UNIVERSIDADE FEDERAL DO RIO DE JANEIRO - UFRJ
Instituto de Matemática - IM
Departamento de Ciência da Computação - DCC
Tópicos Especiais em Sistemas Inteligentes

Gerando uma Base de Conhecimento de uma Pessoa Através de Notícias

Alunos:
Aluizio dos Santos de Lima Filho
Christian Cardozo

Orientador:
João Carlos Pereira da Silva

Co-oritadores:
Fabrício Firmino de Faria
Fábio Rangel

15 de Janeiro de 2017, Rio de Janeiro.

Sumário

Introdução **2**

1. Ferramentas utilizadas e Instalação **3**

1.1 Instalação 3

2. O Projeto **6**

2.1 Estrutura 6

2.2 Execução 7

3. Coleta das Notícias **8**

3.1 Pré-processamento 8

4. Processamento de Linguagem Natural **9**

4.1 Extração de entidades 9

4.2 Extração de relacionamentos 10

5. Avaliação **12**

6. Conclusão **13**

7. Referências **14**

Introdução

Neste trabalho construímos uma ferramenta para coletar notícias sobre uma pessoa. Embora o sistema seja generalizável a qualquer pessoa, nós coletamos apenas notícias sobre a ex-presidente Dilma Rousseff.

Para coletar as notícias, usamos uma biblioteca em python chamada Scrapy que auxilia na coleta de dados em páginas web. Depois de coletados os dados, é realizado um processamento de linguagem natural, onde é criada a base de conhecimento da pessoa em questão. Com essa base, podem ser feitos vários tipos de processamentos para descobrir novas informações sobre o indivíduo analisado.

1. Ferramentas utilizadas e Instalação

Algumas bibliotecas auxiliaram no desenvolvimento deste trabalho. São elas:

- Scrapy - biblioteca em python para a coleta de dados em páginas web
- Aelius - biblioteca em python para realizar a classificação gramatical em português das notícias.
- Nltk - para auxiliar o Aelius
- LX-tagger - também para auxiliar o Aelius

A instalação dos pacotes básicos pode ser feita simplesmente usando o instalador do python: o pip. O Aelius requer uma instalação mais cuidadosa. Estas bibliotecas são essenciais para o funcionamento do presente projeto. A utilização do Python 2.7 é requisito para que as bibliotecas funcionem corretamente.

1.1 Instalação

Para instalar os pacotes básicos desse projeto basta realizar o comando:

```
pip install -r requirements.txt
```

Já para Instalar o Aelius é necessário seguir os procedimentos abaixo:

1. Baixe a versão Aelius-February-25-2013 do Aelius através do seguinte link:
   - https://sourceforge.net/projects/aelius/files/libraries/
2. Crie uma pasta chamada “Applications” no diretório Home:
   - `mkdir ~/Applications`
3. Copie o Aelius para a pasta Applications.
4. Copie a pasta aelius_data para o diretório Home.
5. Crie uma pasta chamada “analises” no diretório Home:
   - `mkdir ~/analises`
6. Adicione o seguinte script no final do arquivo “.profile”, localizado no diretório Home:

```
# path to MXPOST jar file
CLASSPATH="$HOME/jmx/mxpost.jar"
export CLASSPATH
# path to MXPOST binary
PATH="${PATH}:$HOME/jmx"
export PATH
# path to HunPos binaries
PATH="${PATH}:$HOME/Applications/bin"
export PATH
# path to Aelius package
PYTHONPATH="${PYTHONPATH}:$HOME/Applications"
```

```
export PYTHONPATH
# path to Stanford Tagger jar file (eventually, update the version date or the
# folder name to the one of your installed version)
CLASSPATH="${CLASSPATH}:$HOME/stanford-postagger-2012-11-11/stanford-postagger.jar"
export CLASSPATH
# eventually, update the version date or the
# folder name to the one of your installed version
CLASSPATH="${CLASSPATH}:$HOME/apache-opennlp-1.5.2-incubating/lib/opennlp-tools.jar"
export CLASSPATH
PATH="${PATH}:$HOME/apache-opennlp-1.5.2-incubating/bin/"
export PATH
PYTHONPATH="${PYTHONPAtH}:$HOME/Applications"
export PYTHONPATH
```

