Ficha catalográfica automática - SDC/BRO

L732a Lima, Pedro Câmara Cardoso Robuste de
ANÁLISE DE COMENTÁRIOS DE CIDADÃOS EM REDES SOCIAIS PARA DESCOBERTA DE CONHECIMENTO EM CIDADES / Pedro Câmara Cardoso Robuste de Lima; Flavia Bernardini, orientadora. Rio das Ostras, 2017.
44 f.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação)-Universidade Federal Fluminense, Instituto de Ciência e Tecnologia, Rio das Ostras, 2017.

1. Cidades Inteligentes. 2. Redes Sociais. 3. Processamento de Linguagem Natural. 4. Produção intelectual. I. Título II. Bernardini,Flavia, orientadora. III. Universidade Federal Fluminense. Instituto de Ciência e Tecnologia. Departamento de Computação.

CDD -

Bibliotecária responsável: Maria Margareth Vieira Pacheco Rodrigues - CRB7/5874

UNIVERSIDADE FEDERAL FLUMINENSE
CAMPUS DE RIO DAS OSTRAS
INSTITUTO DE CIÊNCIA E TECNOLOGIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

PEDRO CÂMARA C. R. DE LIMA

ANÁLISE DE COMENTÁRIOS DE CIDADÃOS EM REDES SOCIAIS PARA DESCOBERTA DE CONHECIMENTO EM CIDADES

RIO DAS OSTRAS, RJ
2017

PEDRO CÂMARA C. R. DE LIMA

ANÁLISE DE COMENTÁRIOS DE CIDADÃOS EM REDES SOCIAIS PARA DESCOBERTA DE CONHECIMENTO EM CIDADES

Monografia apresentada ao Curso de Bacharelado em Ciências da Computação da do Instituto de Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel.

Orientadora:
Prof.ª Flavia Bernardini

Rio das Ostras, RJ
2017
PEDRO CÂMARA C. R. DE LIMA

ANÁLISE DE COMENTÁRIOS DE CIDADÃOS EM REDES SOCIAIS PARA DESCOBERTA DE CONHECIMENTO EM CIDADES

Monografia apresentada ao Curso de Bacharelado em Ciências da Computação da do Instituto de Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel.

Aprovado em 21 de dezembro de 2017.

BANCA EXAMINADORA

Prof.ª Flávia Bernadini - UFF
Orientadora

Prof. Carlos Bazilio Martins – UFF

Prof.ª Leila Weitzel - UFF

Rio das Ostras, RJ
2017

Aos meus pais

AGRADECIMENTO

Aos meus pais e meu irmão, pelo todo apoio e suporte nesses árduos anos de faculdade.

À minha namorada Laísa Cazelli, pelo apoio.

Aos amigos do grupo ‘Mulekadinha’ que ajudaram nesses anos de estudos.

À professora Flavia Bernadini pela orientação.

RESUMO

As redes sociais têm se tornado um importante canal de troca de informação e comunicação entre cidadãos. Como consequência várias opiniões e fatos são relatadas em forma de comentários. Nos últimos anos com o surgimento de cidades inteligentes, a participação popular nas diretrizes das cidades tem ganhado destaque. Este trabalho propõe construir um sistema que seja capaz de relacionar dados de cidades com comentários. Assim criar uma espécie de sistema *crowdsourcing*, onde os problemas e assuntos relatados pelos cidadãos acerca de um local seja exibida em um mapa. Este trabalho apresenta um estudo de Processamento de Linguagem Natural, mineração de textos em redes sociais, além dos princípios de cidades inteligentes.

Palavras-chave:

Redes sociais, cidades inteligentes, *crowdsourcing*, dados abertos, Processamento de Linguagem Natural.

ABSTRACT

Social networks have become an important channel for communication and exchanging information among citizens. As consequence, several opinions and facts are reported in form of comments. In the last years with the emergence of smart cities, popular participation in the directives of cities has gained prominence. This term paper proposes to build a system that is able to relate city data with comments. Thus, create a kind of crowdsourcing system, where problems and issues reported by citizens about a location are displayed on the map. This term paper presents a study of Natural Language Processing, text mining in social networks and the principle of smart cities.

Keywords:

Social network, smart cities, crowdsourcing, open data, Natural Language Processing.

LISTA DE FIGURAS

Sumário

Capítulo 1

INTRODUÇÃO

1.1 O Problema

Um dos pilares de cidades inteligentes está relacionado ao encontro da participação popular. Em (TERÁN, KASKINA e MEIER, 2016), é proposto um modelo de maturidade para Cidades Cognitivas fortemente embasado em um modelo de maturidade para governos eletrônicos, no qual se espera a participação popular nos processos decisórios governamentais. Já em (DAMERI e ROSENTHAL-SABROUX, 2014) é discutida a questão de construção de valor público por parte dos cidadãos que só pode ser adquirido pela participação popular. As redes sociais oferecem uma seara de informações fornecidas pelos cidadãos em páginas dos governos locais. Assim, muitas pessoas fazem comentários quanto à sua opinião de servidos oferecidos. Essa fonte de informação ainda é pouco explorada para entendimento da qualidade dos serviços prestados.

1.2 Motivação

Serviu como motivação deste trabalho diversas postagens de páginas de Facebook reclamando dos serviços prestados pelas cidades. A Figura 1 mostra um post com o seguinte título: “Iluminação pública precária em ruas do bairro Cidade Praiana. 24/07/17”. Nos comentários observa-se que muitos cidadãos usam deste post para também fazer reclamações sobre ruas de seus bairros. Assim, foi pensado em um sistema que fosse capaz de identificar as ruas dos bairros, marcar essas ruas nos mapas e listar os temas mais mencionados nas postagens.

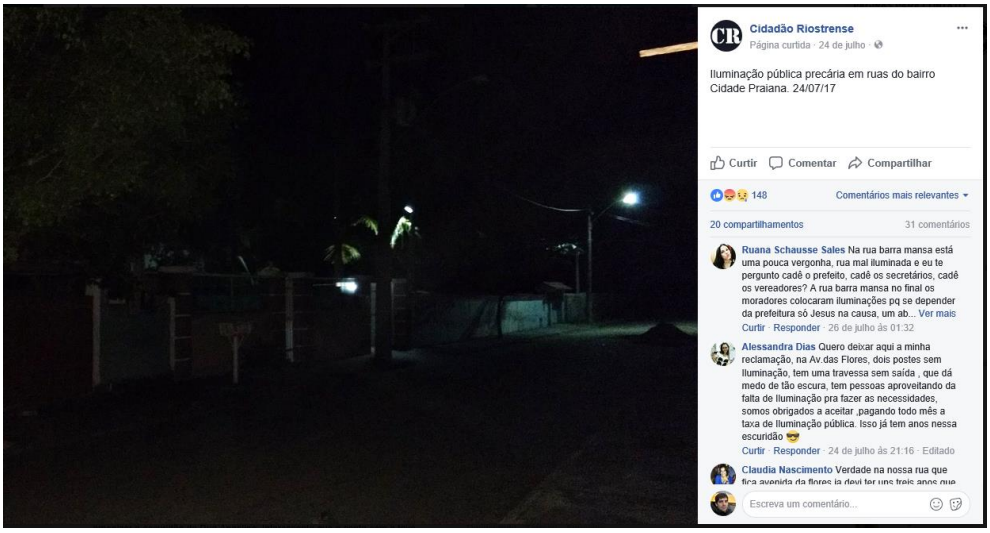

Figura 1 Postagem de uma página sobre problema de iluminação pública

1.3 Objetivo

O objetivo geral deste trabalho é utilizar técnicas de mineração de textos e visualização de dados para descoberta da opinião de cidadãos quanto a problemas apontados em comentários de páginas de prefeituras em redes sociais. São objetivos específicos deste trabalho:

- Realizar um levantamento do referencial teórico sobre cidades inteligentes, *crowdsourcing*, mineração de dados e de textos e processamento de linguagem natural.
- Realizar um estudo sobre a página de um município para verificar a viabilidade do uso de técnicas de mineração e visualização de dados.
- Construir um protótipo na web para visualizar as informações extraídas.

Neste trabalho foi desenvolvido um sistema web para visualizar em um mapa os comentários relativos aos logradouros de cidades assim como os problemas frequentes.

1.4 Metodologia

Para atingir os objetivos deste trabalho foi decidido explorar a visualização dos comentários em um mapa. A premissa dessa solução é trazer informação visual quanto a quantidade de comentários por local de uma cidade. Como no brasil a rede social mais utilizada é o Facebook (WE ARE SOCIAL SG, 2016) foi decidido processar uma página de um município no Facebook para ilustrar o funcionamento da solução proposta. Fazendo análise dos comentários são identificados os bairros e/ou logradouros citados a fim de

identificar a sua localização, bem como quais temas são mais processados frequentemente. A base dos Correios é utilizada para identificar quais logradouros ou bairros são citados. Além disso, o CEP do logradouro é usado para extrair a localização geográfica a que se refere o comentário.

Um cenário de uso da solução é a interação do usuário em alguma página de forma indireta e anônima. Por exemplo, um usuário comenta "A rua Rio de Janeiro no Extensão do Bosque está um breu, pagamos taxa de iluminação pública alta. Absurdo!", em alguma página referente a sua cidade. O sistema será capaz de extrair estes comentários e localizar, se existente, o local em que o problema é citado e o tema. Assim, pode-se indexar os logradouros com maiores carências e quais são os problemas mais citados nas cidades. Para isso foram usadas técnicas de Processamento de Linguagem Natural (PLN), no caso deste trabalho, comentários em redes sociais são processados afim de aumentar o conhecimento. Foi utilizada a biblioteca NLTK (*Natural Language ToolKit*) da linguagem *Python* (NLTK, 2017). Esta biblioteca fornece interface fáceis de usar para mais de 50 corpora e recursos léxicos, juntamente com um conjunto de bibliotecas de processamento de texto. O objetivo do NLTK é facilitar o desenvolvimento de aplicações de Processamento de Linguagem Natural. As principais funcionalidades do NLTK serão explicadas no capítulo 2 deste trabalho.

