587.8	3475.5	2726.3	2170.7
Floresta-Chave-4	1588.1	3491.9	2726.4	2183.5
Floresta-Chave-5	1592.5	3492.2	2729.2	2187.9

Tabela 5.9: Perplexidade para os *corpora* Floresta, Poxim e Chave como conjunto de testes

VazaBarris-N	Floresta	Poxim	Chave
VazaBarris-1	1449.05	7233.80	1550.13
VazaBarris–2	1793.57	14439.00	2036.13
VazaBarris–3	1873.63	15784.10	2164.79
VazaBarris–4	2007.60	16714.20	2340.65
VazaBarris–5	2109.35	17140.8	2466.1

Tabela 5.10: Tamanho dos Corpora PTBR-NE

Corpus	Sentenças	Tokens
Paramopama	12.500	310.000
Primeiro HAREM	5.000	80.000
Segundo HAREM	3.500	100.000
WikiNER	125.821	2.830.000

Tabela 5.11: Tamanho dos Corpora PTBR-NE - (Número de entidades nomeadas)

Corpus	PER	LOC	ORG	TEMPO
Paramopama	7.327	17.461	7.154	10.827
Primeiro HAREM	2.244	2.169	2.039	914
Segundo HAREM	4.018	1.930	1.607	3.291
WikiNER	94.394	199.806	36.086	-

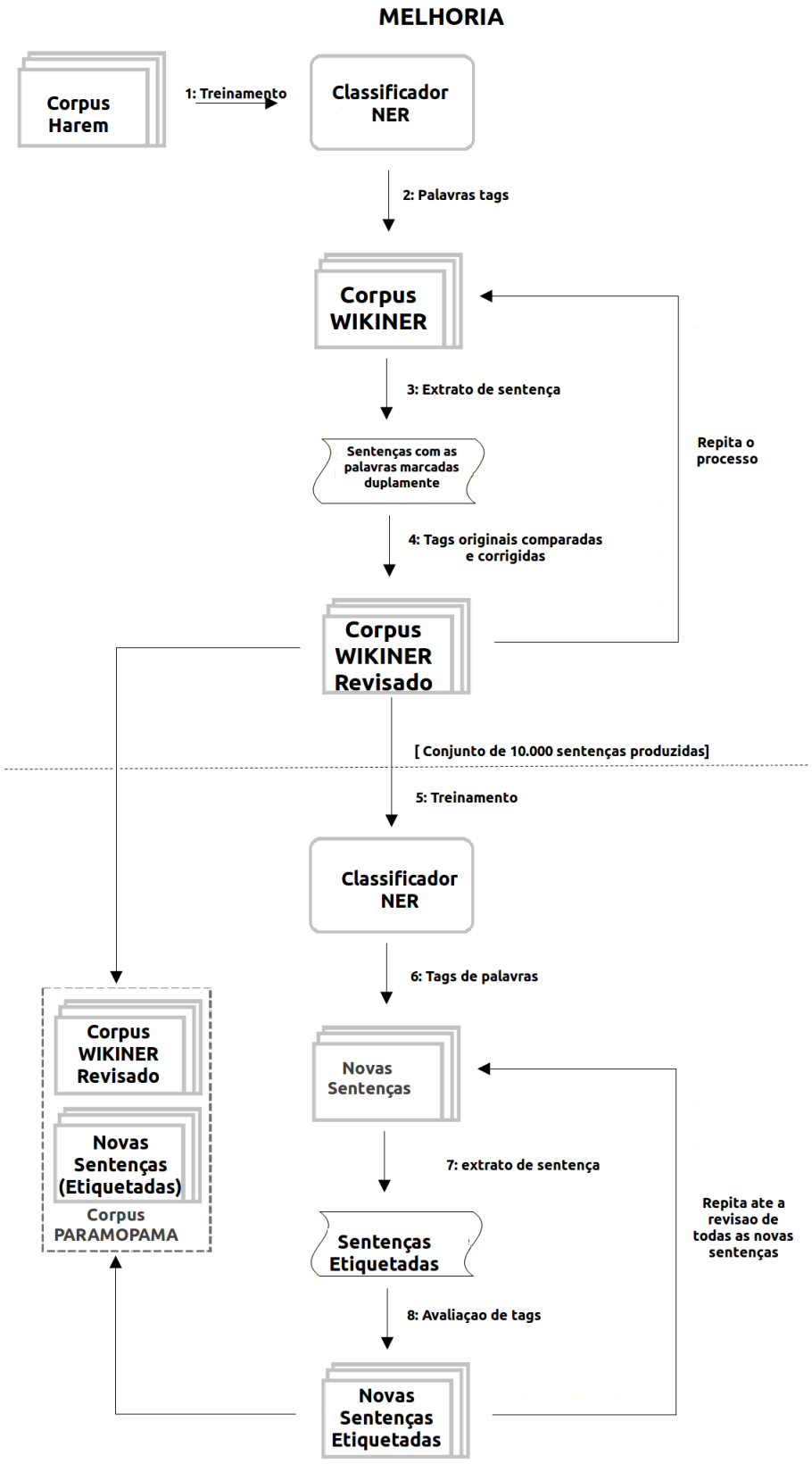

Figura 5.6: Fases de melhoria e expansão da geração corpus

Sentence 1		*Sentence 2*	
A	O	Águeda	LOCAL
bandeira	O	dista	O
nacional	O	de	O
,	O	Lisboa	LOCAL
adotada	O	240	O
em	TEMPO	km	O
1947	TEMPO	,	O
,	O	do	O
é	O	Porto	LOCAL
baseada	O	72	O
na	O	km	O
bandeira	O	e	O
do	O	de	O
Congresso	ORGANIZACAO	Aveiro	LOCAL
Nacional	ORGANIZACAO	cerca	O
Indiano	ORGANIZACAO	de	O
,	O	20	O
desenhada	O	km	O
por	O	.	O
Pingali	PESSOA		
Venkayya	PESSOA		
.	O		

Figura 5.7: Trecho do *corpus* Paramopama

Tabela 5.12: Algumas características usadas pelo classificador NER

Característica	Descrição
Order=2	O número de coisas para a esquerda que tem que ser armazenado em cache para executar o algoritmo Viterbi: o contexto máximo de características de classe usado.
Class	Coloca uma prioridade sobre as classes que equivale a quantas vezes a característica apareceu nos dados de treinamento.
PreviousWord	Dá uma característica para a palavra anterior.
Word	Dá uma característica para a palavra real.
NextWord	Dá uma característica para a próxima palavra.
UseNgrams	Faz as características em n-gramas, ou seja, substrings da palavra, com o máximo de comprimento do N- Gram de 6.
Subclassification	Converte a rotulagem das classes para uma codificação(IO), I (Inside), O (Outside) de classificação de 2 vias para cada classe.
UseGazettes	Usa dicionários para as entidades (pessoa, localização, organização).
WordShape	Característica para formas de uma palavra

Tabela 5.13: Resultados para PESSOA - Conjunto de testes 1

Corpus	Precisão Conj. de dados 1	Recall Conj. de dados 1	Medida-F Conj. de dados 1
First HAREM	76.32%	64.93%	70.16%
Second HAREM	82.55 %	84.70 %	83.61%
Full HAREM	70.03 %	77.61 %	73.63%
WikiNER	78.86 %	58.77 %	67.35%
Paramopama	86.72 %	77.99 %	82.12%
Paramopama + HAREM	80.44 %	81.34 %	80.89%

Tabela 5.14: Resultados para LOCALIDADE(Conjunto de dados 1)

Corpus	Precisão	Recall	Medida-F
First HAREM	74.48%	71.24%	72.82%
Second HAREM	68.97%	89.97%	78.08%
Full HAREM	81.16%	74.92%	77.91%
WikiNER	49.34%	73.69%	59.10%
Paramopama	71.43%	88.63%	79.10%
Paramopama + HAREM	74.51%	88.96%	81.10%

Tabela 5.15: Resultados para ORGANIZAÇÃO (Conjunto de dados 1)

Corpus	Precisão	Recall	Medida-F
First HAREM	68.61%	43.93 %	53.56%
Second HAREM	76.99 %	40.65 %	53.21 %
Full HAREM	81.42 %	42.99 %	56.27 %
WikiNER	50.15%	40.66%	44.91%
Paramopama	81.44%	63.55%	71.39%
Paramopama + HAREM	89.93%	58.41%	70.82%

Tabela 5.16: Resultados para TEMPO (Conjunto de dados 1)

Corpus	Precisão	Recall	Medida-F
First HAREM	3.48%	2.55%	2.94%
Second HAREM	88.74%	85.35%	87.01%
Full HAREM	65.94%	57.96%	61.69%
WikiNER	-	-	-
Paramopama	87.90%	87.90%	87.90%
Paramopama + HAREM	90.06%	92.36%	91.19%

Tabela 5.17: Pontuação Geral (Conjunto de dados 1)

Corpus	Precisão	Recall	Medida-F
First HAREM	63.32%	51.65%	56.89%
Second HAREM	77.18%	76.36%	76.77%
Full HAREM	74.64%	65.50%	69.77%
WikiNER	60.45%	59.44%	59.94%
Paramopama	79.91%	79.66%	79.79%
Paramopama + HAREM	81.25%	80.30%	80.77%

Os valores de precisão obtidos pelo Paramopama (Tabelas 5.13, 5.14, 5.15 e 5.16) foram acima de 80%, o que significa uma taxa baixa de falsos positivos em uma tarefa de classificação de texto, especialmente se compararmos com o *corpus* WikiNER, que alcançou precisão média de 60%.

