

Criação de Corpora para Línguas de Minorias com Recurso ao Web

Universidade do Minho

Projeto nº 9

Trabalho realizado por:

Leonel Gonçalves (A7230) — Tiago Loureiro (a71191)

Orientadores:

Joaquim Macedo — José João

June 26, 2019

Contents

1	Introdução	3
2	Análise e Especificação	4
2.1	Descrição informal do problema e requisitos	4
2.2	Levantamento de requisitos	5
3	Estrutura do Projeto	7
3.1	Menu	7
3.2	Makefile	9
3.3	config.py, utils.py e languages.json	10
3.4	Logs	12
3.5	Processador de textos	13
3.6	Pesquisas Google	16
3.7	Processamento dos links obtidos	18
3.7.1	Documentos Word	19
3.7.2	Documentos PDF	19
3.7.3	Documentos de texto simples	20
3.7.4	Páginas WEB	20
3.8	Identificação da Língua	21
4	Especificação técnica	23
5	Dificuldades	25
6	Próximos passos	26
7	Conclusão	27
8	Referências	28

List of Figures

2.1	Pipeline de execução.	5
3.1	Execução do menu.py”.	7
3.2	Execução do bootstrap.py com ficheiro para processar.	14
3.3	Execução do bootstrap.py sem ficheiro para processar.	15
3.4	Excerto do dicionário Kimbundu.	16
3.5	Execução do modulo crawler.py.	17
3.6	Exemplo do ficheiro queries.json”.	18
3.7	Extração de conteúdo Word”.	19
3.8	Extração de conteúdo PDF”.	20
3.9	Extração de conteúdo Html”.	21
3.10	Execução do modulo identifier.py”.	22
4.1	Estrutura completa do programa.	24

Chapter 1

Introdução

Este trabalho prático enquadra-se na unidade curricular de Projeto do curso de Licenciatura em Ciências da Computação da Universidade do Minho. O que se pretende é que os estudantes explorem um tema ou “problema” proposto pelo(s) orientador(es) e encontrem soluções adequadas.

O projeto que nos foi proposto (Projeto 9), tem como objetivo a criação de um programa, que facilite a procura de informação sobre línguas de minorias, utilizando recursos web.

Ao longo dos próximos *X* capítulos vamos expor o caminho que seguimos para chegarmos ao objetivo final, assim como todas as dificuldades e o que fizemos para as ultrapassar, sem descorar todos os ensinamentos que retiramos desta caminhada.

Palavras-chave: Corpora, Umbundo (Unbundu), Quimbundo (Kinbundu)

Chapter 2

Análise e Especificação

2.1 Descrição informal do problema e requisitos

Angola, ex colónia Portuguesa, tem como língua oficial o Português. No entanto, só 71,15% da população adotou essa língua[1]. Tal deve-se ao facto deve-se às variadas línguas angolanas, algumas delas com uma adoção muito reduzida.

Sendo que existem línguas com uma adoção de apenas 2% da população, é difícil encontrar documentos, de forma manual, para se conseguir construir uma Corpora.

Foi com o intuito de suprimir esta dificuldade que desenvolvemos, ao longo deste semestre, este projeto, sendo que nos focamos nas duas Línguas mais adotadas em Angola para além do Português, que são o Umbundo e o Quimbundo.

Por isso, é necessário o desenvolvimento de um programa, onde fosse possível retirar informação via web, como documentos, textos e urls onde fossem processados por um processo de verificação para avaliar se contém informação útil sobre uma dada língua. Foi utilizado a língua *Python3* e as blibliotecas *Google-Search-API*, *BeautifulSoup*, *Fulltext* e *Flake8*.

2.2 Levantamento de requisitos

De início, não tínhamos uma ideia definida da estruturação deste projeto. Sem nenhuma base no conhecimento destas linguagens, tivemos muita dificuldade de como iríamos avaliar um texto, e como seria possível classifica-lo com a sua língua.

Sendo assim, seguimos as sugestões dos nossos orientadores e pesquisamos de forma manual na web textos comuns em várias línguas e fáceis de encontrar, são eles, a declaração dos direitos humanos, documentos religiosos, dicionário medico, ...