1.5 Organização do Trabalho

Este trabalho está organizado em quatro capítulos, além da introdução. No Capítulo 2 são apresentadas teorias de Cidades Inteligentes, *crowdsourcing,* mineração de textos em redes sociais e Processamento de Linguagem Natural e é feito uma revisão da literatura de artigos que estão relacionado a este trabalho.

No Capítulo 3 é mostrado a arquitetura do sistema. Seguindo um modelo de processos, as etapas do sistema são detalhadas. No Capítulo 4 é explicado o desenvolvimento do sistema e alguns resultados. No Capítulo 5, é apontado o que pode melhorar em trabalhos futuros e a conclusão.

Capítulo 2

REFERENCIAL TEÓRICO

2.1 Cidades Inteligentes

Segundo (QUIRINO, SANTOS, *et al.*, 2016) *Smart City* ou Cidade Inteligente está relacionado à construção de cidades em que se prioriza o uso eficiente dos recursos existentes ao invés do investimento e esforço na sua ampliação ou modificação. Inteligente refere-se a processos informatizados, lidando com um gigantesco volume de dados, isto é, *Big Data*, redes em nuvens e comunicação autônoma entre diversos objetos (IoT, do inglês *Internet of Things*). Este processamento inteligente servirá como referência e norteará as decisões dos governos ou empresas (LEMOS, 2013).

Para que uma cidade possa integrar serviços informatizados, é importante seguir políticas de disponibilização de dados abertos e interoperabilidade para que sistemas possam utilizar as informações. Algumas cidades do brasil já disponibilizam dados abertos em formas de relatórios para que cidadãos possam acompanhar as atuações do governo (CAFEZEIRO, VITERBO, *et al.*, 2017). Mas muitas vezes diferentes setores de um governo implementam políticas de acesso distintas e os dados não obedecem a um padrão definido.

2.2 *Crowdsourcing*

Crowdsourcing é um modelo de resolução de problemas online por meio da contribuição de um grande número de pessoas, chamado de sistema/serviço *crowdsourcing* aqueles que seguem esse modelo, ferramentas como o *Wikipédia*, *Waze* e *YouTube* são exemplos deste (QUIRINO, SANTOS, *et al.*, 2016). A ideia é atrativa pois como funciona online qualquer pessoa com acesso ao serviço pode contribuir, qualquer que seja a sua distância, escolaridade ou renda. Como as soluções são do produto fator humano demandam um elevado grau de cognição.

De forma geral as entidades envolvidas no modelo são (QUIRINO, SANTOS, *et al.*, 2016):

1. Beneficiado do problema: Quem propõe o problema a ser resolvido;
2. Plataforma *crowdsourcing*: Responsável por gerenciar/organizar o problema e as soluções geradas;

3. *Crowd:* Pessoas que contribuem para o serviço.

O tipo de colaboração pode ser explícito, quando o usuário gera uma solução diretamente, implícita quando o colaborador fornece uma solução indiretamente ou mista, quando utiliza as duas formas (QUIRINO, SANTOS, *et al.*, 2016).

As redes sociais hoje em dia têm sido usadas por grande parte da população para reclamar de algum serviço não prestado corretamente pelo governo, estas reclamações servem, mesmo que indiretamente para contribuir com o desenvolvimento das cidades.

2.3 Mineração de Dados em Redes Sociais

Segundo (HAN e KAMBER, 2011) Mineração de dados é o processo de descobrir informações úteis em dados de larga escala e também descoberta de conhecimento a partir de dados, este processo consiste em pré-processamento de dados, mineração de dados e pós-processamento. Mineração de dados é também é parte integrante de muitos campos relacionados, como estatística, aprendizado de máquina, reconhecimento de padrões, etc.

Diariamente vários conteúdos são gerados por usuários, essa tendência tende a crescer exponencialmente no futuro. Consequentemente, é importante para produtores, consumidores e provedores de serviços determinar a utilidade de dados gerados por usuários. De acordo com (GUNDECHA e LIU, 2012) o crescimento das redes sociais é conduzido por estes desafios: (1). Como o usuário pode ser ouvido? (2). Qual fonte de informação deve usar um usuário? (3). Como a experiência do usuário pode ser melhorada?

Dados gerados em redes sociais são diferentes dos convencionais, eles são vastos, aleatórios, distribuídos, desestruturados e dinâmicos.

2.4 Processamento de Linguagem Natural (PLN)

Com base em pesquisa (PEREIRA, 2013), a PLN é classificada em principalmente três aspectos da língua natural, são elas, som (fonologia), estrutura (morfologia e sintaxe) e significado (semântica e pragmática). A sintaxe define a estrutura, com base na forma como as palavras se relacionam em uma frase. Na PLN a classificação das palavras numa frase é chamada de *Part-of-Speeching (POS) Tagging.* O POS é usado para pré-processo de classificação semântica de um texto. No caso deste trabalho foi construindo e treinado um

POS tagging para identificar as localidades mencionadas nos comentários e posteriormente analisar os temas mais frequentes nos comentários.

Para isso foi usado a biblioteca NLTK (NLTK, 2017) que se define como plataforma líder para construir sistemas que usam linguagem humana e textual. As principais funcionalidades são:

- *Tokenizer* de sentenças: Criar uma lista de sentenças contidas em um texto.
- *Tokenizer* de palavras: Criar uma lista de palavras (*tokens*) contidas numa sentença.
- *Part-of-Speeching Tag (POS Tag)*: Classificação da palavra em uma frase.
- Reconhecimento de Entidades Nomeadas: Identificar entidades (pessoas, empresas, datas, etc.) no texto.

Além dessas funcionalidades o NLTK possui uns corpora[1] de diferentes linguagens com sentenças e palavras já classificadas sintaticamente afim de treinar e construir um sistema classificador.

2.4.1 Tokenizer

Para fazer a aplicação dos taggers é necessário quebrar o texto em sentenças e as sentenças em palavras, esse processo é chamado de *tokenization*. Existem duas abordagens, para linguagens delimitados por espaço e para linguagens não segmentadas. Na maioria das linguagens é utilizada a abordagem delimitadas por espaço, onde as palavras são separadas por espaço. Para linguagens de símbolos, como chinês, é necessária informação léxica e morfológica adicional. Entretanto, mesmo para linguagens delimitadas por espaço existem algumas exceções, por exemplo na língua portuguesa as palavras “Av.”, “Sr.” e “Sra.” são usadas para descrever palavras curtas e não devem ser separadas pelo ponto. A biblioteca NLTK possui módulos que fazem a *tokenization* de sentenças e palavras na língua portuguesa.

[1] Corpora: Conjunto de informações sobre um assunto.

2.4.2 Part-of-Speech (POS) Tagging e Corpus

Nesta etapa de classificação das palavras quanto a sua sintaxe numa frase (*Part-of-Speech*) é necessário a construção de um classificador. Por isso é imprescindível o uso de um corpus[2], pois levaria muito tempo criar um a partir do zero. Na língua portuguesa existem dois corpora que foram analisados e classificados sintaticamente, são eles o Floresta (FLORESTA SINTÁ(C)TICA , 2010) e o Mac-Morpho (FONSECA e G. ROSA, 2013).

Foi usado o corpus brasileiro de textos na língua portuguesa formado por 1 milhão de palavras de artigos publicados no *Jornal Folha de São Paulo, 1994* classificados em POS tag, o Mac-Mopho (ALUÍSIO, PELIZZONI, *et al.*, 2003), lançado em 2003 e revisado posteriormente (FONSECA e G. ROSA, 2013) (FONSECA, G. ROSA e ALUÍSIO, 2015). Utilizando uma metodologia descrita em (YUMUSAK, DOGDU e KODAZ, 2014) foi usado diferentes *taggers* disponíveis no NLTK para avaliar o corpus Mac-Morpho.

Podemos ver as sentenças classificadas e quantidade através do seguinte comando mostrado na Figura 2.

```
from nltk.corpus import mac_morpho
sents = mac_morpho.tagged_sents()
print(sents)
print(len(sents))
 [[('Jersei', 'N'), ('atinge', 'V'), ('média', 'N'), ('de', 'PREP'), ('Cr$', 'CUR'), ('1,4', 'NUM'), ('milhão',
'N'), ('em', 'PREP|+'), ('a', 'ART'), ('venda', 'N'), ('de', 'PREP|+'), ('a', 'ART'), ('Pinhal', 'NPROP'),
('em', 'PREP'), ('São', 'NPROP'), ('Paulo', 'NPROP')], [('Programe', 'V'), ('sua', 'PROADJ'),
('viagem', 'N'), ('a', 'PREP|+'), ('a', 'ART'), ('Exposição', 'NPROP'), ('Nacional', 'NPROP'), ('do',
'NPROP'), ('Zebu', 'NPROP'), (',', ','), ('que', 'PRO-KS-REL'), ('começa', 'V'), ('dia', 'N'), ('25',
'N|AP')], ...]

51397
```

Figura 2. Sentenças e a quantidade classificadas do MacMorpho

Percebe-se que a primeira sentença das 51397 classificadas foi: “Jersei atinge média de Cr$ 1,4 milhão em a venda de a Pinhal em São Paulo”. Como temos versões classificadas de um texto podemos a partir dela classificar outros textos, isto é feito usando *taggers* do NLTK, cada *tagger* foi avaliado e são descritos abaixo.

---

[2] Corpus: Conjunto de documentos sobre um tema.