7. Instale pacotes necessários para o Aelius:
    - `sudo apt-get install python-numpy python-yaml python-matplotlib`
8. Baixe a versão 2.0.1rc1 do Nltk através do link:
    - https://pypi.python.org/pypi/nltk/2.0.1rc1
9. Descompactar o arquivo baixado e dentro da pasta descompactada executar o seguinte comando para instalação:
    - `sudo python setup.py install`
10. Acesse o prompt do Python e execute os comandos:
    - `import nltk`
    - `nltk.download("punkt")`
    - `nltk.download("stopwords")`
11. Baixe o POSTagger através do seguinte link:
    - http://lxcenter.di.fc.ul.pt/tools/download/POSTagger.zip
12. Descompacte o POSTagger e mova-o para a pasta Applications que foi criada no diretório Home.
13. Dentro da pasta ~/HOME/aelius_data, crie um link simbólico usando o seguinte comando:
    - `ln -s ~/Applications/POSTagger/Tagger/MX-PoST/UTF-8_Model_Cintil_Written/ lxtagger`
14. Crie uma pasta “bin” dentro de Applications.
    - `mkdir ~/Applications/bin`
15. Coloque na pasta “bin” o Hunpos. Para baixá-lo, utilize o repositório deste projeto no GitHub através do link:

- Para sistemas 32 bits: https://github.com/NLP-TESI/PeopleNewsBuilder/tree/master/hunpos/bin-x64
- Para sistemas 64 bits: https://github.com/NLP-TESI/PeopleNewsBuilder/tree/master/hunpos/bin-x86

16. Modifique as permissões de execuções do Hunpos executando o comando:
    - `chmod +x Applications/bin/*`

2. O Projeto

2.1 Estrutura

Os arquivos python que contém o código:

- **Crawler.py** - contém o código que coleta as notícias nas páginas web sobre uma pessoa.
- **extractor.py** - contém o código que extrai o conhecimento das notícias sobre uma pessoas, ou seja, extrai entidades e relacionamentos.
- **knowledge.py** - contém o código que permite a manipulação do conhecimento coletado e também salva as entidades e as relações arquivo em csv
- **main.py** - contém o código para execução de toda a ferramenta que já está programado para coletar notícias da Dilma. Para coletar as notícias é necessário passar a palavra crawler como parâmetro na chamada desse arquivo.
- **NamedEntity.py** - contém o código da estrutura de uma entidade nomeada.
- **New.py** - contém o código da estrutura de uma notícia coletada pelo crawler.
- **preprocessing.py** - contém o código do pré-processamento realizado nas notícias.
- **relationships.py** - contém o código da estrutura de um conjunto de relações, criado para evitar relações repetidas.
- **TESIUtil.py** - contém códigos de funções que auxiliam na extração de entidades.

Outros arquivos:

- **aelius_tagset.pdf** - é um pdf de uma página web que contém as marcações gramaticais usadas pela biblioteca Aelius.
- **requirements.txt** - contém a lista de bibliotecas básicas usadas pelo projeto, esse arquivo é usado pelo pip para instalar todas as bibliotecas necessárias em um comando só.

Pastas:

- **hunpos -** contém arquivos compilados usados na instalação do Aelius.
- **database** - contém uma pasta que contém os arquivos das notícias coletados, essa pasta é gerada só depois que ocorrer a execução do código do projeto.
- **knowledge_base** - contém uma pasta que contém os arquivos csv's das entidades e das relações extraídas.

- **Resultados** - pasta que contém as planilhas utilizadas para o cálculo de avaliação na seção 5 do trabalho. As planilhas contém uma coluna booleana de classificação feita manualmente sobre a corretude da lista de entidades e relações.

2.2 Execução

Depois que toda a instalação for concluída, então será possível executar o projeto. A execução do projeto é simples, basta executar o seguinte comando:

```
python main.py crawler
```

O parâmetro crawler pode ser omitido se a coleta das notícias tiver sido realizada pelo menos uma vez. Esse comando irá gerar vários arquivos de notícias e dois csv's um contendo as entidades é outro com as relações.

3. Coleta das Notícias

Para realizar a coleta das notícias a biblioteca Scrapy foi fundamental, mas o início da coleta não é feito através do Scrapy. Primeiro, precisamos coletar todos os links de notícias sobre a pessoa e depois passamos esses links para o Scrapy poder trabalhar. Essa coleta de links é feita de uma forma automática, no entanto, o link inicial tem que ser inserido de forma manual.