De seguida, criamos um formato para os nossos ficheiros, de modo a que os textos sejam processados de forma correta, com a diferenciação de linguagens no caso de textos bilingues.

Através desses ficheiros, foi criado dicionários para cada língua, visto que tínhamos textos base para cada uma das linguagens.

O problema era que não tínhamos uma maneira exata sem ser por pesquisa manual, e com ajuda de entendidos na área para uma classificação correta dos textos.

Para resolver este problema, foi necessário pensar numa estratégia de uma pipeline de execução, e após uma reflexão de como queríamos estruturar o nosso projeto, com a ajuda dos nossos orientadores, foi criada a seguinte pipeline:

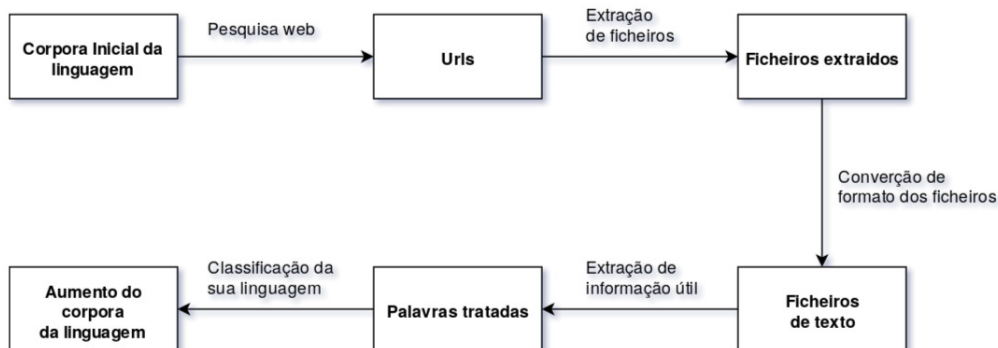


Figure 2.1: Pipeline de execução.

Desta forma, chegamos à conclusão de que não seria necessário formatar os textos, pois caso o utilizador utilizasse textos mal formatados como base, iria criar dicionários errados, pois poderiam entrar palavras nesses dicionário de outras línguas, o que daria resultados errados ao classificar se uma palavra pertence a esse mesmo. Simplesmente teríamos de ter documentos constituídos por palavras só de uma dada língua, para podermos ter uma base sólida para

explorar, de forma a conseguir desenvolver um automatismo para processar esses textos e recolher mais documentos úteis a estudo a partir das palavras extraídas no processamento.

Esta pipeline foi pensada da seguinte forma:

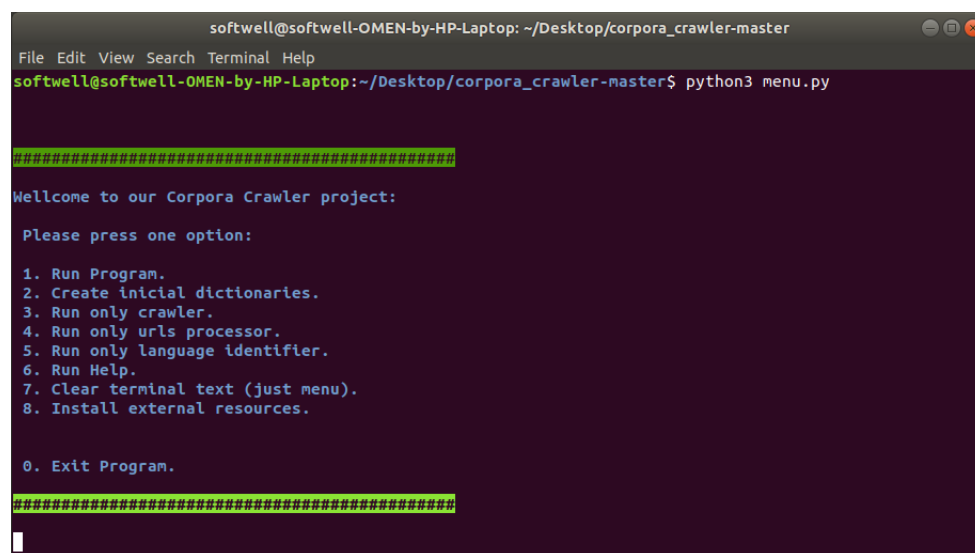
- Através de um corpora estruturado para cada língua, é criado um dicionário;
- Com esse dicionário, é feito uma pesquisa ao Google, com frases constituídas por palavras aleatórias desse dicionário;
- Com o resultado dessas pesquisas, obtido os *urls*;
- É feito uma extração da informação da página, ou extraído os seus documentos;
- Esses ficheiros são convertidos, todos para ficheiros texto, para serem mais fáceis de processar;
- É extraída a informação útil desses documentos, ou seja, retirado todos as palavras sendo feita uma limpeza as palavras, retirando pontuação, dígitos e caracteres especiais;
- É feita uma avaliação para parágrafo, como uma classificação da sua língua para possível aumento do corpora da sua língua, e verificar a riqueza do texto para uma dada língua .