Aplicação do POS Taggers

Para avaliar cada um dos *taggers* foi separado dois tipos de dados em treinamento e de testes. Nos dados de treinamento foram usados 80% do corpus Mac-Morpho e os 20% restantes foi usado para testar o *tagger* e avaliar sua porcentagem de acerto. A Figura 3 mostra a separação destes dados.

```
from nltk.corpus import mac_morpho
import nltk

sent_tokenizer = nltk.data.load('tokenizers/punkt/portuguese.pickle')

tsents = mac_morpho.tagged_sents()
tamCorpus = int(len(tsents)*0.8)
sentTreino = tsents[:tamCorpus]
sentTeste = tsents[tamCorpus:]
```

Figura 3. Dados de treinamento e testes do corpus

Default Tagger

O mais comum entre os *taggers*, antes de usar deve-se verificar a *tag* que mais ocorre no corpus, para a partir disso classificar as palavras. Basicamente o resultado vai ser a porcentagem da *tag* mais comum.

```
from nltk.corpus import mac_morpho
import nltk
tags = [t for s in mac_morpho.tagged_sents() for (p, t) in s]
frequencia = nltk.FreqDist(tags)
print(frequencia.most_common(5))
defaultTagger = nltk.DefaultTagger('N')
resultado = defaultTagger.evaluate(sents.tsents)
print(resultado*100.0)
# Precisão foi de 20.208%

[('N', 236462), ('ART', 151891), ('NPROP', 114318), ('PREP', 104364), ('V', 98056)]
```

Figura 4. Exemplo do Default Tagger

Com uma análise prévia a *tag* 'N' (Nome) é a mais comum, por isso foi usada para classificação. A Figura 4 mostra um exemplo de utilização do *tagger* onde indica que a precisão foi de 20%.

Regexp Tagger

Usa expressões regulares para classificação, é iniciado com um vetor de padrões e sua *tag*. Deve ser usado junto com o *default tagger*.

```
import nltk
import sents

patterns = [
    (r'(da|do|de|das|dos)$', 'PREP'), # Preposições
    (r'.*ndo$', 'V-GER') # Gerundios
]
defaultTagger = nltk.DefaultTagger('N')
regexTagger = nltk.RegexpTagger(patterns, backoff=defaultTagger)
resultado = regexTagger.evaluate(sents.sentTeste)
print(resultado*100.0)

# Precisão foi de 23.130%
```

Figura 5. Exemplo do Regexp Tagger

Como mostrado na Figura 5, o *RegexpParser* foi capaz de acertar 23% das ocorrências apenas passando os padrões de preposições (PREP) e gerúndios (V-GER) para o classificador. Como a complexidade da língua portuguesa é grande, não podemos facilmente definir padrões para a sintaxe de palavras.

Unigram Tagger

Atribui uma *tag* que mais aparece para uma palavra. Por exemplo, a palavra 'a' aparece na maioria das vezes como artigo, então este algoritmo a classifica como artigo para todas as ocorrências.

```
import nltk
import sents
tagger = nltk.UnigramTagger(sents.sentTreino)
resultado = tagger.evaluate(sents.sentTeste)
print(resultado*100.0)
# Precisão foi de 79.993%
```

Figura 6. Exemplo do Unigram Tagger

A Figura 6 exibe um exemplo para utilização deste *tagger*. A precisão é de aproximadamente 80%. Também é possível combinar várias *taggers* para que a precisão seja aumentada.

2.4.3 Expressões Regulares

Muitas das tarefas de Processamento de Linguagem Natural envolvem a busca de padrões em textos, essa tarefa pode ser complicada quando aplica-se operações básicas de *strings*. Por isso foi escolhido o uso de Expressões regulares. Expressões regulares são sequências de caracteres que definem um padrão de busca. Essas sequencias são usadas principalmente em compiladores. (BARBOSA, VIEIRA, *et al.*, 2017).

Na seção seguinte veremos que o uso de expressões regulares é usado também para encontrar entidades nos textos. Suponha que busca encontrar preços em alguma parte do texto, basicamente estaria procurando por (*R$)* seguindo de um ou mais números.

O uso de expressões regulares na linguagem *Python* é feito com a importação da biblioteca '*re*'. Por exemplo, podemos definir todas as palavras em verbos que tenham gerúndio com a expressão (*.*ndo*). A figura 4 mostra o uso para construir classificadores *POS-Tag.*

Os principais identificadores são mostrados a seguir.

- *d*: Qualquer número;
- *D*: Qualquer caractere, menos número;
- *s*: Espaço;
- *S*: Qualquer caractere, menos espaço;
- *w*: Qualquer letra;
- *W*: Qualquer caractere, menos letra;
- *. (ponto)*: Qualquer caractere, menos nova linha.

Os identificadores são usados juntos de modificadores:

- *{i,n}*: Espera encontrar de *i* a *n* repetições;
- *+*: Encontra 1 ou mais;
- *?*: Encontra 0 ou 1;
- *: Encontra 0 ou mais;
- /: Encontra ambos, ex: "*d*/*w*", encontra números e letras.

2.4.4 Extração de informação com *Chunking*

Muitas vezes procuramos encontrar alguma informação dentro de um texto. Em redes sociais podemos encontrar quantidades de textos e informação surpreendentes. No entanto a complexidade da linguagem pode tornar o acesso a informação muito mais difícil. O método para compreender o significado de textos é chamado de extração de informação, onde dados não estruturados são convertidos em dados estruturados. O estado da PLN ainda está longe de ser capaz de construir representações de propósito geral de significado sem restrições (BARBOSA, VIEIRA, *et al.*, 2017). A técnica para encontrar padrões é chamada de *chunking*, a qual segmenta sequências de multi-*tokens*. Para isso definimos uma gramática usando expressões regulares, consistindo em regras que indicam como uma frase deve ser fragmentada.

2.5 Revisão da Literatura

2.5.1 Introdução ao Processamento de Linguagem Natural usando Python

O artigo (BARBOSA, VIEIRA, *et al.*, 2017) tem como objetivo apresentar uma abordagem de Processamento de Linguagem Natural utilizando o a ferramenta NLTK (*Natural Language ToolKit)*, do P*ython*. Inclui exemplo práticos e aplicações básicas que permitem que os leitores pratiquem os conhecimentos em linguagem natural. Ao final da leitura deve-se compreender os elementos essenciais da PLN. Neste trabalho foi usado muitos dos exemplos mostrados do processamento de textos com o *NLTK*.

2.52 Estratégias *crowdsourcing* para aplicativos de cidades

Este artigo (QUIRINO, SANTOS, *et al.*, 2016) relaciona o uso de *crowdsourcing* com cidades inteligentes. Apresenta quatro estratégias de *crowdsourcing* para desenvolvimento de sistemas. A plataforma desenvolvida *Searchlight* tem como objetivo facilitar o desenvolvimento e prototipação de aplicações *crowdsourcing*. Um princípio apresentado no artigo e usado neste trabalho é a divisão de responsabilidades do sistema. Os autores propõem a divisão em três camadas básicas e independentes: coleta de dados, processamento e visualização.

Capítulo 3

ARQUITETURA DO SISTEMA

O sistema foi concebido para ser dividido logicamente em três módulos, de extração, processamento e visualização. O módulo de extração é responsável por obter os comentários do Facebook, logradouros da base dos Correios e obter as coordenadas dos logradouros. O modelo em BPMN (Notação de Modelagem de Processos de Négocio, do inglês, *Business Process Modeling Notation*) da Figura 7 abaixo mostra as etapas de extração e processamento. Os agrupamentos (*lanes)* 'Extração de Logradouro', 'Georeferenciamento' e 'Extração de Textos' são processos do módulo de extração. Os outros dois agrupamentos 'Identificação de Localidade' e 'Pré-processamento de texto' são tarefas do módulo de processamento de textos.