O link inicial foi obtido no site G1 e a resposta à requisição feita para esse link é um JSON que contém uma lista dos links das notícias. Esse procedimento foi escolhido pois garante que serão obtidos links corretos. Outro método seria realizar as extrações dos links usando o Scrapy, mas esse não foi realizado.

Depois que todos os links são coletados, o scrapy começa a coletar as notícias extraídas do HTML em cada um deles. Quando o Scrapy termina a coleta, todas as notícias coletadas são salvas em arquivos. Os nomes desses arquivos são os identificadores das notícias, que ficam salvas no formato JSON.

3.1 Pré-processamento

Com os dados já coletados, agora só precisamos remover caracteres indesejados que estejam no texto. Mas esses dados em sua maioria estão limpos e as marcações de HTML já foram removidas com a ajuda do Scrapy. Devido a isso, só tivemos que remover os caracteres ‘[’ e ‘]’ que estavam causando classificações gramaticais incorretas.

4. Processamento de Linguagem Natural

Esse é o ponto em que começamos a extração de conhecimento da pessoa. Primeiro é realizado uma separação por sentença de cada texto, depois em cada uma dessas sentenças é realizada a separação por palavras (tokens). No final é feito a classificação gramatical de cada token. Esse resultado é passado para a extração de entidades e depois para a de relacionamentos.

4.1 Extração de entidades

Para cada notícia coletada, é realizado um processamento de extração de entidades. Como uma entidade pode aparecer diversas vezes e de formas diferentes (como com nome e o sobrenome), é mantido um dicionário de entidades global que é preenchido e alterado para cada notícia processada. As entidades nomeadas extraídas neste processo podem ser de categorias diversas devido ao contexto do trabalho, como nome de pessoa, lugar, região, organização, etc.

Primeiro, as entidades são analisadas no âmbito local, ou seja, para cada notícia. O texto de cada notícia é tokenizado por sentenças usando o Aelius. Depois, cada sentença da notícia é anotada usando o método anota_sentencas. Neste momento, o Aelius realiza a classificação gramatical das sentenças usando seu POSTagger. Afim de reduzir erros do POSTagger, um caso especial é classificado neste momento com a classe NPR, o token: “Lava Jato”.

Após as sentenças estarem classificadas, é realizado o processamento de extração de entidades locais. Apenas sentenças que contém a entidade principal do problema (“Dilma”) serão processadas para extração de entidades. Este processamento busca tokens classificados com a classe NPR. Caso os tokens estejam em sequência e classificados como NPR, como por exemplo “Polícia Federal”, eles são agregados para formar uma só entidade. As entidades identificadas neste processo sofrem uma alteração de classes de NPR para NE, recebem um id único e são salvas em um dicionário local temporário.

Por fim, é necessário que para cada dicionário de entidades locais identificadas verificar se ela já está no dicionário global. Esta verificação é feita afim de agregar entidades como “Dilma” e “Dilma Rousseff” como uma entidade só. Para fazer essa verificação, é utilizado um cálculo de similaridade de strings:

Similarity(str1, str2) = cont*0.4 + jaro_wikler * 0.6

Onde “jaro_winkler” é o algoritmo de Jaro Winkler para cálculo de similaridade entre strings, e “cont” é calculado com as seguintes regras:

- (str1 contido em str2 ou str2 contido em str1) e (primeira palavra de str1 igual a primeira palavra de str2):

    - cont = 1
- (str1 contido em str2 ou str2 contido em str1) e (primeira palavra de str1 não é igual a primeira palavra de str2):
    - cont = 0.5
- Caso contrário:
    - cont = 0

Cada entidade local será adicionada a uma entidade global já existente caso a similaridade seja maior que o valor de threshold 0.7. Este valor foi alcançado de forma empírica e pode variar de acordo com a origem dos dados e do problema. Caso haja mais de uma entidade candidata com o valor de threshold alcançado, é escolhida aquela com o maior valor de similaridade. Caso não haja entidades candidatas viáveis, é criada uma nova entrada no dicionário global com o id da entidade local em questão. O dicionário global contém o resultado da extração das entidades ao final do processamento de todas as notícias.