Chapter 3

Estrutura do Projeto

3.1 Menu

Foi criado o modulo *menu.py*, pois pensamos que seria vantajoso para o utilizador uma interface mais amigável na sua utilização. Na seguinte imagem é possível observar a inicialização do menu.



```
softwarell@softwarell-OMEN-by-HP-Laptop: ~/Desktop/corpora_crawler-master
File Edit View Search Terminal Help
softwarell@softwarell-OMEN-by-HP-Laptop:~/Desktop/corpora_crawler-master$ python3 menu.py

#####

Wellcome to our Corpora Crawler project:

Please press one option:

1. Run Program.
2. Create inicial dictionaries.
3. Run only crawler.
4. Run only urls processor.
5. Run only language identifier.
6. Run Help.
7. Clear terminal text (just menu).
8. Install external resources.

0. Exit Program.

#####
```

Figure 3.1: Execução do menu.py”.

Temos as seguintes opções de execução:

- 1. - Executa todos os módulos consecutivamente, ou seja, execução completa do projeto.
- 2. - Executa só o *bootstrap.py*, responsável pela criação dos dicionários das linguagens.
- 3. - Executa só o *crawler.py*, responsável pela pesquisa na web.
- 4. - Executa só o *processor.py*, responsável pela extração de informação dos links das pesquisas.
- 5. - Executa só o *identifier.py*, responsável por classificar o texto e extração de dados.
- 6. - Faz *cat* do ficheiro *help.txt*. O ficheiro *help.txt* contém uma pequena explicação do funcionamento do projeto e o que as suas variantes realizam.
- 7. - Simplesmente limpa o terminal, deixando apenas o menu.
- 8. - Este programa precisa do *Python3* para a sua execução, mas foram utilizados bibliotecas adicionais. Por isso temos este comando para uma fácil instalação dessas bibliotecas.
- 0. - Para sair do programa.

3.2 Makefile

Foi criado uma *Makefile*, pois o utilizador pode não ser fã da utilização do menu, então também tem a possibilidade de executar todo o projeto assim.

Esta abordagem traz também a vantagem de facilitar a integração em mecanismos de sistema, como por exemplo *crons*. Na *Makefile*, estão disponíveis os seguintes comandos:

- `help`: Faz cat do ficheiro "help.txt".
- `install`: Instala as bibliotecas adicionais necessárias à execução do programa.
- `bootstrap`: Inicializa bootstrap.
- `crawler`: Inicializa crawler.
- `identifier`: Inicializa identifier.
- `processor`: Inicializa processor.
- `pipeline`: Inicializa os três comandos acima, por ordem

3.3 config.py, utils.py e languages.json

Foram criados estes ficheiros, com o intuito de não haver repetição de código: *config.py*, *utils.py* e *languages.json*.

Todos os ficheiros *Python* que servem de ajuda aos programas principais (*bootstrap.py*, *crawler.py*, *processor.py* e *identifier.py*) , estão todos guardados na diretoria 'helpers/'.