Outro resultado importante representado nas Tabelas 5.15 e 5.20 é a melhoria do reconhecimento do termo ORGANIZAÇÃO em relação a outros *corpora* em Pt-BR: Paramopama alcançou um medida-F de quase 72% contra 56% do *corpus* HAREM.

Conforme apresentado nas Tabelas 5.16 e 5.21, o *corpus* Primeiro HAREM tinha baixa classificação de desempenho para a entidade TEMPO em ambos os conjuntos de teste. A razão é que as palavras deste corpus com a entidade TEMPO especificam apenas palavras que

Tabela 5.18: Resultados para PESSOA (Conjunto de testes 2)

Corpus	Precisão	Recall	Medida-F
First HAREM	64.11%	54.90%	59.15%
Second HAREM	77.48%	82.11%	79.73%
Full HAREM	63.05%	67.01%	64.97%
WikiNER	83.16%	68.62%	75.19%
Paramopama	83.08%	76.30%	79.55%
Paramopama + HAREM	80.74%	78.42%	79.56%

Tabela 5.19: Resultados para LOCALIDADE(Conjunto de dados 2)

Corpus	Precisão	Recall	Medida-F
First HAREM	75.68%	57.81%	65.55%
Second HAREM	77.40%	84.53%	80.81%
Full HAREM	80.35%	59.63%	68.46%
WikiNER	67.16%	91.30%	77.39%
Paramopama	78.53%	86.86%	82.49%
Paramopama + HAREM	81.69%	85.46%	83.54%

Tabela 5.20: Resultados para ORGANIZAÇÃO (Conjunto de dados 2)

Corpus	Precisão	Recall	Medida-F
First HAREM	45.68%	37.67%	41.29%
Second HAREM	75.79%	46.40%	57.56%
Full HAREM	50.87%	36.50%	42.50%
WikiNER	76.05%	45.08%	56.61%
Paramopama	70.26%	66.27%	68.21%
Paramopama + HAREM	73.03%	62.81%	67.54%

Tabela 5.21: Resultados para TEMPO (Conjunto de dados 2)

Corpus	Precisão	Recall	Medida-F
First HAREM	5.00%	3.99%	4.44%
Second HAREM	89.02%	85.75%	87.36%
Full HAREM	65.52%	54.76%	62.65%
WikiNER	-	-	-
Paramopama	92.01%	90.86%	91.43%
Paramopama + HAREM	93.24%	93.41%	93.32%

Tabela 5.22: Pontuação Geral (Conjunto de dados 1)

Corpus	Precisão	Recall	Medida-F
First HAREM	55.07%	44.35%	49.13%
Second HAREM	79.40%	79.26%	79.33%
Full HAREM	68.12%	58.00%	62.65%
WikiNER	72.53%	75.90%	74.17%
Paramopama	81.26%	81.90%	81.58%
Paramopama + HAREM	82.66%	82.02%	82.34%

representam datas. Mas nos conjuntos de teste, há muitas expressões temporais diferentes que também devem ser marcadas como TEMPO. Por exemplo, na frase "No primeiro dia do ano, Lula viajou para Brasília", a expressão "No dia Primeiro do Ano"deve ser marcada como TEMPO.

Algumas entidades aumentam as possibilidades de erro de rotulagem. Este é o caso de LOCALIZAÇÃO e ORGANIZAÇÃO. Dependendo do contexto, uma palavra pode se referir a um local ou uma organização. Os valores de precisão para classe de LOCALIZAÇÃO, por exemplo, mostram que Paramopama tem pior desempenho comparado ao HAREM, como mostram as Tabelas 5.14 e 5.19.

Esta seção apresentou o Paramopama, um *corpus* de entidade nomeada para o Português do Brasil. O Paramopama está totalmente marcado com as seguintes classes: PESSOA,

LOCALIZAÇÃO, ORGANIZAÇÃO E TEMPO. Um modelo de aprendizado foi apresentado para o Paramopama e por outros *corpora* conhecidos, e foram devidamente avaliados usando precisão, cobertura, e medida F.

5.3 Método para coleta de dados de *streaming* para modelo de linguagem

Esta seção propõe um mecanismo automático de coleta de tweets para a criação de um *corpus* de alta qualidade a partir do uso de *relevance feedback* para tarefa de previsão de palavras. Como resultado, experimentos com o mecanismo proposto foram realizados e comparados com *corpus* de dados do twitter aleatório. Esses experimentos usaram diversas alterações de parâmetros e encontrou um melhor conjunto de parâmetros para um bom desempenho da proposta.

5.3.1 Solução Proposta

O objetivo deste estudo de caso é coletar dados na Web para gerar modelos de linguagem de alta qualidade para tarefas conversacionais. Existem potencialmente na Web um conjunto grande de dados que poderiam ser utilizados para isso. No inicio deste capítulo foram realizadas uma série de análises sobre a utilidade do uso de diferentes corpora em português para a construção de modelos de linguagem para tarefas conversacionais. Um dos seus resultados mostra que textos de redes sociais, mais particularmente do Twitter, obtiveram menores valores de perplexidade para essas tarefas em comparação, por exemplo, com textos de notícias. Baseado nisso e dado nosso objetivo, vamos focar em textos do Twitter para construção de modelos de linguagem.

Uma estratégia simples para utilização de textos do Twitter para o nosso problema seria selecionar uma quantidade aleatória de mensagens (ou *tweets*) para a construção do corpus para o modelo de linguagem, similar com o que foi feito no Capítulo 5. Neste capítulo, pretendemos realizar essa seleção de forma mais elaborada, selecionando o conjunto de *tweets* que sejam mais relevantes para a nossa tarefa. Para isso propomos o uso de Relevance Feedback

5.3.2 Relevance Feedback

Relevance feedback é um processo em que as consultas de usuário em sistemas de recuperação de informação podem ser seletivamente modificadas na tentativa de recuperar os documentos mais relevantes de uma coleção (R.; B., 2008)(ROCCHIO, 1971). A consulta pode ser modificada com ajustes de pesos de termos, inclusão de novos termos ou combinação das duas abordagens. A modificação da consulta original pode envolver o usuário no processo de recuperação ou envolver a seleção automática de propriedades dos documentos recuperados (chamado de pseudo-relevance feedback). Em resumo, *Relevance Feedback* possibilita que novos documentos sejam adicionados a um resultado com base na sua similaridade para com documentos considerados como relevantes.

Porém, *relevance feedback* iterativo pode melhorar sistemas de recuperação de informação, devido ao fato de sistemas desse tipo requerer documentos julgados suficientes. Dessa forma, o termo *Pseudo-Relevance Feedback* (PRF) é uma técnica de expansão de consulta que assume que n (n>0) documentos de recuperação inicial são relevantes e a partir daí extrai os termos de expansão a partir destes documentos.

Pseudo-Relevance Feedback pode ser implementado usando o algoritmo de Rocchio (ROCCHIO, 1971; MANNING; RAGHAVAN; SCHüTZE, 2008). O algoritmo busca encontrar um vetor consulta, denotado como $\vec{q}_m$, que maximiza a semelhança com os documentos relevantes (D^r) e minimiza a semelhança com os documentos não relevantes (D^i) dado pela Equação 5.1:

$$\vec{q}_m = \alpha\vec{q}_0 + \beta\frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r}\vec{d}_j - \gamma\frac{1}{|D_i|}\sum_{\vec{d}_i \in D_i}\vec{d}_i \tag{5.1}$$

Onde α, β e γ são os pesos atribuídos a cada termo e ponderam a importância do *feedback* em relação à consulta original. A partir de q_0 a nova consulta se move para o centro dos documentos relevantes e torna-se distante do centro dos documentos considerados não relevantes.