4.2 Extração de relacionamentos

Depois da extração de entidades, é feita a extração de relacionamentos entre essas. A extração é realizada analisando o texto que agora já está marcado com as classificações gramaticais. Assim a seguintes marcações são analisadas:

- “NE” - (*named entity*) essa é marcação de entidade nomeada que é colocada na extração de entidades. Ela nos permite identificar quais entidades farão parte do relacionamento.
- “VB” - (*verb*) com esta identificamos verbos e verificamos se esses verbos estão entre as entidades para montarmos um relacionamento, caso exista mais de um verbo entre as entidades esses verbos são unidos formando o relacionamento.
- “CONJ” e “,” - (*conjuction*) caso uma dessas marcações seja encontrada entre duas entidades então nenhum relacionamento é criado, pois essas marcações determinam o fim de uma oração em português, com isso sabemos que as entidades não podem formar um relacionamento.
- “WPRO”, "(" e ")" - estes foram adicionados apenas para remover relacionamentos sem sentido que eram formados.

O algoritmo funciona da seguinte maneira:

1. Busca por uma entidade.
2. Se for o fim da sentença ou uma conjunção for encontrada, retorna para 1.
3. Busca por um relacionamento.
4. Se for o fim da sentença ou uma conjunção for encontrada, retorna para 1.
5. Busca por outra entidade.
6. Forma um relacionamento.

Um caso especial foi adicionado como não sendo uma relação em que a palavra ‘ex’ aparece como a relação entre as entidades, isso ocorreu pois o Aelius estava classificando-a como um verbo. Além disso somente relações que contém a entidade principal, que no caso é a Dilma, é capturado.

5. Avaliação

Para avaliar o percentual de acerto da nossa ferramenta, realizamos a coleta de 1000 notícias sobre a presidente Dilma. No entanto o Scrapy só conseguiu coletar 937. Depois de quase 30 minutos em execução, o programa conseguiu extrair 1262 entidades e 1114 relações. Assim, para obter os resultados de corretude dos dados extraídos, foi feita uma classificação manual de cada entidade e de cada relação, onde verificamos se as entidades identificadas pelo programa eram de fato uma entidade e, no caso da relação, verificando se a relação fazia algum sentido.

Com isso conseguimos obter percentual de acerto de 76,36% em entidades e 56.33% em relações. Os resultados mostraram-se promissores, especialmente para as entidades. Ainda há possibilidades de melhorias, tanto na extração de relações como nas de entidades. As possibilidades podem incluir a construção de uma base de conhecimento não necessariamente sobre a ex-presidente Dilma.

6. Conclusão

A construção automática de uma base de dados de uma pessoa utilizando notícias mostrou ser uma tarefa bastante desafiadora. As dificuldades encontradas com o processamento de linguagem natural para a língua portuguesa assim como outros desafios de extração de informações mostram como a área pode ser explorada.

Os resultados obtidos no presente trabalho atingiram as expectativas iniciais. A construção de um Crawler para baixar notícias, a extração de entidades nomeadas e relações utilizando as técnicas de processamento de linguagem natural formam um projeto que pode ser utilizado para formar a base de conhecimentos de outra pessoa qualquer com notícias suficientes na web.

Além do projeto, a avaliação feita na seção 5 mostra que em grande maioria as entidades estão sendo classificadas de forma correta e que alguns processos de refinamento podem melhorar ainda mais os resultados.

As diversas técnicas e teorias de processamento de linguagem natural aplicadas no presente trabalho foram de extrema importância para alcançar os objetivos propostos, podendo assim contribuir com experimentos práticos para os conhecimentos da área.

Este trabalho, assim como seus códigos e arquivos binários podem ser baixados através do seu repositório no GitHub[1].

[1] https://github.com/NLP-TESI/PeopleNewsBuilder

7. Referências

Scrapy. Disponível em: <https://scrapy.org/>. Acesso em 15 de janeiro de 2017.
Aelius. Disponível em: < http://aelius.sourceforge.net/>. Acesso em 15 de janeiro de 2017.
Nltk. Disponível em: < http://www.nltk.org/>. Acesso em 15 de janeiro de 2017.
LX-Tagger. Disponível em: <http://lxcenter.di.fc.ul.pt/tools/pt/conteudo/LXTagger.html>. Acesso em 15 de janeiro de 2017.
G1, Disponível em: <http://g1.globo.com/>. Acesso em 15 de janeiro de 2017.