O ficheiro *config.py* tem as variáveis que são usadas nos vários ficheiros *Python*, daí a decisão de as centralizar num só ficheiro. Podemos observar a seguir, as variáveis utilizadas.

```
import string

LANGUAGES_PATH = 'languages/'
LANGUAGES_FILE = 'languages.json'
LOG_PATH = 'logs/app.log'
PROVISORY_FILES_PATH = 'tmp/'
DOCS_TO_PROCESS_PATH = 'docs/to_process/'
DOCS_PROCESSED_PATH = 'docs/processed/'
PROBABLE_DOCS_PER_LANGUAGE_PATH = 'docs/probable/'
# Simbolos especiais => !"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~
SPECIAL_CHAR = string.punctuation + """ + ' ' + "\n" + "\t"
```

O ficheiro *utils.py* é constituído por todas as funções auxiliares comuns entre módulos, e/ou funções que não fazem sentido existir dentro de uma classe.

```
def get_languages();
def sort_dict_by_key(dictionary);
def clean_word(word);def clean_filename(word);
def random_string(length=16);
def load_json_file(filename, default={});
def save_json_file(filename, dictionary);
def add_to_language_probable_assets(language_slug, text);
```

O ficheiro *languages.json* é constituído pelas línguas a trabalhar. Assim, sempre que queremos iterar sobre as mesmas, temos um ficheiro com a informação centralizada.

```
[
  {
    "slug": "umbundu",
    "names": ["Umbundu", "Umbundo"]
  },
  {
    "slug": "kimbundu",
    "names": ["Kimbundu", "Quimbundo"]
  },
  {
    "slug": "kikongo",
    "names": ["Kikongo"]
  },
  {
    "slug": "tchocue",
    "names": ["Tchocue", "Umbundo"]
  },
  {
    "slug": "portuguese",
    "names": ["Portuguese"],
    "search": false
  },
  {
    "slug": "english",
    "names": ["English"],
    "search": false
  }
]
```

3.4 Logs

Indo de encontro à abordagem de facilitar a integração com outras ferramentas (ex: crons), criamos um sistema de *logs* simples, onde descrevemos todas as operações realizadas pelos programas, assim como *erros* e *warnings* que possam ocorrer. Também é possível visualizar a data da sua execução, como da língua a ser tratada.

A implementação deste sistema traz também a vantagem de termos um registo das execuções do programa.

Todos estes *logs* são guardados no ficheiro *app.log*, na diretoria 'log/'. Vejamos os seguintes exemplos extraídos do ficheiro *app.log*, após uma execução do programa.

[INFO]: 2019-06-25 15:15:38,099 – [Kimbundu] Searching 'uambe tutula munue jimbta'

[INFO]: 2019-06-25 15:20:26,518 – Checking probabilities **for** 'docs/to_process/g1.globo.com_fPGsQsKGShoNfWko.html.txt'

[INFO]: 2019-06-25 15:20:26,520 – Saving text in 'portuguese' probable docs folder as 'g1.globo.com_fPGsQsKGShoNfWko.html.txt.0'

3.5 Processador de textos

Durante a pesquisa, deparámos com textos nos mais diversos formatos e, por vezes, com misturas de línguas (muitas vezes com linhas intercaladas de português e Umbundo/Quimbundo), por isso, o nossa primeira versão do processador, foi um processador manual. Acordamos entre nós um formato a partir do qual iríamos trabalhar e, só depois disso, é que começamos a criar um módulo *Python* que processa-se esse texto de forma a criar uma lista de palavras para cada uma das línguas.

Assim, foi criado o módulos *bootstrap.py*, com o intuito da criação de um dicionário de cada língua, um ficheiro *dictionary.json* com informações úteis como as palavras constituintes da língua, o número de ocorrências, a lista das palavras antecessoras e uma lista com as palavras sucessoras.

A verificação dos textos já processados é essencial, pois o nosso dicionário contém o número de ocorrências de cada palavras, e se voltarmos a processar os mesmos textos, o número de ocorrências iria incrementar, dando valores errados. Assim, é possível verificar as palavras mais comuns de cada língua.

Note-se que quanto melhor e mais rico for o dicionário inicial, melhor serão os resultados.