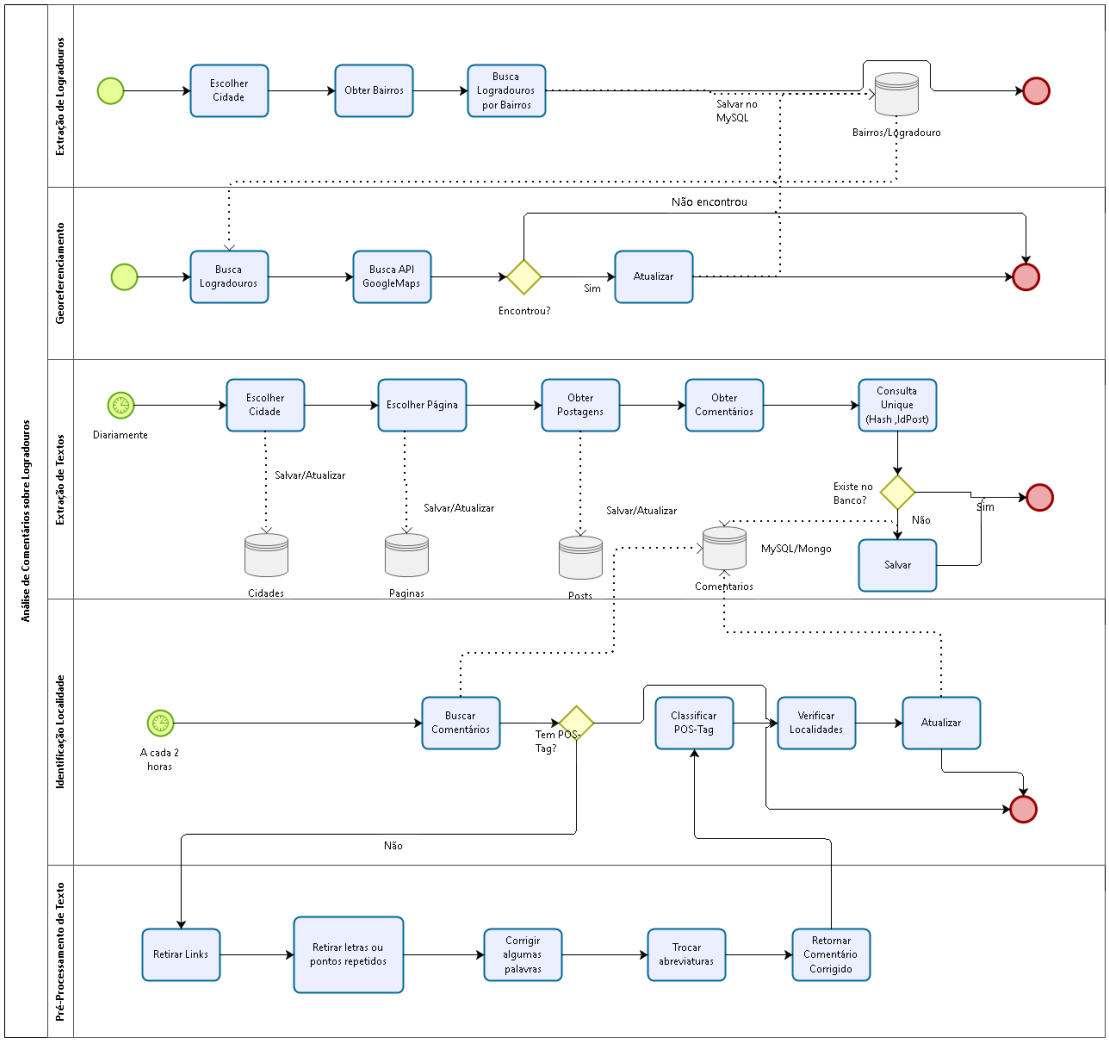

Figura 7. Modelo abstrato da arquitetura do Sistema

3.1 Detalhamento das atividades de extração

O início do desenvolvimento foi feito utilizando o banco de dados MySQL. Todo dado imutável como as cidades, bairros, logradouros, páginas, posts e comentários do Facebook foram salvos no MySQL. Na etapa de extração dos comentários foi necessária a utilização de um banco NoSQL, pois para classificar os comentários em *POS-tag*, analisar as localidades e os temas foi preciso salvar muitos dados aleatórios até encontrar um modelo definitivo. Isto no banco relacional seria um trabalho mais complicado, pois para cada novo campo inserido no comentário teria que alterar a estrutura da tabela, no Mongo isto não é necessário. A Figura 8 mostra o modelo relacional do banco Mysql. No Mongo alguns campos foram inseridos nessas entidades, esse aspecto será melhor detalhado na seção de visualização.

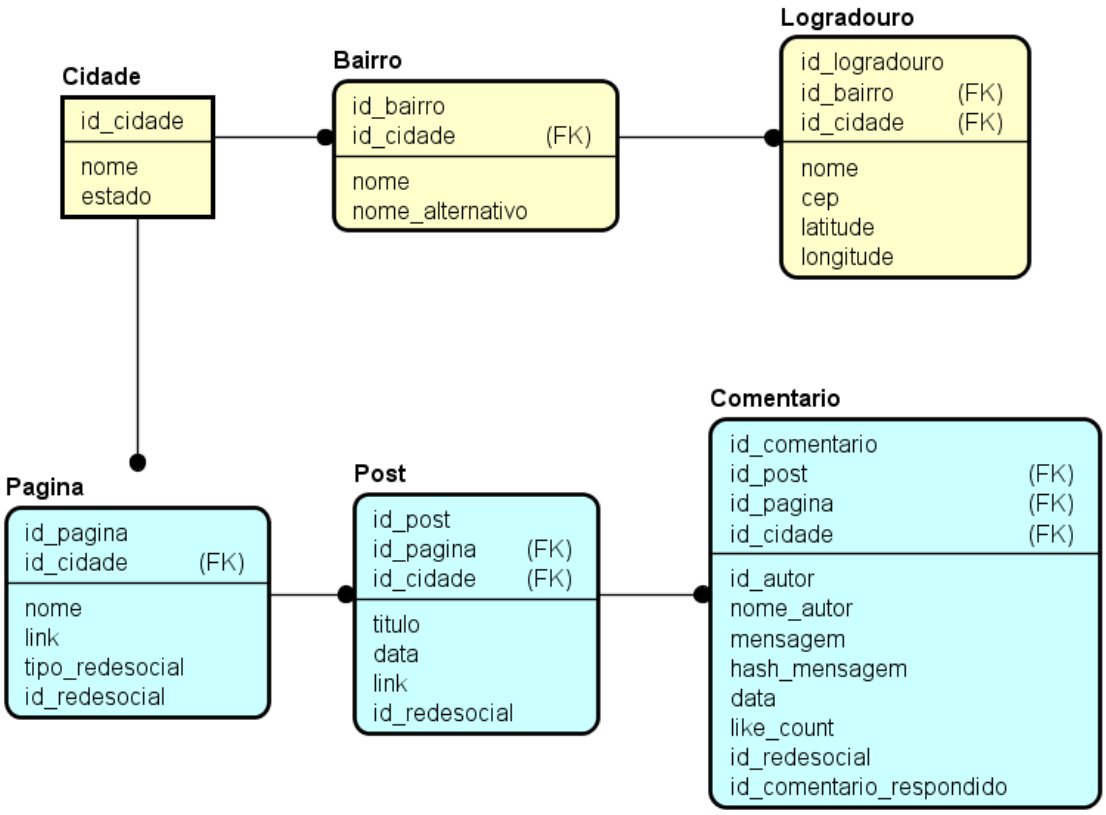

Figura 8. Modelo de entidade e relacionamento do Mysql

3.1.1 Bairros e Logradouros da Cidade

Para facilitar a inserção de cidades foi desenvolvido um sistema para terminal de comando na linguagem C# onde entra-se com o nome da Cidade e o Estado e o arquivo .CSV dos bairros, feito isso insere estes dados das entidades Cidade, Bairro e Logradouro no Mysql.

Se não existir um arquivo contendo os logradouros da cidade ou bairro, o processo de extração é feito através do site dos Correios pelo link:

http://www.buscacep.correios.com.br/sistemas/buscacep/resultadoBuscaLogBairro.cfm

Esta página permite entrar com o UF, Localidade e Bairro e visualizar todos os logradouros, juntamente com o CEP. A Figura 10 mostra o resultado da consulta da Figura 9.

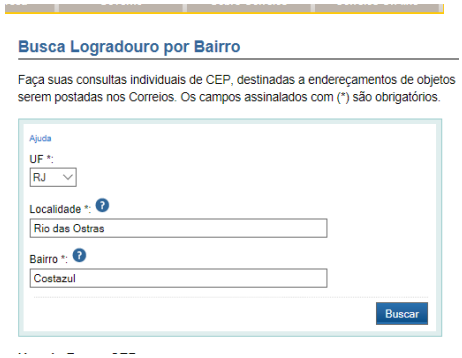

Figura 9. Consulta dos logradouros no site dos Correios

DADOS ENCONTRADOS COM SUCESSO.

[Anterior][Próxima][Nova Consulta] 1 a 50 de 71

Logradouro/Nome:	Bairro/Distrito:	Localidade/UF:	CEP:
Avenida Almirante Heleno Nunes - até 495/496	Costazul	Rio das Ostras/RJ	28895-114
Avenida Almirante Heleno Nunes - de 497/498 ao fim	Costazul	Rio das Ostras/RJ	28895-198
Avenida Costazul	Costazul	Rio das Ostras/RJ	28895-154
Avenida dos Bandeirantes	Costazul	Rio das Ostras/RJ	28895-314
Avenida Governador Roberto Silveira - até 599/600	Costazul	Rio das Ostras/RJ	28895-266
Avenida Governador Roberto Silveira - de 601/602 ao fim	Costazul	Rio das Ostras/RJ	28895-270
Avenida Irene dos Santos Ferreira	Costazul	Rio das Ostras/RJ	28895-178
Rodovia Amaral Peixoto	Costazul	Rio das Ostras/RJ	28895-310
Rua Alceu Lúcio Gomes	Costazul	Rio das Ostras/RJ	28895-062
Rua Alexandre Barbosa - até 573/574	Costazul	Rio das Ostras/RJ	28895-306
Rua Alexandre Barbosa - de 575/576 ao fim	Costazul	Rio das Ostras/RJ	28895-282
Rua Alexon Correa da Silva	Costazul	Rio das Ostras/RJ	28895-162
Rua Alfredo Pecegueiro do Amaral	Costazul	Rio das Ostras/RJ	28895-174
Rua Amaral	Costazul	Rio das Ostras/RJ	28895-070
Rua André Fillipe Ribeiro da Silva	Costazul	Rio das Ostras/RJ	28895-110

Figura 10. Resultado da consulta dos logradouros do site dos Correios

Para obter os dados dos correios e inserir no banco não seria eficiente fazer este processo manualmente. Por isso foi desenvolvido um robô com a função de percorrer o html da página e obter os dados. Utilizando o robô, foi extraído os dados da tabela e inserido no banco de dados para cada bairro. É importante ressaltar que o site bloqueia muitos acessos em sequência, e por isso cada consulta foi realizada com intervalo de 10 segundos.

3.1.2 Coordenadas de Logradouros

Utilizando o serviço Google Maps Geocoding API (GOOGLE, 2017) que oferece a geocodificação de endereços, isto é, obter as coordenadas. São feitas consultas para cada logradouro. O processo é feito fazendo uma requisição *HTTP* do endereço *url* da *API*, passando os parâmetros do logradouro. É retornado um *JSON* com as coordenadas.

Com os logradouros devidamente extraídos, é feita uma requisição passando o CEP e a cidade como parâmetro. A Figura 11 mostra um exemplo para obter a coordenada do CEP 28893-080.

Figura 11. Exemplo do Google Maps Geocoding API

Para esta tarefa foi feito um programa em Python no qual busca-se as coordenadas dos logradouros e atualiza no Mongo. As coordenadas não são atualizadas no Mysql.

3.1.3 Posts e Comentários sobre a Cidade

Para obter os dados do Facebook é necessário utilizar o serviço disponibilizado pelo mesmo, chamado de *Facebook Graph Api* (Graph API v2.10, 2017), baseada em *HTTP*. Cada requisição retorna um documento *JSON* e é obrigatória passar como parâmetro o *Access Token*, que é a chave de autenticação. Na página do *Facebook Graph* foi criado um aplicativo para obter um *Access Token*.

Foram utilizados os serviços *Page*, *Posts* e *Comments* do *Facebook Graph* chamados de *Root Nodes* (FACEBOOK, 2017), que serão explicados a seguir. Cada *Root Node* tem vários campos (chamados de *fields*) que contém os atributos que deseja obter.