5.3.3 Descrição da Solução

A nossa proposta é utilizar o algoritmo de Pseudo-Relevance Feedback para obter palavras que serão utilizadas como filtros de tweets para a criação de corpus para modelos de lin-

guagem. Nossa solução funciona então em duas fases. Na primeira fase (Aprendizado), a partir de uma amosta inicial de dados, ele usa *Pseudo-Relevance Feedback* para identificar as palavras a serem utilizadas como filtro, e na segunda etapa (Filtragem), aplica esse filtro sobre o restante do dados. A figura 5.8 apresenta uma visão geral dessa solução.

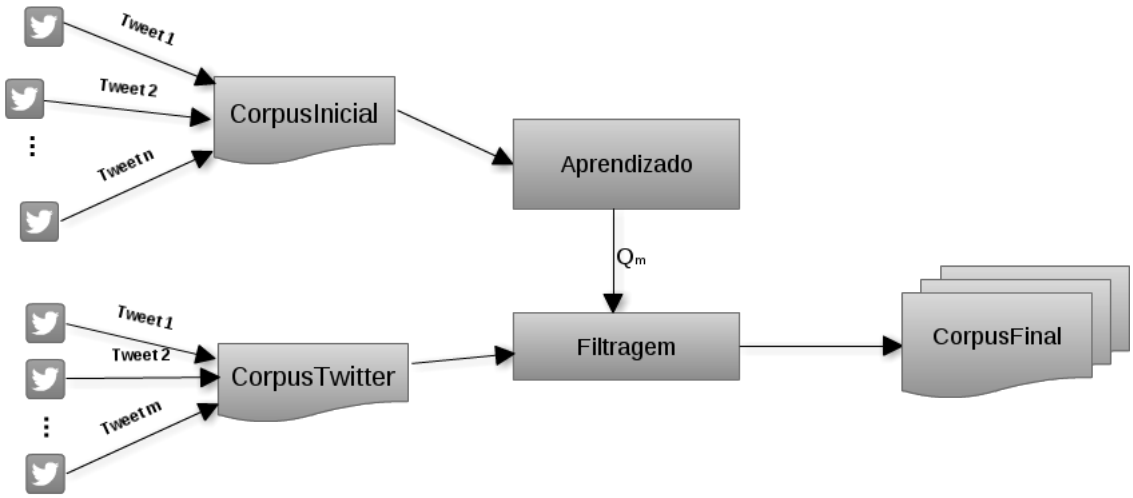

Figura 5.8: Visão geral da Solução

Na etapa de aprendizado, nós implementamos o algoritmo de Pseudo-Relevance Feedback para nosso problema. Mais especificamente, o algoritmo recebe como entrada um conjunto de tweets e um filtro de mensagens, que identifica se um tweet é relevante ou não baseado em algum critério. Na nossa implementação, esse filtro de mensagens é criado medindo-se a perplexidade do tweet em relação a uma pequena amostra de tweets rotulados como relevantes. Então, um tweet é considerado relevante se sua perplexidade em relação a essa amostra for menor que um certo limiar que definimos empiricamente. Tweets considerados relevantes são adicionados ao vetor relevante, $\vec{V_r}$, e os não relevantes ao vetor irrelevante $\vec{V_{irr}}$. A partir deles, calcula-se a similaridade entre cada documento recuperado pelo algoritmo de Rocchio dado na Equação 5.1 onde $\alpha = 0$, $\beta = 1$ e $\gamma = 1$. A partir do vetor resultante, selecionam-se a k palavras com maior peso para serem utilizadas na etapa posterior. Finalmente, na etapa de Filtragem essas k palavras juntamente com o filtro de mensagens, utilizados no Learning, são utilizados para filtrar os tweets que serão utilizados para a criação do corpus.

Algoritmo 2: Implementação do *Pseudo-Relevance FeedBack*

Entrada: CorpusInicial, N, Z
Saída: $< Q^m >$ {Vetor de palavras com maiores pesos}
início

inicialização;
Seja N o número máximo de iteração na busca;
Seja Z o número de palavras necessárias ao vetor $< Q^m >$;
Seja $Filtro(tweet)$ o filtro utilizado para classificação do tweet;
Seja $\alpha = 1$, $\beta = 1$ e $\gamma = 1$;
$V^r \leftarrow \emptyset$;
$V^i \leftarrow \emptyset$;
$j \leftarrow 0$;
repita
  se $Filtro(tweet)$ **então**
    $V^r \leftarrow V^r + vetorizacao(tweet)$;
  fim
  senão
    $V^i \leftarrow V^i + vetorizacao(tweet)$;
  fim
  j++;
até *j = N* ;
para cada $Palavra \in V^r$ **faça**
  $Palavra \leftarrow (\alpha.Palavra) + (\beta.V^r) - (\gamma.V^i)$;
fim
$Q^m \leftarrow Z$ primeiras Palavras;
return $< Q^m >$
fim

5.3.4 Avaliação Experimental

Para realizar a avaliação utilizamos o corpus Poxim, que é um corpus construído pelo rastreamento de tweets. Ele consiste de cerca de 300 mil tweets rastreados, aproximadamente 3 milhões de palavras. Poxim foi coletado utilizando o procedimento de *streaming*, permitindo que os textos fossem capturados de forma constantes à medida que fossem lançados na base

de dados do Twitter e disponibilizados na web. A coleta dos dados foi feita no período de Fevereiro e Março de 2015 e o corpus está disponível em [9].

Para os nossos experimentos foram utilizados os 100 mil primeiros tweets do corpus Poxim. Estas mensagens foram submetidas a um procedimento de limpeza, incluindo remoção de *links* de URLs, nomes de usuários e hashtags.

Para medir a perplexidade de cada *tweet* na etapa de aprendizado foi criado um modelo de linguagem para cada um utilizando SRILM (STOLCKE, 2002). A perplexidade foi computada por um conjunto de teste de diálogos em redes sociais, e-mails e blogs, que foram gentilmente cedidas por voluntários, estes conjuntos estão disponíveis em [10]

O desempenho de cada abordagem depende de algumas características no algoritmo proposto. Que estão listadas abaixo:

- C1: Número de tweets do corpus usados para a fase de Aprendizado;
- C2: Número de termos selecionados pela fase de Aprendizado;
- C3: Limite máximo de perplexidade $Per(max)$ para o tweet ser considerado relevante ou não;

A Tabela 5.23 mostra 3 configurações da nossa abordagem com seus respectivos valores para esses parâmetros. Para cada uma delas, foram selecionados 1000 tweets do corpus resultante da fase de Filtragem.

5.3.5 Número de tweets do corpus usados para a fase de Aprendizado

Para o primeiro experimento, nós variamos o tamanho do corpus usado para treinamento. De acordo com a figura 5.9, o melhor número de tweets para obtenção do conjunto de documentos relevantes foi um total de 1000 tweets.

5.3.6 Número de termos selecionados pela fase de Aprendizado

Nesse experimento nós variamos o número de termos selecionados do vetor final, resultante da combinação do vetor positivo (tweets relevantes) e negativo (tweets não relevantes). A fi-

[9] https://www.dropbox.com/s/fc1ckak8rkvgkgz/Poxim.rar?dl=0

[10] Conjunto de teste para o cálculo da perplexidade: https://www.dropbox.com/sh/g9m7frwedcvmwh5/AACxD7MpGWJomxo FGSq6Ua?dl=0

Nome	C2	C3	C4
RL1	1000	100	10.0
RL2	10000	100	10.0
RL3	1000	1000	10.0
5RL4	10000	1000	10.0
RL5	5000	100	10.0
RL6	1000	50	10.0
RL7	1000	500	10.0
RL8	1000	100	50.0
RL9	1000	100	100.0

Tabela 5.23: Configurações de diferentes valores de parâmetros

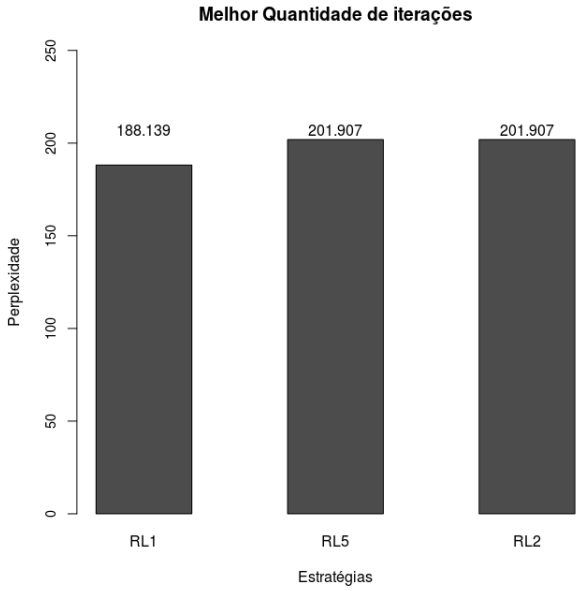

Figura 5.9: Melhor quantidade de iterações

gura 5.10 mostra que o conjunto menor de termos selecionados na fase aprendizado melhora a perplexidade do corpus final, devido ao motivo que quanto maior o conjunto de termos, mais palavras do conjunto de documentos não-relevantes serão consideradas.