Podemos observar na seguinte imagem, a execução do *bootstrap.py*, sem nenhum dos ficheiros dados como base processados.

```
[INFO]: 2019-06-25 23:44:58,098 - Bootstrapping 'umbundu' language with [
'bootstrap/umbundu/dicionario.txt',
'bootstrap/umbundu/proverbios-em-umbundu.txt']
[INFO]: 2019-06-25 23:44:58,098 - Adding 'dicionario.txt' words to
'umbundu' language assets
[INFO]: 2019-06-25 23:44:58,104 - Adding 'proverbios-em-umbundu.txt'
words to 'umbundu' language assets
[INFO]: 2019-06-25 23:44:58,111 - Bootstrapping 'kimbundu' language with [
'bootstrap/kimbundu/pai-nosso.txt',
'bootstrap/kimbundu/o-homem-e-a-mulher.txt',
'bootstrap/kimbundu/declaracao-dos-direitos-humanos.txt',
'bootstrap/kimbundu/dic.txt',
'bootstrap/kimbundu/o-homem-e-o-monstro.txt',
'bootstrap/kimbundu/segunda-oracao-e-kalumba.txt',
'bootstrap/kimbundu/o-coelho.txt']
[INFO]: 2019-06-25 23:44:58,111 - Adding 'pai-nosso.txt' words to
'kimbundu' language assets
[INFO]: 2019-06-25 23:44:58,113 - Adding 'o-homem-e-a-mulher.txt'
words to 'kimbundu' language assets
[INFO]: 2019-06-25 23:44:58,116 - Adding
'declaracao-dos-direitos-humanos.txt' words to 'kimbundu'
language assets
[INFO]: 2019-06-25 23:44:58,139 - Adding 'dic.txt' words to 'kimbundu'
language assets
[INFO]: 2019-06-25 23:44:58,208 - Adding 'o-homem-e-o-monstro.txt'
words to 'kimbundu' language assets
[INFO]: 2019-06-25 23:44:58,267 - Adding 'segunda-oracao-e-kalumba.txt'
words to 'kimbundu' language assets
[INFO]: 2019-06-25 23:44:58,329 - Adding 'o-coelho.txt' words to
'kimbundu' language assets
```

Figure 3.2: Execução do bootstrap.py com ficheiro para processar.

Com a seguinte imagem, é possível observar a mesma execução, mas desta vez com os ficheiros já processados.

```
[INFO]: 2019-06-25 22:02:35,523 - Bootstrapping 'umbundu' language
[WARNING]: 2019-06-25 22:02:35,523 - There is no files to add to
'umbundu' language assets
[INFO]: 2019-06-25 22:02:35,523 - Bootstrapping 'kimbundu' language
[WARNING]: 2019-06-25 22:02:35,524 - There is no files to add to
'kimbundu' language assets
[INFO]: 2019-06-25 22:02:35,524 - Bootstrapping 'kikongo' language
[WARNING]: 2019-06-25 22:02:35,524 - There is no files to add to
'kikongo' language assets
[INFO]: 2019-06-25 22:02:35,524 - Bootstrapping 'tchocue' language
[WARNING]: 2019-06-25 22:02:35,524 - There is no files to add to
'tchocue' language assets
[INFO]: 2019-06-25 22:02:35,524 - Bootstrapping 'portuguese' language
[WARNING]: 2019-06-25 22:02:35,524 - There is no files to add to
'portuguese' language assets
[INFO]: 2019-06-25 22:02:35,524 - Bootstrapping 'english' language
[WARNING]: 2019-06-25 22:02:35,524 - There is no files to add to
'english' language assets
```

Figure 3.3: Execução do bootstrap.py sem ficheiro para processar.