Page

Utilizado para obter uma página específica do Facebook, com o access token é possível ver os campos públicos de uma página de restrição pública. Existem dezenas de campos, no desenvolvimento deste trabalho foi utilizado apenas três:

Nome	Descrição	Tipo
`id`	Atributo de identificação único de uma página	`string`
`name`	Nome da página	`string`

`link`	Endereço web da página	`string`

Tabela 1 Campos do node Page do Facebook Graph Api

A url de requisição fica assim:

GET /v2.10/{page-id}?fields=name,link,id&access_token={access-token} HTTP/1.1
Host: graph.facebook.com

Posts

As postagens referentes a uma página são feitas utilizando o node *posts*. Abaixo são explicados os campos utilizados:

Nome	**Descrição**	**Tipo**
`id`	Atributo de identificação único de um post. O Facebook utiliza junto com o id da página separando-os com '_', ex: {page-id}_{post-id}.	`string`
`created_time`	O momento em que o post foi publicado inicialmente.	`datetime`
`link`	Endereço web do post.	`string`
`message`	Mensagem da publicação.	`string`

Tabela 2 Campos do node Posts do Facebook Graph Api

A url de requisição fica assim:

GET /v2.10/{page-id}/posts?fields=message,id,created_time,link &access_token={access-token} HTTP/1.1
Host: graph.facebook.com

Comments

Da mesma forma que os posts, o node *comments* é utilizado para obter comentários referentes a um *post*.

Nome	**Descrição**	**Tipo**
`id`	Atributo de identificação único de um comentário no post.	`string`
`created_time`	O momento em que o comentário foi publicado inicialmente.	`datetime`
`from`	A pessoa que publicou o comentário.	`User`

like_count	Número de vezes em que o comentário foi 'curtido'	int32

Tabela 3 Campos do node Comments do Facebook Graph Api

A url de requisição fica assim:

GET /v2.10/{post-id}?fields=comments{id,created_time,from,message,like_count}
&access_token={access-token} HTTP/1.1
Host: graph.facebook.com

Parâmetros de Busca

Com os nodes de comentários e posts é possível modificar a busca com base no tempo e na quantidade de dados a ser retornado.

A busca com base no tempo é usada para obter dados especificando um intervalo de data usando o Unix timestamp. A lista abaixo mostra os parâmetros para a busca.

- `until`: um carimbo de data e hora Unix ou um valor de dados `strtotime` que aponta para o final do intervalo de dados baseados no tempo.
- `since`: um carimbo de data e hora Unix ou um valor de dados `strtotime` que aponta para o início do intervalo de dados baseados no tempo.

Para limitar a quantidade de resultados a ser retornado basta adicionar o comando `.limit({quantidade})`, onde quantidade é um valor inteiro do número de dados a serem retornados.

Capítulo 4

DESENVOLVIMENTO DO SISTEMA

Antes de iniciar o desenvolvimento do sistema é necessário definir em que plataforma este será projetado. A utilização da plataforma web se deve ao fato de que todos usuários com acesso à internet conseguem navegar em um *website*. Além disso, é necessário decidir em que servidor e sistema operacional o sistema vai rodar e quais tecnologias serão utilizadas. Neste trabalho, foi decidido que o sistema deveria executar em qualquer sistema operacional. Por isso, a tecnologia utilizada para o servidor foi o *.Net Core* (MICROSOFT, 2017), que é um *framework* de código aberto que é possível construir aplicações *web* (*ASP.NET*) e sistemas nativos para Windows, MacOs e Linux. Na aplicação *web* foi utilizado os frameworks *Angular* e o *Bootstrap* para implementação da interface gráfica.

Todo o código fonte deste trabalho está disponível no repositório online do *GitHub* pelo caminho *github.com/pedroccrl/tcc*.

4.1 Módulos

Conforme explicado no Capítulo 3, o sistema foi dividido logicamente em três módulos. Nesta seção será explicado o funcionamento das tarefas de extração e recuperação de dados, processamento e visualização.

4.1.1 Extração e Recuperação de Dados

O modelo de classe da Figura 12 estrutura e relação entre os objetos do sistema de extração e recuperação de dados. Veremos na seção 4.1.3 de visualização que este módulo também é usado para buscar os dados e exibir no site.

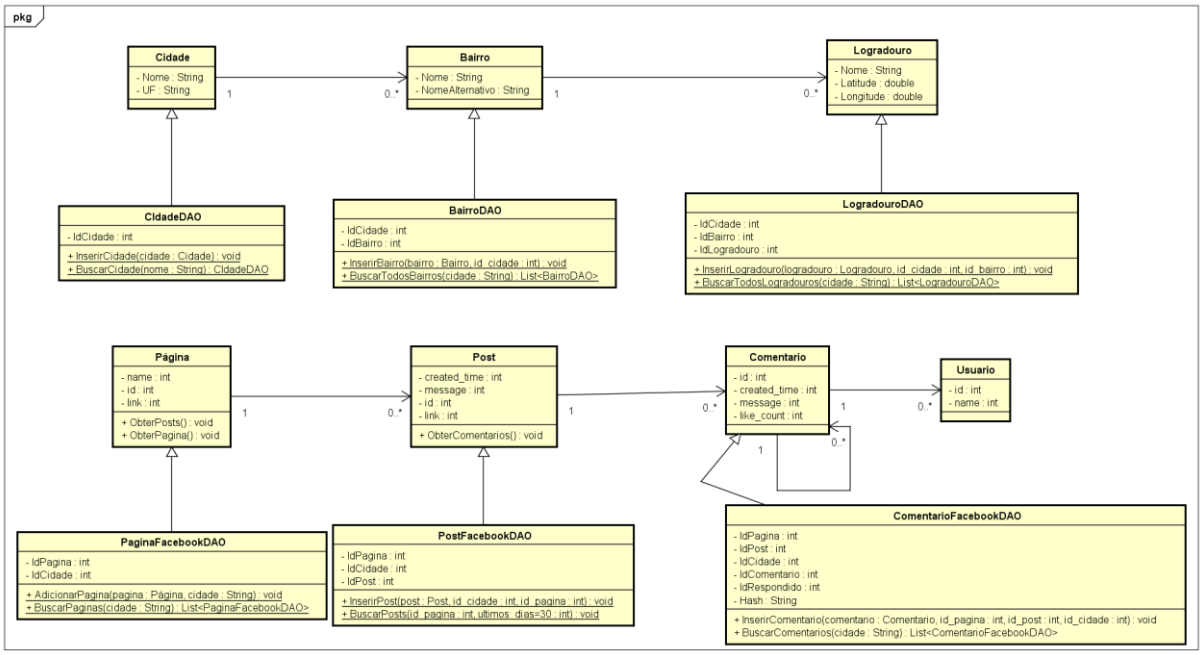

Figura 12 Diagrama UML do modelo de classes

Neste módulo são realizadas as funções manuais como é mostrado no caso de uso da Figura 13. É executado como um programa de console (*terminal*) em que as funções são executadas por um comando.

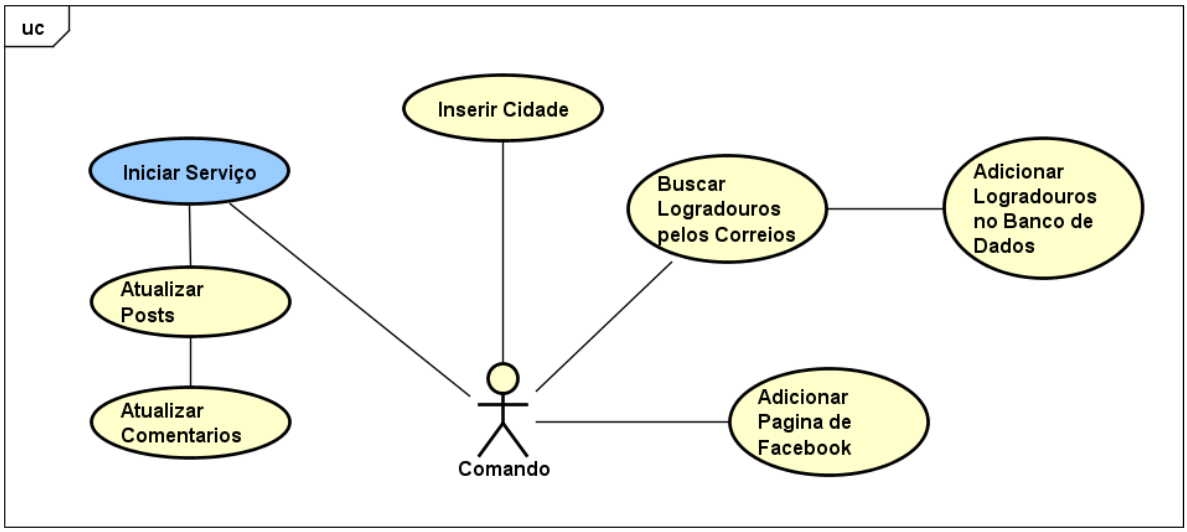

Figura 13 Casos de Uso do Módulo Comando

No início do desenvolvimento deste trabalho as ações de inserção de cidades, bairros, logradouros, páginas, posts e comentários eram realizadas de forma manual. Após a construção do modulo de Processamento de Dados foi feito um serviço para que

os comentários e posts fossem atualizados e processados automaticamente. A seguir será descrito os casos de uso da Figura 13 conforme modelo visto em (BEZERRA, 2007).

Inserir Cidade (CSU01)

Sumário: O gerenciador realiza a inserção dos dados de uma cidade.

Ator Primário: Administrador

Fluxo Principal

1. O gerenciador requisita a inserção de cidade.
2. Sistema requisita o nome e UF da cidade e um arquivo .CSV dos bairros caso tiver.
3. O gerenciador fornece os dados.
4. O sistema insere a cidade no banco de dados do *mysql* e informa ao ator.

Pós-condições: A cidade foi inserida no banco de dados.

Adicionar página do *Facebook* (CSU02)

Sumário: O gerenciador realiza a inserção de uma página do Facebook.