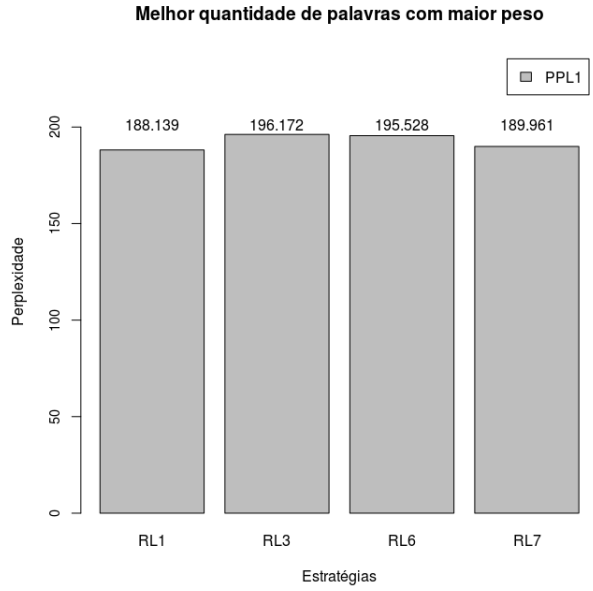

Figura 5.10: Número de termos selecionados pela fase de Aprendizado

5.3.7 Limite de perplexidade

Nesse experimento nós variamos o número do limite de perplexidade. A figura5.11 mostra que o conjunto com o menor limite de perplexidade de um *tweet* melhora a perplexidade final do *corpus*.

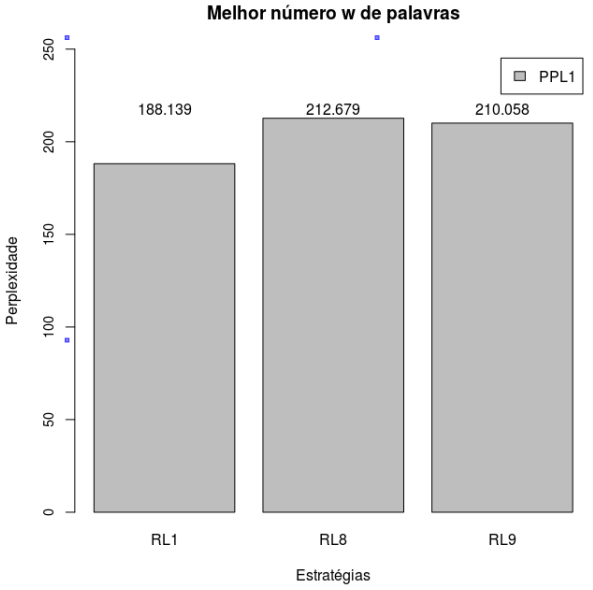

Figura 5.11: Melhor limite de Perplexidade

5.3.8 Uso de Emoticons nos Dados para Aprendizado

Uma característica particular ao Twitter é o uso de emoticons e emojis pelos usuários. A figura 5.12 apresenta os valores de perplexidade do corpus resultante para as 3 configurações com o uso ou não de emoticons na fase de Aprendizado. Para todas as configurações, o Aprendizado sem emoticons obtêm menores valores de perplexidade. Isso significa que emoticons introduzem ruído na geração das palavras na fase de Aprendizado.

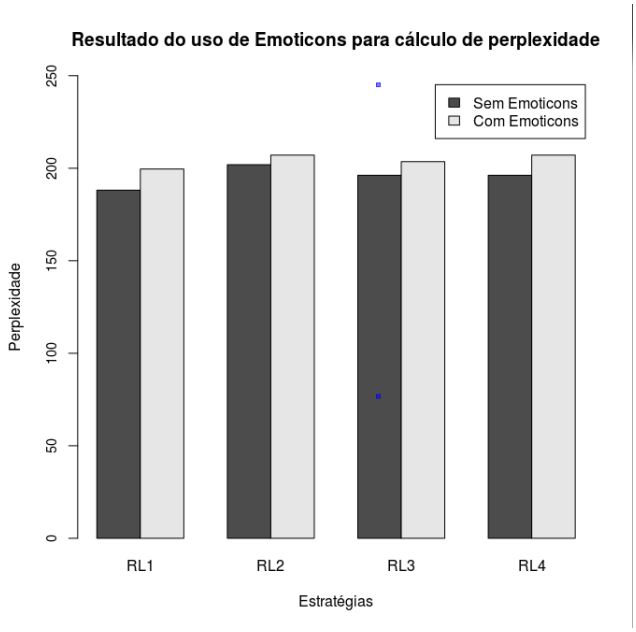

Figura 5.12: Uso de emoticons

5.3.9 Comparação com Baselines

Entre todos os resultados apresentados o que teve um melhor efeito na perplexidade foi o conjunto de características apresentado na Tabela 5.24. O melhor número de tweets do corpus usados para a fase de aprendizado foi de 1000 tweets, enquanto que o número de termos selecionados pela fase de aprendizado foi o conjunto de 100 termos, pois quanto mais termos o conjunto tinha mais palavras dos documentos não-relevantes eram consideradas. Por último, o melhor limite de perplexidade encontrado foi 10.0.

A Tabela 5.25 apresenta um novo teste comparando os resultados do melhor conjunto de características com o corpus geral. Foram feitos três conjuntos para efeito de comparação ser melhor. O primeiro foram pegos 1000 tweets aleatórios do *corpus* gerado pelo melhor

Nome	C1	C2	C3
RL1	1000	100	10.0

Tabela 5.24: Melhor Conjunto de características

conjunto. O segundo foram pegos 1000 tweets aleatórios do corpus Poxim disponível em [11]. Para o terceiro conjunto foram utilizados 1000 tweets aleatórios com perplexidade menor que 10.0. A Tabela 5.25 mostra que o melhor conjunto de características apresenta uma melhor perplexidade comparando às outras baselines.

Nome	PPL1	PPL2
Melhor RL	188.139	885.939
Tweets aleatórios	217.686	948.053
Tweets aleatórios menos que 10.0	206.049	1002.13

Tabela 5.25: Resultado final

5.4 Exemplo prático de aplicação

Um exemplo prático de aplicação para estes *corpora* ou outros que possam vir a ser gerados com uso da parametrização aqui proposta é a previsão de palavras e sentenças orientada a ajudar pessoas com paralisia cerebral, em particular, crianças com tetraplegia espótica.

Vários trabalhos utilizam esforços computacionais para a *Alternative and Augmented Communication*(AAC), Porém devem lidar com duas questões importantes: a primeira é a necessidade de hardware e software específico, que possa envolver diferentes tipos de deficiência e o segundo são as diferentes linguagens de comunicação que estes trabalhos tem que lidar.

Porém, os sistemas descritos por (SANTOS, 2014) tem algumas limitações para o uso como uma ferramenta de AAC personalizada para o Brasil. Entre estas limitações é incluída

[11] https://www.dropbox.com/s/fc1ckak8rkvgkgz/Poxim.rar?dl=0

a ausência de *corpora* em português que auxilie ferramentas para previsão de palavras e sentenças para auxiliar a comunicação de pessoas com paralisia cerebral.

Dessa forma, este trabalho propõe auxiliar o software/hardware PickNClick (MACEDO, 2014) que usa a previsão de palavras e frases fornecidas pelo modelo N-gram, a fim de promover uma AAC para pessoas com deficiências. O *focused web crawler* proposto apresentará *corpora* específico que auxilie a ferramenta PickNClick.

PickNClick utiliza modelagem N-gram para prever palavras e permite utilizar tanto uma base de dados genérica como uma personalizada por usuário. A entrada da ferramenta é dada por meio de uma lista de palavras e frases mais prováveis de serem apresentadas pelo usuário.

A entrada de dados é ativada por um dispositivo conectado que detecta a mordida de um usuário. Em seguida, PickNclick recebe os valores e envia mensagem ao módulo de processamento e interface gráfica para ocorrer a comunicação.

O PickNclick teve uma extensão e se tornou o CA2JU ACELERADO e o CA2JU ILUSTRADO. O CA2JU ACELERADO tem como objetivo acelerar a escrita de crianças com paralisia cerebral. A aceleração da escrita é feita através da técnica de previsão N-Gram e Hidden Markov Models (HMM), com o auxílio de uma base textual inicial para efetuar o treinamento do modelo de previsão.

O CA2JU ILUSTRADO tem o objetivo de acelerar a escrita com a ajuda de figuras para a criação de frases. Permite ao usuário selecionar pictogramas para criar frases através da conversão de sequências de imagens em frases escritas. Ele executa esta tarefa com diferentes técnicas de PLN como reconhecimento de entidades nomeadas.