Ator Primário: Administrador

Fluxo Principal

1. O gerenciador requisita a inserção de página do Facebook.
2. O sistema requisita o nome da cidade referente a página.
3. O gerenciador informa o nome da cidade.
4. O sistema requisita o ID da página.
5. O gerenciador insere o id.
6. O sistema realiza uma busca pelo id para verificar se a página existe e pergunta ao ator para adicionar.
7. O gerenciador confirma a inserção da página.
8. O sistema insere a página no banco de dados *mysql.*

Fluxo de Exceção (6)

1. Sistema não encontra a página e retorna ao menu principal.

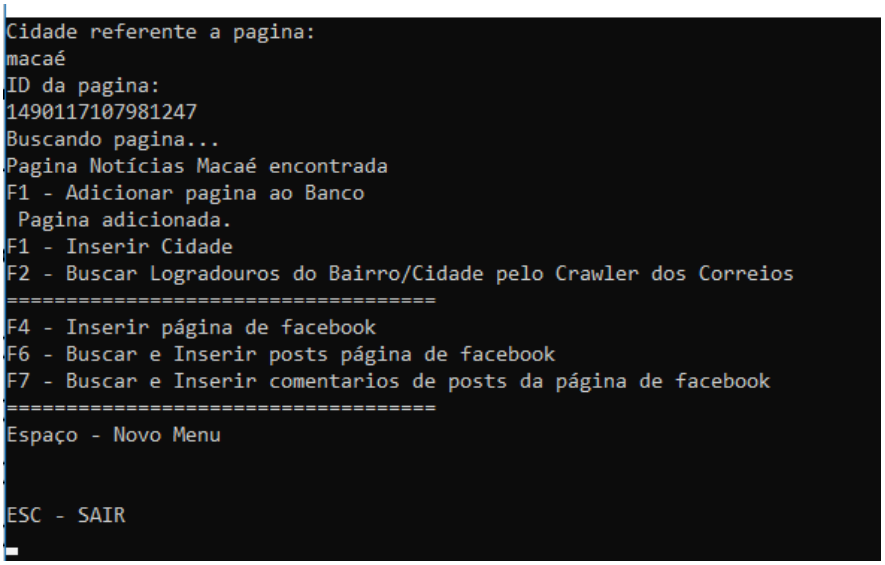

Figura 14 Tela do CSU02 para inserção de página do Facebook no sistema

Busca Logradouro pelos Correios (CSU03)

Sumário: O gerenciador requisita a busca de logradouros de uma cidade para o sistema adicionar ao banco de dados *mysql*.

Ator Primário: Administrador

Fluxo Principal

1. O gerenciador requisita a busca de logradouros.
2. O sistema requisita a cidade em que quer fazer a busca.
3. O gerenciador insere o nome da cidade.
4. O sistema mostra quantos bairros existem daquela cidade, começa a busca pelos logradouros, insere os logradouros encontrados no banco de dados *mysql,* informa pro ator e encerra o caso de uso.

Fluxo de Exceção (4)

1. O sistema encontra erro e retorna ao menu principal.

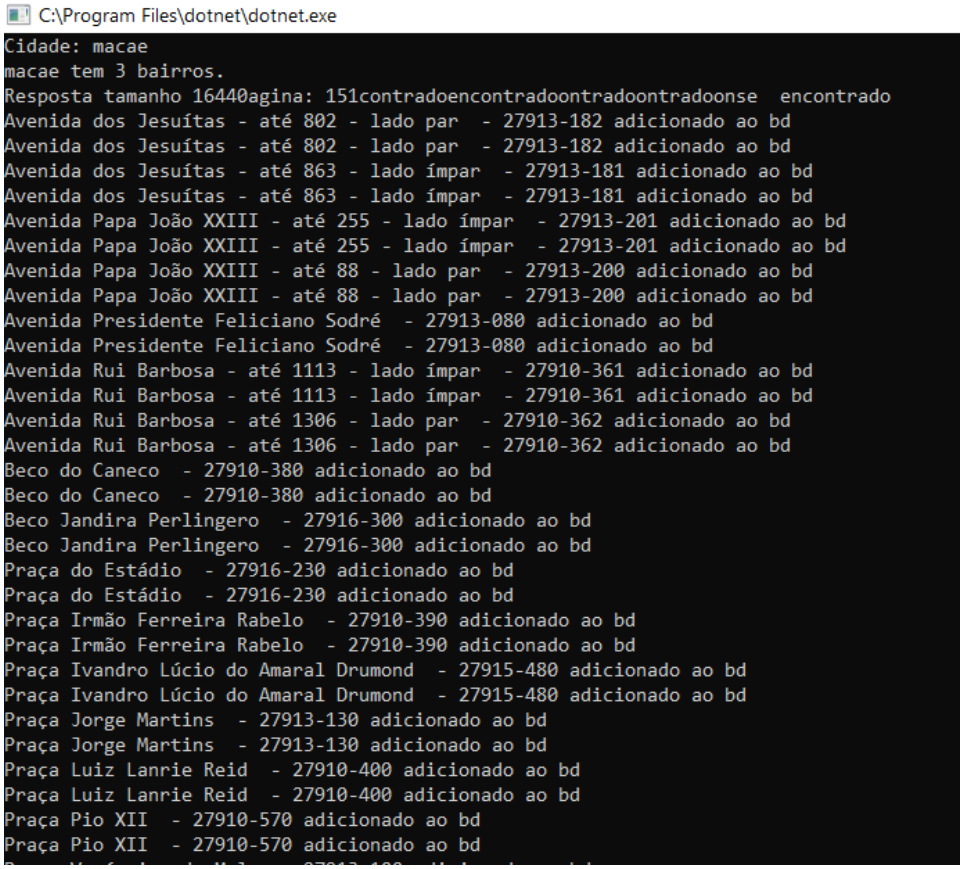

Figura 15 Tela do CSU03 para busca e inserção de logradouros

Iniciar Serviço (CSU04)

Sumário: Gerenciador inicia o serviço de atualização de posts e comentários de todas as páginas presentes no banco de dados mysql.

Ator Primário: Administrador

Ator Secundário: Sistema

Fluxo Principal

1. Gerenciador inicia o serviço
2. Sistema busca os posts e seus comentários das páginas dos últimos 30 dias.
3. Sistema insere novos posts e comentários no banco de dados *mysql*.
4. Sistema dorme por 24 horas e reinicia o serviço do passo 2.

Fluxo Alternativo

1. Gerenciador interrompe a execução do serviço.
2. Sistema não adiciona novos comentários e retorna ao menu principal..

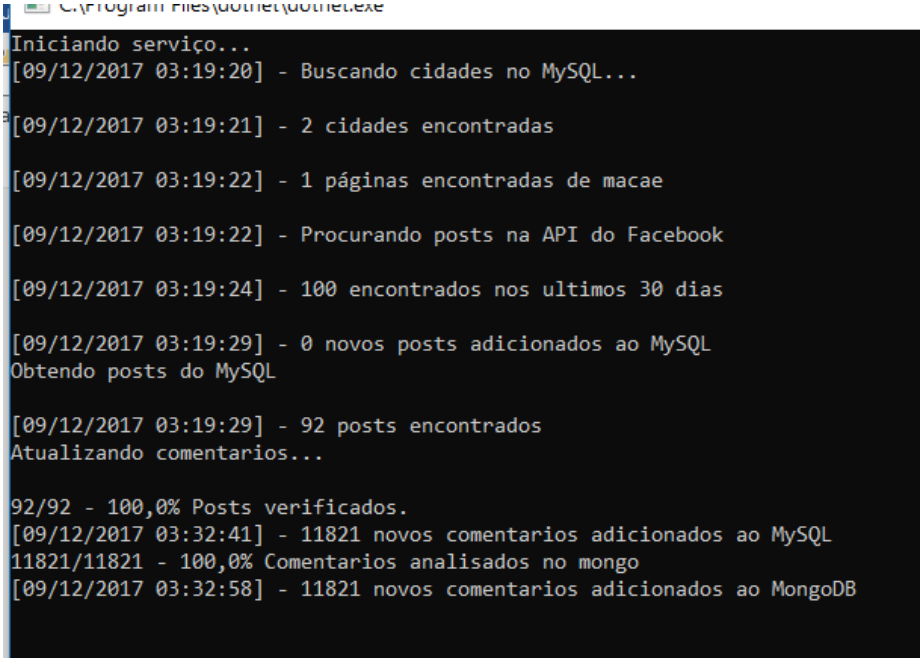

Figura 16 Tela do CSU04 para atualizar posts e comentários automaticamente.

Nesta seção foi explicado como funciona a inserção de dados no sistema, a seção 4.2.2 descreve como os comentários são processados.

4.1.2 Processamento dos comentários

O módulo de processamento dos comentários é responsável pelas funções de geocodificação dos logradouros, identificar logradouros citados em comentários e analisar temas mais frequente.

Pré-processamento de textos

Neste trabalho o uso de expressões regulares foi usado na tarefa de pré-processamento de texto. A Figura 6 mostra o código usado com a biblioteca '*re*' em *Python*.

```
import re

def CorrigirTexto(texto):
    p = re.compile(r'http://|www\S*', re.IGNORECASE) # retirar links
    texto = p.sub('', texto)
    p = re.compile(r'(\w)\1{2,}', re.IGNORECASE) # retirar mais que 2
letras repetidas
    texto = p.sub(r'\1', texto)
    p = re.compile(r'([!.?])\1{1,}', re.IGNORECASE) # retirar mais que 1
ponto repetido e adicionar um espaço após
    texto = p.sub(r'\1 ', texto)
    p = re.compile(r' ([!?.])') # retirar um espaço antes de pontuação
    texto = p.sub(r'\1', texto)
    p = re.compile(r'(\w)(ao)\s', re.IGNORECASE) # colocar ~ numa palavra
terminada em ao
    texto = p.sub(r'\1ão ', texto)
    texto = re.sub(r'[A-Z]{2,}|[a-z][A-Z]+', to_lower, texto) # substitui
mais uma palavra maiscula para minuscula

    for (abr, s_abr) in abreviacao:
        p = re.compile(abr, re.IGNORECASE)
        texto = p.sub(s_abr, texto)
    return texto
```

Figura 17 Uso de Expressões Regulares no pré-processamento de texto

A Figura 16 mostra a função '*CorrigirTexto*' que tem as funcionalidades:

- Retirar links;
- Retirar mais que duas letras repetidas;
- Retirar pontuação repetida;
- Retirar espaço antes de pontuação;
- Colocar '~' em palavras terminas com 'ao';
- Substitui palavras inteiras em maiúsculo para minúsculo;
- Substituir abreviações de uma lista definida, ex: 'vc' é substituído por 'você'.