Um exemplo prático de aplicação para o *corpus* Paramopama, um corpus rotulado com entidades nomeadas, pode ser a utilização do *corpus* para treinamento de técnicas de aprendizado de máquina para o reconhecimento de diversas informações presentes num texto, como dados biomédicos, entre eles proteínas, dados de DNA e RNA e tipos de células em textos relativos a área médica. Também pode ser usado para detectar e classificar nomes de pessoas, locais, organizações e informações de tempo em um texto, que podem ser usadas em aplicativos de QA, análise de sentimento e sumarização.

Capítulo 6

Ferramenta Web para geração multi-parametrizada de *corpora* linguísticos

No Capítulo 5 foram descritas três operações de criação e avaliação de *corpora*: um *corpus* web, denominado VazaBarris, um *corpus* de dados do twitter, denominado Poxim, e um *corpus* para reconhecimento de entidades nomeadas denominado de Paramopama.

Este capítulo apresenta a ferramenta para geração multi-parametrizada de *corpora* linguístico que reflete a proposta de parametrização descrita no capítulo 4, automatiza o processo de geração de *corpora* linguísticos e torna o usuário interessado o ator central deste processo.

A seção 6.1 apresenta os requisitos necessários na ferramenta. A seção 6.2 apresenta a demonstração da ferramenta de gerenciamento.

6.1 Requisitos da ferramenta de gerenciamento

A ferramenta para geração multi-parametrizada de *corpora* linguístico deve respeitar uma série de pré-requisitos para conceber um *corpus* que atenda as demandas do usuário interessado.

A seguir temos uma lista de necessidades que são atendidas pela ferramenta:

- Usuário define como os dados serão rastreados, se por páginas web ou por dados do twitter;
- Usuário define a língua desejada ao *corpus*;
- Usuário define o modo como o rastreamento iniciará, se por palavras-chaves ou por URLs sementes (no caso de um web *crawler*) ou via *streamming* de dados (no caso de um crawler com dados do twitter);
- Usuário define a utilização de domínio no rastreamento de dados;
- Usuário delimita o tamanho dos dados;
- Usuário define o uso de sinônimos quando o rastreamento iniciar por meio de palavras-chave;

6.2 Demonstração da ferramenta de gerenciamento

Veremos a seguir as funções da ferramenta web. Serão vistos os principais módulos da ferramenta e como o usuário deve proceder para realizar as principais operações envolvendo a construção do *corpus*.

6.2.1 Módulo de Coleta

O Módulo de Coleta, também chamado de *crawler*, é responsável pela aquisição do *corpus* inicial, a coleta pode ser dada por páginas Web ou por dados do twitter.

Caso a escolha do usuário seja por um *corpus* com dados da web, este pode ser baseado em uma lista contendo URLs iniciais dadas pela escolha de um domínio no diretório DMOZ. Em seguida, o módulo busca em cada página visitada referências para outras páginas. A coleta dos dados também pode ser realizada por meio de um conjunto de palavras-chave (o usuário pode solicitar o uso de sinônimos para cada palavra), o qual será feito uma busca no Bing e as URLs resultantes serão as URLs iniciais do crawler.

Caso a escolha do usuário seja por um *corpus* com dados retirados do twitter, a seleção dos dados pode ser dado pela *Streaming* API do Twitter4j, onde será coletados todos os tweets em português que estão sendo disponibilizados na base de dados do Twitter. Ou pela

coleta dos dados por um conjunto de palavras-chave, onde irá coletar somente os tweets que tenham estas palavras.

A figura 6.1 apresenta a seleção das configurações iniciais para o módulo de coleta.

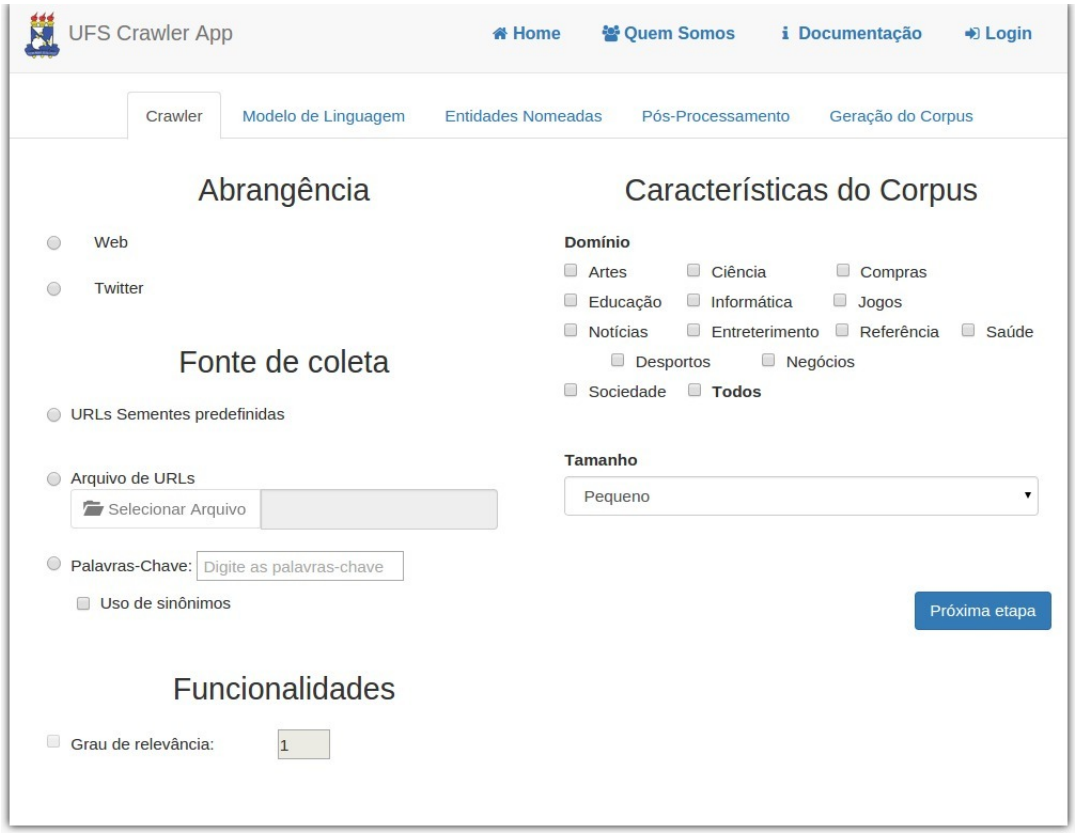

Figura 6.1: Tela para coleta de informações do futuro *corpus*

6.2.2 Módulo de módelo de Linguagem

O Módulo de Modelo de linguagem recebe as páginas coletadas pelo módulo de busca e cria um modelo de linguagem baseado no *corpus* coletado. A ferramenta aborda a técnica dos N-grams com N variando de 1 até 5. Caso o usuário queira que a ferramenta calcule o modelo de linguagem, é necessário que envie um arquivo de teste.

A figura 6.2 apresenta a seleção das configurações para a construção do modelo de linguagem do *corpus*.