Identificar Logradouros Citados

Inicialmente foi pensado em fazer a identificação de logradouros utilizando *POS-Tag*. Mas a complexidade dos comentários fez com que optasse por uma solução mais simples. A identificação é feita por operação básica de *string*, primeiro os logradouros são definidos quanto a seus tipos no banco de dados do *mongodb*. A Figura 18 mostra o campo "Tipo" do logradouro. Isto é feito para que quando encontrar um comentário que tenha escrito "rua", realizar apenas a busca desses tipos de logradouros.

Após identificar se algum tipo de localidade (rua, praça, rodovia, etc.) é encontrada no comentário, é realizado um *loop* para verificar a existência do nome do logradouro no comentário. O logradouro é inserido no comentário no campo “Logradouros” conforme mostra a Figura 18.


```
{
    "_id" : "1005467779555835_1005479609554652",
    "created_time" : "24/07/2017 19:12:51",
    "message" : "Travessa sao pedro cidade beira mar na mais
completa escuridão  uma vergonha",
    "like_count" : 0,
            ...
    "TemLogradouro" : true,
    "Logradouros" : [
        {
            "_id" : NumberLong(946),
            "Nome" : "Travessa São Pedro ",
            "Cep" : "28890-206",
            "Latitude" : -22.5442509,
            "Longitude" : -41.9783192,
            "IdBairro" : NumberLong(33),
            "IdCidade" : NumberLong(2),
            "Bairro" : {
                "Nome" : "CIDADE BEIRA MAR",
                "NomeAlternativo" : "",
                "Cidade" : "rio das ostras"
            },
            "Tipo" : "Travessa"
        }
    ]
}
```


Figura 18 Comentário com logradouro citado

Temas mais frequentes

Neste trabalho foi utilizado como dados textuais os comentários realizados por cidadãos em páginas do Facebook. Estes comentários algumas vezes são escritos com erros de português, abreviações, repetição de letras para dar ênfase (ex: “A rua está muuuuito escura”), etc. Isto torna ainda mais complicada a tarefa de extração das informações. Por isso, neste trabalho foi feito apenas uma análise dos temas mais frequentes. O objetivo é buscar as maiores necessidades e opiniões dos cidadãos, como iluminação pública, saneamento básico, corrupção entre outros.

```
chunkGram1 = """Sujeito: {<N.*>+<PREP.*>*<KS>*<ADJ>*<N.*>+}"""
chunkGram2 = """Sujeito: {<NPROP*>+}"""
chunkGram3 = """Sujeito: {<N.*>+<ADJ>+}"""
```

Figura 19 Gramática utilizada no *chunking*

A Figura 19 mostra os padrões utilizadas neste trabalho:

- *chunkGram1*: Busca encontrar nomes com preposições, como 'Rio das Ostras'.
- *chunkGram2*: Busca encontrar nomes próprios, como bairros, ruas, pessoas.
- *chunkGram3*: Busca encontrar um substantivo seguido de adjetivo, como 'iluminação pública'.

Os temas frequentes podem mostrar uma carência de determinado assunto. Com o *chunking* explicado na seção 2.4.4 Extração de informação com *Chunking*, foram extraídos os nomes citados e utilizando a função de frequência de distribuição do *NLTK*, chamada de *nltk.FreqDist,* listados os cem nomes mais frequentes*.* Os nomes foram salvos numa coelção particular no banco de dados do MongoDB.

4.1.3 Visualização

O modelo da Figura 20 mostra o diagrama de pacotes baseado em um modelo visto em (KATOCH, 2011) e adaptado para *UML*. Este diagrama tem as propriedades:

- *Client App*: Aplicação executada no navegador do usuário.
- *View:* Página *HTML*. Os métodos podem indicar um evento (*click* de botão, *link, etc.*).
- *Component:* Componente do Angular para a comunicação com o servidor e popular a *View*. Os métodos indicam uma requisição *HTTP*.
- *Path:* Caminho para o *Controller,* por exemplo o caminho: http://*servidor.com*/api/Comentarios pode ser utilizado para manipular os comentários.
- *Controller:* Onde o servidor responde pelas requisições do *Component*. Neste modelo apenas responde por *HTTP GET*.

Para compreender melhor o funcionamento dessas características do modelo de pacotes é preciso entender como funcionam os *frameworks* AngularJS e *ASP Web Api*.

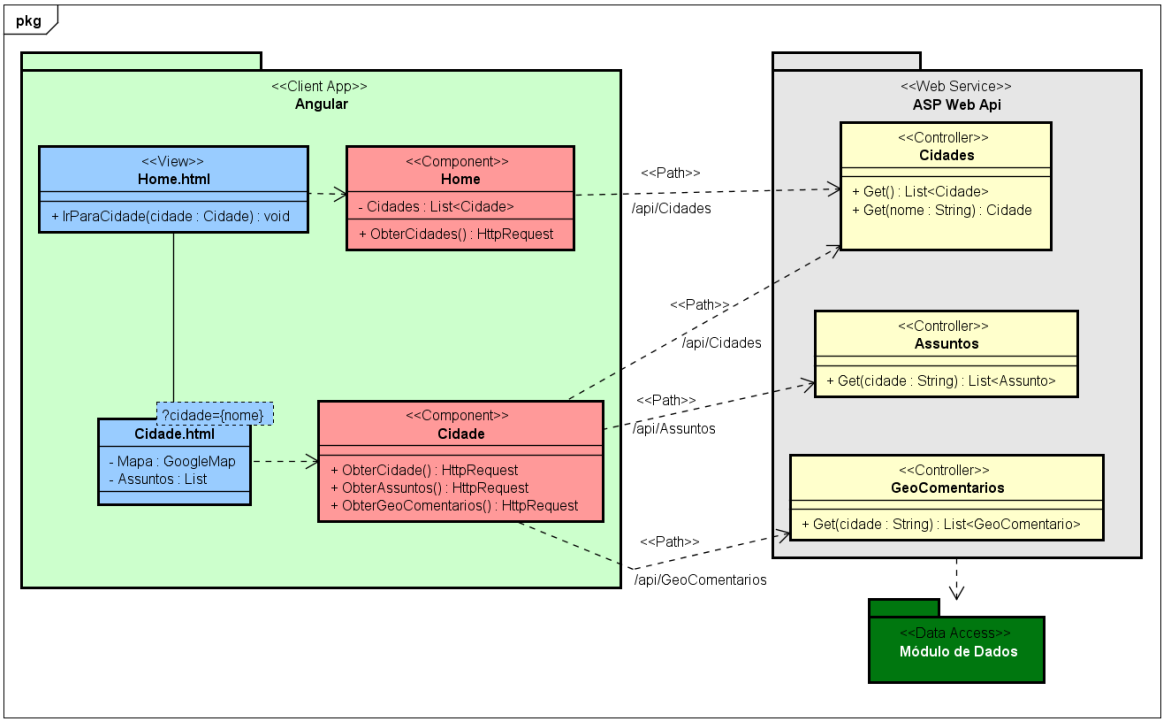

Figura 20 Modelo de Pacote do módulo de visualização

O *ASP Web Api* é um *framework* para criação de serviços *REST.* O *REST* usa protocolo *HTTP*, e seu acesso é feito por um endereço na web. Esse serviço abstrai os detalhes de implementação e por usar um protocolo já conhecido é possível fazer a maioria das aplicações se comunicarem (FIELDING, 2000). Neste trabalho foi utilizado este *framework* para criação de um *web service*. Os dados que foram extraídos e processados neste trabalho são acessíveis na *web* por meio dos controladores (chamados de *Controller* no *ASP)*. No site foram usados os seguintes dados: comentários com logradouros geocodificados, assuntos frequentes e as cidades. Portanto três controladores foram criados. A Figura 20 mostra esses controladores definidos pelo tipo *<<Controller>>*. É possível notar que este pacote chamado de *WebService* se relaciona com o módulo de dados mostrado na Figura 12.

Para construir o site foi necessário decidir como seria feita a requisição de dados do servidor. A escolha do Angular se deve principalmente por duas características: facilitar a requisição HTTP e o preenchimento do *HTML* das páginas com os dados obtidos chamados de *data binding*. Esse *framework* tem outras características importantes, mas vamos nos preocupar apenas como estas duas indicadas anteriormente. O Angular é baseado em componentes, onde cada um desses componentes é formado por um arquivo *HTML*, *CSS* e *TypeScript*. No *HTML* é feito a construção da página *web* associado ao estilo

da página feito no *CSS*. No *TypeScript* são realizadas as operações de *data binding* e requisição *HTTP* para aquela página. O *data binding* funciona como uma ligação bidirecional de dados (do inglês, *Two-Way data binding*), esta característica abstrai a complexidade para exibir os dados na página.

As requisições são feitas da seguinte forma: o componente do angular se comunica com o *WebService* para receber os dados. Estes dados vêm no tipo *JSON* e são convertidos para objetos no código *TypeScript*. A Figura 21 mostra as classes (no *TypeScript* são definidas como *interface*) dos dados que são passados dos *Controllers* para o componente do Angular.

```
interface Cidade {
    cidade: string;
    bairros: number;
    logradouros: number;
    paginas: number;
    posts: number;
    comentarios: number;
}

interface Bairro {
    nomeAlternativo: string;
    nome: string;
    cidade: string;
}

interface GeoComentario {
    comentario: string;
    id_comentario: number;
    logradouro: Logradouro[];
}

interface Assunto {
    temas: Mencao[];
    qualidades: Mencao[];
}

interface Mencao {
    key: string;
    value: number;
}

interface Logradouro {
    idBairro: number;
    nome: string;
    cep: string;
    longitude: number;
    _id: number;
    latitude: number;
    tipo: string;
    bairro: Bairro;
    idCidade: number;
}
```

Figura 21 Interfaces do Angular no módulo de visualização

Voltando ao módulo dos pacotes da Figura 19 é possível notar dois componentes realizados com o Angular. O *Home* é a página inicial do trabalho. Através da comunicação com o servidor é buscado a lista de cidades cadastradas no sistema com o número de comentários. Clicando na cidade o usuário é direcionado para a página do componente 'Cidade'. Na página da cidade, além da Cidade são obtidos os GeoComentários e Assuntos.

A tela de Rio das Ostras é mostrada na Figura 22. A coluna da esquerda exibe os dados da Cidade (nome e quantidade de bairros, logradouros, páginas, *posts* e comentários). A do meio exibe os comentários georeferenciados no mapa. E a coluna da direita exibe os assuntos mais frequentes seguido do número de menções.

Figura 22 Tela do website do sistema para cidade de Rio das Ostras

4.2 Estudo de caso

Até a data de entrega deste trabalho os números encontrados para a cidade de Rio das Ostras foram:

- Bairros: **66** bairros de um arquivo *CSV* enviado por uma funcionária da Subsecretaria de Tecnologia da Informação de Rio das Ostras.
- Logradouros: **1505** obtidos pelo robô para percorrer o site dos Correios.
- Página: Só foi feita para a página do Facebook "**Cidadão Riostrense**".
- Postagens: **881** posts entre janeiro de 2016 e novembro de 2017.
- Comentários: **32157** entre janeiro de 2016 e novembro de 2017.

Com o processamento desses dados foi identificado a menção de localidades[3] em 2054 comentários e os logradouros georeferenciados foram encontrados em 184 comentários. A partir da lista dos cem assuntos mais frequentes, foi feita uma filtragem do que seria mais importante. Esta funcionalidade de listar os assuntos mais frequentes teve resultados interessantes. Foi possível perceber que o principal tema comentado nos últimos meses na cidade de Rio das Ostras foi 'iluminação pública'. Na lista a seguir, são listados os principais assuntos (filtrados) encontrados pelo sistema:

1. Iluminação pública

[3] Localidades podem ser: 'rua', 'praça', 'alameda', 'rodovia', 'beco', 'estrada', 'largo', 'ramal', 'travessa'.

2. Taxa de iluminação pública
3. Carlos Augusto (Prefeito de Rio das Ostras)
4. Poder público
5. Queima de fogos
6. Saneamento básico
7. Ministério público
8. Guarda municipal
9. Falta de educação
10. Deus conforte
11. Dinheiro público
12. Coleta de lixo
13. Descaso total

Capítulo 5

CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresenta um referencial teórico sobre Cidades Inteligentes, *Crowdsourcing,* mineração de dados de redes sociais e Processamento de Linguagem Natural. Foram detalhadas as tarefas de PLN para pré-processamento de textos, como construir um classificador *POS-Tag*, o uso de expressões regulares e extração de informações de textos com *chunking*. Este trabalho traz como contribuição a apresentação de um sistema construído que permite verificar que tipos de assuntos estão sendo falados nas redes sociais sobre locais de cidades bem como a geolocalização de diversos deles. Os comentários são então visualizados em um mapa e os assuntos mais frequentes numa lista. Essas opiniões dos cidadãos encontravam-se distribuídas em milhares de comentários e posts no Facebook. Com esse sistema podemos indexar e listar o que podem ser os maiores problemas de uma cidade. Assim, além de funcionar como servidor *web, o* sistema desenvolvido é capaz de obter e servir como provedor de dados.

A maior dificuldade no desenvolvimento deste trabalho foi na tarefa de Processamento de Linguagem Natural, pois a complexidade de extrair informações em textos não estruturados é grande. O corpus Mac-Morpho (FONSECA e G. ROSA, 2013) foi classificado para um texto dos anos 90 extraído de um jornal. Muitas palavras de linguagem informal são usadas na internet, portanto o classificador pode não estar completo neste sentido. O ponto a melhorar neste trabalho é o classificador *POS-Tag* que em algumas frases classifica as palavras erroneamente. Por exemplo, em algumas situações o classificador pode classificar uma preposição como um nome substantivo. Esta falta de precisão do classificador prejudicou a listagem. Ainda, a lista de assuntos mais frequentes mostra também assuntos pouco relevantes, como ‘lima’, ‘anos’, ‘da cidade’, ‘meses’ entre outros. Para um trabalho futuro seria importante realizar uma filtragem de assuntos que podem não ser importantes.

Quanto ao sistema, há pontos de melhoria a serem acrescentados futuramente. Na página das cidades poderiam ser exibidos os comentários por intervalo de datas. Ainda, a análise de sentimentos também é interessante nesse problema, pois o sentimento das pessoas nos comentários pode indicar a polaridade de sentimento de comentários em um

bairro, por exemplo. Assim poderíamos ter um maior conhecimento sobre que local tem pior ou melhor sentimento.

REFERÊNCIAS BIBLIOGRÁFICAS

ALUÍSIO, S. et al. **An Account of the Challenge of Tagging a Reference Corpus for Brazilian Portuguese**. Proceedings of the 6th International Conference on Computational Processing of the Por tuguese Language. [S.l.]: [s.n.]. 2003.

BARBOSA, J. L. N. et al. Introdução ao Processamento de Linguagem Natural usando Python. In: PIAUÍ, I. E. R. D. I. D. **Livro Anais - Artigos e Minicursos**. 1. ed. [S.l.]: ERIPI, v. 1, 2017. Cap. 5, p. 336-360.

BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML**. 3ª. ed. [S.l.]: Elsevier Brasil, 2007.

BRABHAM, D. C. Crowdsourcing as a model for problem solving: An introduction and cases. **Convergence**, v. 14, p. 75-90, 2008.

CAFEZEIRO, I. et al. Strengthening of the Sociotechnical Approach in Information Systems Research. In: BOSCARIOLI , C.; ARAUJO, R. M.; MACIEL, R. S. P. **Grand Research Challenges in IS in Brazil**. [S.l.]: Brazilian Computer Society, 2017. Cap. 11.

COCCHIA, A. Smart and Digital City: A Systematic Literature Review. In: DAMERI, R. P.; ROSENTHAL-SABROUX, C. **Smart City:** How to Create Public and Economic Value with High Technology in Urban Space. 1. ed. [S.l.]: Springer, 2014. Cap. 2, p. 13-43.

DAMERI, R. P.; ROSENTHAL-SABROUX, C. Smart City and Value Creation. In: DAMERI, R. P.; ROSENTHAL-SABROUX, C. **Smart City:** How to Create Public and Economic Value with High Technology in Urban Space. [S.l.]: Springer, 2014. Cap. 1.

FACEBOOK. Graph API v2.10, 18 Julho 2017. Disponivel em: <https://developers.facebook.com/docs/graph-api>. Acesso em: 27 set. 2017.

FLORESTA SINTÁ(C)TICA. Projecto Floresta Sintá(c)tica, 2 Agosto 2010. Disponivel em: <http://www.linguateca.pt/Floresta/>. Acesso em: 14 nov. 2017.

FONSECA, E. R.; G. ROSA, J. L. Mac-Morpho Revisited: Towards Robust Part-of-Speech Tagging, 2013.

FONSECA, E. R.; G. ROSA, J. L.; ALUÍSIO, S. M. Evaluating word embeddings and a revised corpus for part-of-speech tagging in Portuguese. **Journal of the Brazilian Computer Society**, 2015.

GOOGLE. Google Maps Geocoding API, 6 Abril 2017. Disponivel em: <https://developers.google.com/maps/documentation/geocoding/start>. Acesso em: 15 nov. 2017.

GUNDECHA, P.; LIU, H. Mining Social Media: A Brief Introduction. **INFORMS Tutorials in Operations Research**, 2012.

HAN, J.; KAMBER, M. **Data Mining:** Concepts and Techniques. 4ª. ed. [S.l.]: Elsevier, v. 1, 2011.

KATOCH, S. Design and implement RESTful web services with Rational Software Architect. **IBM DeveloperWorks**, 20 Dezembro 2011. 1-16.

LEMOS, A. Cidades Inteligentes. **Fundação Getulio Vargas**, v. 12, n. 2, p. 46-49, 2013.

MICROSOFT..NET Core e software livre..**NET Core**, 2017. Disponivel em: <https://msdn.microsoft.com/pt-br/library/dn878908(v=vs.110).aspx>. Acesso em: 10 nov. 2017.

NLTK. Natural Language Toolkit. **NLTK Project**, 2017. Disponivel em: <http://www.nltk.org/>. Acesso em: 20 out. 2017.

PEREIRA, S. D. L. Processamento de Linguagem Natural, 2013.

QUIRINO, W. et al. **Estratégias crowdsourcing para aplicativos de cidades**. XII Brazilian Symposium on Information Systems. Florianópolis, SC: [s.n.]. 2016.

TERÁN, L.; KASKINA, A.; MEIER, A. Maturity Model for Cognitive Cities. In: PORTMANN, E.; FINGER, M. **Towards Cognitive Cities:** Advances in Cognitive Computing and its Application to the Governance of Large Urban Systems. [S.l.]: Springer, v. 63, 2016. Cap. 3.

WE ARE SOCIAL SG. Digital in 2016. **SlideShare**, 2016. Disponivel em: <https://www.slideshare.net/wearesocialsg/digital-in-2016/104-wearesocialsg_104JAN2016_TOP_ACTIVE_SOCIAL>. Acesso em: 04 dez. 2017.

YUMUSAK, ; DOGDU, ; KODAZ,. Tagging Accuracy Analysis on Part-of-Speech Taggers. **Journal of Computer and Communications**, p. 157-162, 2014.