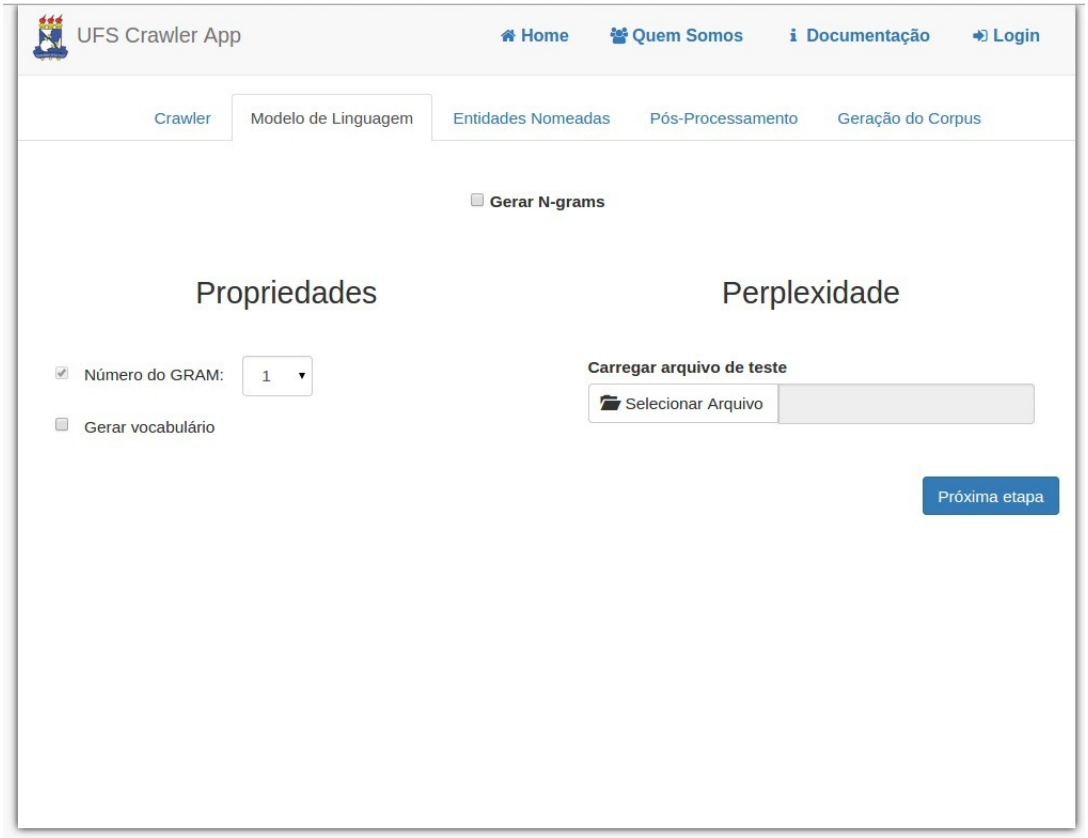

Figura 6.2: Tela para módulo de construção do modelo de linguagem

6.2.3 Módulo de Entidades Nomeadas

O módulo de Entidades Nomeadas recebe as páginas coletadas pelo módulo de busca e cria um modelo de entidade nomeada baseado no *corpus* coletado. A ferramenta baseou-se no modelo de aprendizagem *Stanford Named Entity Recognizer* (NER) (FINKEL; GRENAGER; MANNING, 2005). E foram utilizadas as seguintes entidades nomeadas: pessoa, organização, localização, tempo e outros.

A figura 6.4 apresenta a seleção das configurações para a construção do modelo de entidades nomeadas do *corpus*.

6.2.4 Módulo de Pós-Processamento

O módulo de pós-processamento tem três opções para o usuário: (i) Limpar caracteres especiais, (ii) eliminar pontuação e (iii) eliminar stopwords. A seção4.2 apresenta como esse módulo se comporta.

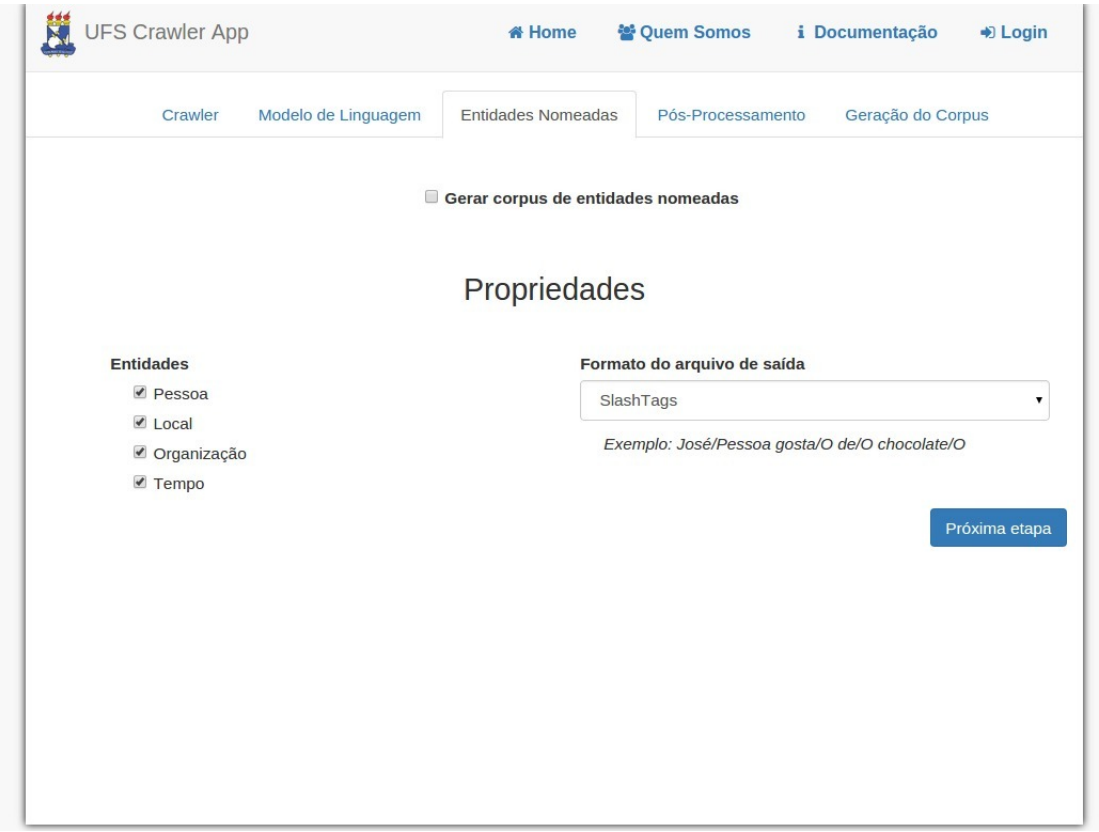

Figura 6.3: Tela para módulo de construção do modelo de entidades nomeadas

A figura 6.4 apresenta a seleção do pós-processamento do *corpus*.

6.3 Implementação

O módulo de coleta (*crawler*) está dividido em duas partes: a coleta de dados da *web* e coleta de dados do Twitter. Para implementação do módulo de coleta dos dados da *web* foi utilizado o *crawler* Memex (CHAKRABARTI; BERG; DOM, 1999b), conforme explicado na seção5.1. O Memex, projeto criado pela *Continuum Analytics*, é uma aplicação web que fornece interfaces fáceis de usar para os dados de coleta, análise e gráficos web de rastreamento. A aplicação de coleta foi desenvolvida em JAVA e apresenta simplicidade na atividade de modificação do seu código para adequação do comportamento as necessidades de coleta.

A implementação do módulo de coleta de dados do twitter utilizou Java e a API Twitter4j, uma API interface Java para publicar Twitter. A biblioteca language-Detection (SHUYO,

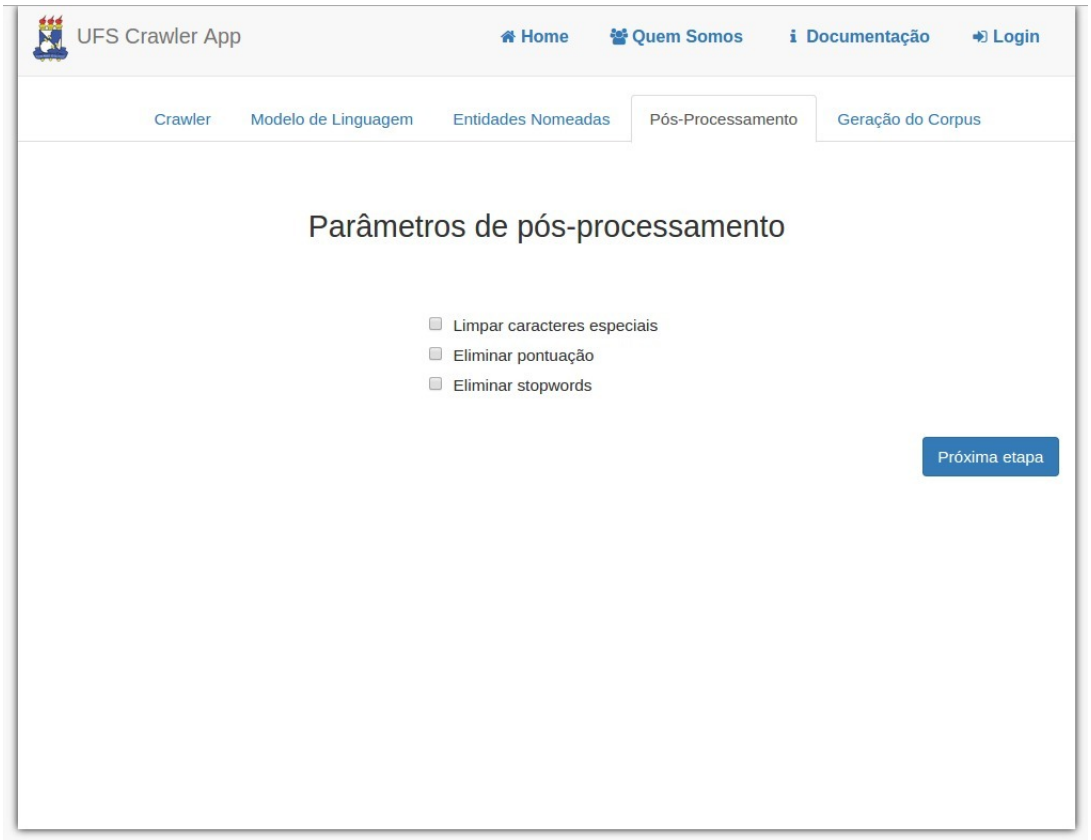

Figura 6.4: Tela para módulo de seleção do Pós-processamento do *Corpus*

2010) foi usada para identificar automaticamente *tweets* em Português.

O modelo de linguagem do *corpus* resultante da primeira etapa foi implementado utilizando a ferramenta SRILM, que é uma ferramenta para a construção e aplicação de modelos de linguagens estatísticos e foi desenvolvida no laboratório SRI *Speech Technology and Research Laboratory*.

Foi utilizado a implementação em java *Stanford* NER para o reconhecimento de entidades nomeadas. *Named Entity Recognition* (NER) rotula sequências de palavras em um texto, como pessoas e nomes de empresas. Stanford NER é um classificador em sequência que implementa o Conditional Random Fields (CRF).

Finalmente, para a etapa de limpeza do *corpus* foram utilizados algoritmos de processamento de linguagem natural em Python/NLTK. *Natural Language Toolkit* (NLTK) (LOPER; BIRD, 2002) é uma bilioteca em python que trabalha com a construção de programas de PLN e oferece mais de 50 corpora e recursos lexicais.

6.4 Ambiente de experimentação

A ferramenta está disponível para consulta executando em localhost. Para ter acesso a ferramenta é necessário pedir aos desenvolvedores.

Capítulo 7

Conclusão

Esta dissertação de mestrado tratou da proposta de elementos para parametrização das etapas de pré-processamento e pós-processamento no processo de geração automática de *corpora* linguísticos a partir de dados textuais abundantemente disponíveis na Internet. Com estes elementos de parametrização definidos claramente, o usuário interessado pode se tornar ator central do processo de geração desses *corpora* enviesando conscientemente esta geração de modo a garantir que o *corpus* gerado atenda exatamente a seus desejos de pesquisa e/ou desenvolvimento. Estes elementos foram incorporados no desenvolvimento de uma ferramenta web a ser diponibilizada publicamente para fins de pesquisa.

Os quatro objetivos específicos inicialmente traçados foram atendidos da seguinte forma.

1. Propor elementos de parametrização para as etapas de pré-processamento e de pós-processamento na geração de *corpora* linguísticos. Elementos foram propostos e estão relacionados no capítulo 4.

2. Mostra a efetividade da proposta de parametrização na concepção de três diferentes *corpora* da língua portuguesa: (i) *corpus* com dados provenientes de páginas Web, (ii) *corpus* com dados provenientes do microblogging Twitter e (iii) *corpus* anotado para reconhecimento de entidades nomeadas. Variando-se elementos de parametrização, foram concebidos os *corpora* VazaBarris, Poxim e Paramopama.

3. Avaliar os *corpora* gerados quanto à qualidade de seu modelo de linguagem. Os *corpora* VazaBarris e Poxim foram devidamente avaliados com a métrica perplexidade do modelo de linguagem enquanto o *corpus* Paramopama foi avaliado com respeito

as métricas de precisão, cobertura e medida-F; as avaliações mostram resultados superiores ao Estado da Arte. A partir da análise dos resultados obtidos com o *corpus* Poxim, em especial, um estudo mais pormenorizado sobre manipulação de dados de streaming do Twitter foi conduzido, o que culminou com a proposta de um método de coleta automática de dados baseado em *relevance feedback* para a tarefa de previsão de palavras, devidamente validado; e

4. Desenvolver uma ferramenta que reflita os elementos de parametrização propostos, automatize o processo de geração de *corpora* e possa ser utilizada de forma online por pesquisadores, estudantes ou interessados em Linguística de *Corpus* de uma maneira geral. A ferramenta foi criada e está disponível online.

Ao atender aos quatro objetivos traçados, esta dissertação contribui de forma prática para as pesquisas em Linguística Computacional para a lingua portuguesa do Brasil.

Trabalhos futuros envolvem a devida validação do uso da ferramenta, que pode ser realizada através da aplicação, coleta e avaliações de formulários online preenchidos por uma boa amostra de usuários voluntários. Outro trabalho futuro é a utilização dos *corpora* concebidos em estudos de caso reais. Um desses ja vem sendo conduzido no âmbito da criação de uma ferramenta terapêutica para Comunicação Alternativa Ampliada, em parceria com o Departamento de Fonoaudiologia da UFS.

REFERÊNCIAS

AGGARWAL, C. C.; AL-GARAWI, F.; YU, P. S. Intelligent crawling on the world wide web with arbitrary predicates. In: *Proceedings of the 10th International Conference on World Wide Web*. New York, NY, USA: ACM, 2001. (WWW '01), p. 96–105.

ALMEIDA CAMPOS, M. L. de; GOMES, H. E. Taxonomia e classificação: o princípio de categorização. *DataGramaZero-Revista de Ciência da Informação*, v. 5, p. 9, 2010.

ALMPANIDIS, G.; KOTROPOULOS, C.; PITAS, I. Combining text and link analysis for focused crawling-an application for vertical search engines. *Inf. Syst.*, Oxford, UK, UK, p. 886–908, 2007. ISSN 0306-4379.

Assis, G. T. d. *Uma abordagem baseada em gênero para coleta temática de páginas da web*. Tese (Doutorado) — Universidade Federal de Minas Gerais, 2008.

BAEZA-YATES, R. A.; RIBEIRO-NETO, B. *Modern Information Retrieval*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

BARBOSA, L.; FREIRE, J. An adaptive crawler for locating hidden-web entry points. In: ACM. *Proceedings of the 16th international conference on World Wide Web*. [S.l.: s.n.], 2007. p. 441–450.

BATSAKIS, S.; PETRAKIS, E. G.; MILIOS, E. Improving the performance of focused web crawlers. *Data Knowledge Engineering*, v. 68, n. 10, p. 1001 – 1013, 2009.

BERBER SARDINHA, T.; MOREIRA FILHO, J.; ALAMBERT, E. O corpus brasileiro. *Comunicaçao ao VII Encontro de Lingüıstica de Corpus*, 2008.

BICK, E. *The parsing system"Palavras": Automatic grammatical analysis of Portuguese in a constraint grammar framework*. [S.l.]: Aarhus Universitetsforlag, 2000.

BICK, E. et al. Floresta sintá (c) tica: Ficção ou realidade. *Avaliação conjunta: um novo paradigma no processamento computacional da língua portuguesa*, p. 291–300, 2007.

BOOS, R. et al. brwac: A wacky corpus for brazilian portuguese. In: BAPTISTA, J. et al. (Ed.). *Computational Processing of the Portuguese Language*. [S.l.]: Springer International Publishing, 2014, (Lecture Notes in Computer Science, v. 8775). p. 201–206.

BORGES, L. d. O.; MOURÃO, L. *Recuperação de Informação-: Conceitos e Tecnologia das Máquinas de Busca*. [S.l.]: Bookman Editora, 2013.

BRA, P. et al. Information retrieval in distributed hypertexts. In: *In RIAO*. [S.l.: s.n.], 1994. p. 481–491.

BRIN, S.; PAGE, L. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 30, n. 1-7, p. 107–117, abr. 1998. ISSN 0169-7552.

CHAKRABARTI, S.; BERG, M. v. d.; DOM, B. Distributed hypertext resource discovery through examples. In: *Proceedings of the 25th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. (VLDB '99), p. 375–386.

CHAKRABARTI, S.; BERG, M. van den; DOM, B. Focused crawling: A new approach to topic-specific web resource discovery. In: *Proceedings of the Eighth International Conference on World Wide Web*. New York, NY, USA: Elsevier North-Holland, Inc., 1999. (WWW '99), p. 1623–1640.

CHAKRABARTI, S.; PUNERA, K.; SUBRAMANYAM, M. Accelerated focused crawling through online relevance feedback. In: *Proceedings of the 11th International Conference on World Wide Web*. New York, NY, USA: ACM, 2002. (WWW '02), p. 148–159.

CHINCHOR, N. Muc-7 test scores introduction. In: *Proceedings of the seventh message understanding conference*. [S.l.: s.n.], 1998.

CHO, J.; GARCIA-MOLINA, H.; PAGE, L. Efficient crawling through url ordering. *Comput. Netw. ISDN Syst.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 30, n. 1-7, p. 161–172, abr. 1998. ISSN 0169-7552.

DEERWESTER, S. C. et al. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 1990.

DILIGENTI, M. et al. Focused crawling using context graphs. In: *Proceedings of the 26th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000. (VLDB '00), p. 527–534.

DODDINGTON, G. R. et al. The automatic content extraction (ace) program-tasks, data, and evaluation. In: *LREC*. [S.l.: s.n.], 2004.

DYBKJAER, L.; HEMSEN, H.; MINKER, W. *Evaluation of text and speech systems*. [S.l.]: Springer Science & Business Media, 2007.

ELYASIR, A. M. H.; ANBANANTHEN, K. S. M. Focused web crawler. In: *Proc. Intl. Conf. on Information and Knowledge Management (CIKM)*. [S.l.: s.n.], 2012.

ESTER, M.; KRIEGEL, H.-P.; SCHUBERT, M. Accurate and efficient crawling for relevant websites. In: *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*. [S.l.]: VLDB Endowment, 2004. (VLDB '04), p. 396–407.

FERNEDA, E. *Recuperação de Informação: Análise sobre a contribuição da Ciência da Computação para a Ciência da Informação*. Tese (Doutorado) — Universidade de São Paulo, 2003.

FETTERLY, D.; CRASWELL, N.; VINAY, V. Measuring the search effectiveness of a breadth-first crawl. In: BOUGHANEM, M. et al. (Ed.). *Advances in Information Retrieval*. [S.l.]: Springer Berlin Heidelberg, 2009, (Lecture Notes in Computer Science, v. 5478). p. 388–399.

FINKEL, J. R.; GRENAGER, T.; MANNING, C. Incorporating non-local information into information extraction systems by gibbs sampling. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. [S.l.: s.n.], 2005. p. 363–370.

FREITAS, C. et al. Vampiro que brilha... rá! desafios na anotaçao de opiniao em um corpus de resenhas de livros. *ENCONTRO DE LINGUÍSTICA DE CORPUS*, v. 11, 2012.

HEARST, M. A. Untangling text data mining. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. [S.l.: s.n.], 1999. p. 3–10.

HERSOVICI, M. et al. The shark-search algorithm. an application: Tailored web site mapping. *Comput. Netw. ISDN Syst.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 30, n. 1-7, p. 317–326, abr. 1998. ISSN 0169-7552.

HU, W.-C. et al. An overview of world wide web search technologies. 2001.

KLEINBERG, J. M. Authoritative sources in a hyperlinked environment. *J. ACM*, ACM, New York, NY, USA, v. 46, n. 5, p. 604–632, set. 1999. ISSN 0004-5411.

KOEHN, P. et al. Moses: Open source toolkit for statistical machine translation. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*. [S.l.: s.n.], 2007. p. 177–180.

KURAMOTO, H. Sintagmas nominais: uma nova proposta para a recuperação de informação. IASI, 2002.

LAFFERTY, J.; MCCALLUM, A.; PEREIRA, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

LARANJEIRA, B. et al. Comparing the quality of focused crawlers and of the translation resources obtained from them. In: CHAIR), N. C. C. et al. (Ed.). *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Reykjavik, Iceland: European Language Resources Association (ELRA), 2014.

LIU, H.; JANSSEN, J.; MILIOS, E. Using hmm to learn user browsing patterns for focused web crawling. *Data Knowl. Eng.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 59, n. 2, p. 270–291, nov. 2006. ISSN 0169-023X.

LOPER, E.; BIRD, S. Nltk: The natural language toolkit. In: *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*. [S.l.: s.n.], 2002. p. 63–70.

MACEDO, H. et al. Patent. Picknclick v1.0. 2014. 07 2014.

MANNING, C. D.; RAGHAVAN, P.; SCHüTZE, H. *Introduction to Information Retrieval*. Cambridge: Cambridge University Press, 2008.

MARTIN, J. H.; JURAFSKY, D. Speech and language processing. *International Edition*, 2000.

MCCALLUM, A.; LI, W. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. [S.l.: s.n.], 2003. p. 188–191.

MCCALLUM, A. et al. A machine learning approach to building domain-specific search engines. In: *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. (IJCAI'99), p. 662–667.

MCCALLUMZY, A. et al. Building domain-specific search engines with machine learning techniques. 1999.

MEDEIROS CASELI, H. de. Indução de léxicos bilíngües e regras para a tradução automática. 2007.

MENCZER, F. Arachnid: Adaptive retrieval agents choosing heuristic neighborhoods for information discovery. 1997.

NAJORK, M.; WIENER, J. L. Breadth-first crawling yields high-quality pages. In: *Proceedings of the 10th International Conference on World Wide Web*. New York, NY, USA: ACM, 2001. (WWW '01), p. 114–118.

OLSTON, C.; NAJORK, M. Web crawling. *Found. Trends Inf. Retr.*, Now Publishers Inc., Hanover, MA, USA, v. 4, n. 3, p. 175–246, mar. 2010. ISSN 1554-0669.

PIRINEN, T.; LINDÉN, K. et al. Finite-state spell-checking with weighted language and error models. In: *Proceedings of LREC 2010 Workshop on creation and use of basic lexical resources for less-resourced languages*. [S.l.: s.n.], 2010.

PRASATH, R.; OZTüRK, P. Finding potential seeds through rank aggregation of web searches. In: *Proceedings of the 4th International Conference on Pattern Recognition and Machine Intelligence*. Berlin, Heidelberg: Springer-Verlag, 2011. (PReMI'11), p. 227–234.

PRIYATAM, P. N. et al. Seed selection for domain-specific search. In: *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2014. (WWW Companion '14), p. 923–928.

R., B.; B., R. *Modern Information Retrieval*. [S.l.]: Addison-Wesley Publishing Company, 2008.

RENNIE, J.; MCCALLUM, A. Using reinforcement learning to spider the web efficiently. In: *Proceedings of the Sixteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. (ICML '99), p. 335–343.

ROBERTSON, S. E.; SPARCK JONES, K. Document retrieval systems. In: WILLETT, P. (Ed.). London, UK, UK: Taylor Graham Publishing, 1988. cap. Relevance Weighting of Search Terms, p. 143–160.

ROCCHIO, J. J. Relevance feedback in information retrieval. Prentice-Hall, Englewood Cliffs NJ, 1971.

ROCHA, P.; SANTOS, D. Cetempúblico: Um corpus de grandes dimensões de linguagem jornalística portuguesa. *Actas do V Encontro para o processamento computacional da língua portuguesa escrita e falada, PROPOR*, v. 2000, p. 131–140, 2000.

ROCHA, P.; SANTOS, D. Clef: Abrindo a porta à participa internacional em avaliação de ri do português. *Avaliação conjunta: um novo paradigma no processamento computacional da língua portuguesa. IST Press, Lisbon*, 2006.

SANTOS, D.; CARDOSO, N. A golden resource for named entity recognition in portuguese. In: *Computational Processing of the Portuguese Language*. [S.l.]: Springer, 2006. p. 69–79.

SANTOS, D.; ROCHA, P. The key to the first clef with portuguese: topics, questions and answers in chave. In: *Multilingual Information Access for Text, Speech and Images*. [S.l.]: Springer, 2005.

SANTOS, F. et al. Computational system for alternative and augmented communication for people with cerebral palsy. In: *Proceedings of the 7th Euro American Conference on Telematics and Information Systems*. New York, NY, USA: ACM, 2014. (EATIS '14), p. 39:1–39:2.

SARDINHA, T. B. Linguística de corpus: historico e problemática. scielo, v. 16, p. 323 – 367, 00 2000. ISSN 0102-4450.

SETTLES, B. Biomedical named entity recognition using conditional random fields and rich feature sets. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*. [S.l.: s.n.], 2004. p. 104–107.

SHUYO, N. Language detection library for java. 2010.

SPARCK JONES, K. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, MCB UP Ltd, v. 28, n. 1, p. 11–21, 1972.

STOLCKE, A. Srilm - an extensible language modeling toolkit. In: . [S.l.: s.n.], 2002. p. 901–904.

TJONG KIM SANG, E. F.; DE MEULDER, F. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. [S.l.: s.n.], 2003. p. 142–147.

TRNKA, K. et al. Topic modeling in fringe word prediction for aac. In: ACM. *Proceedings of the 11th international conference on Intelligent user interfaces*. [S.l.: s.n.], 2006. p. 276–278.

WEISCHEDEL, R.; BRUNSTEIN, A. Bbn pronoun coreference and entity type corpus. *Linguistic Data Consortium, Philadelphia*, v. 112, 2005.

ZAÏANE, O. R.; STRILETS, A. Finding similar queries to satisfy searches based on query traces. In: *Proceedings of the Workshops on Advances in Object-Oriented Information Systems*. London, UK, UK: Springer-Verlag, 2002. (OOIS '02), p. 207–216.

ZHANG, Y.; YIN, C.; YUAN, F. An application of improved pagerank in focused crawler. In: *Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on*. [S.l.: s.n.], 2007. v. 2, p. 331–335.