Podemos observar na seguinte imagem, um exemplo de um dicionário Kimbundu criado pelo *bootstrap.py*. Cada palavra contém o número de ocorrências, e dois dicionários de palavras, um com as palavras que ocorrem antes da atual e outro com as que ocorrem depois. Cada uma das palavras deste dicionários, têm também o número de ocorrências, respeitando sempre ocorrer antes ou depois, respetivamente.


```

{
  "Otwana": {
    "next": {
      "twamwivu": 1
    },
    "occurrences": 2,
    "prev": {
      "kya": 1
    }
  },
  "Uhangele": {
    "next": {
      "yu": 1
    },
    "occurrences": 2,
    "prev": {
      "umujibha": 1
    }
  },
  "Wahemena": {
    "next": {
      "yu": 1
    },
    "occurrences": 2,
    "prev": {
      "otwana": 1
    }
  },
  "Wasuluka": {
    "next": {
      "ni": 1
    },
    "occurrences": 2,
    "prev": {}
  },
  "Wate": {
    "next": {
      "ophangu": 1
    },
  },

```

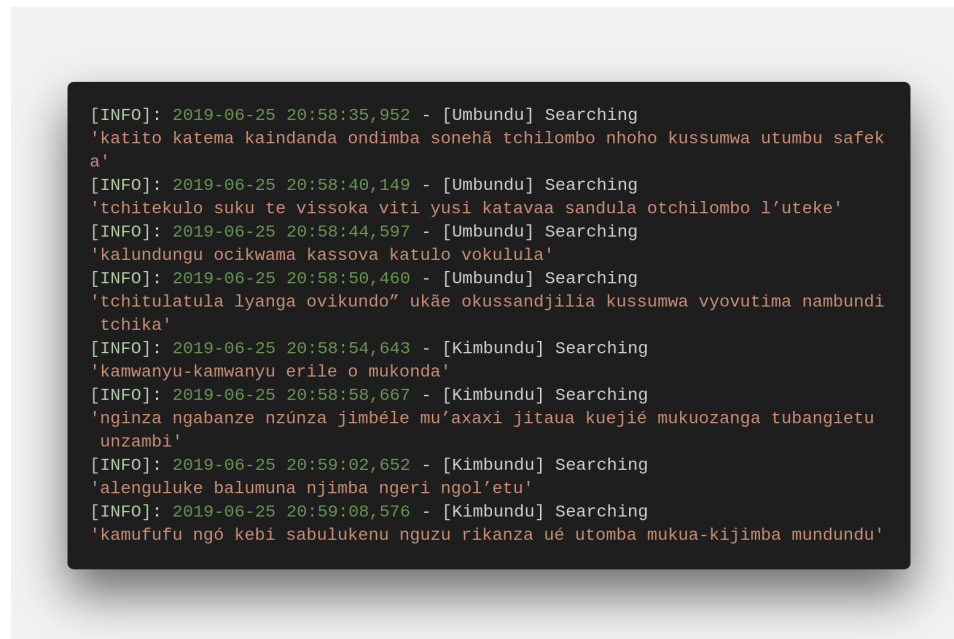
Figure 3.4: Excerto do dicionário Kimbundu.

3.6 Pesquisas Google

Foi criado o modulo *crawler.py* que é responsável pela pesquisa no Google para procurar *urls* com informação relativa a freses, construídas aleatoriamente a partir do dicionário de cada uma das línguas, guardando os resultados para mais tarde serem tratados. Este programa funciona maioritariamente sobre uma API, que nos permite pesquisar no Google, e obter resultados. Esta API pode ser obtida no seguinte link:

<https://github.com/abenassi/Google-Search-API>

Para cada língua, abre-se o seu dicionário, guardando as palavras pertencentes a essa língua. Este método é assim usado, para poder criar uma lista de pesquisas, para encontrar vários tipos de textos na web de cada língua. Na seguinte figura, podemos ver a execução do modulo *crawler.py*.



```
[INFO]: 2019-06-25 20:58:35,952 - [Umbundu] Searching
'katito katema kaindanda ondimba sonehã tchilombo nhoho kussumwa utumbu safek
a'
[INFO]: 2019-06-25 20:58:40,149 - [Umbundu] Searching
'tchitekulo suku te vissoka viti yusi katavaa sandula otchilombo l'uteke'
[INFO]: 2019-06-25 20:58:44,597 - [Umbundu] Searching
'kalundungu ocikwama kassova katulo vokulula'
[INFO]: 2019-06-25 20:58:50,460 - [Umbundu] Searching
'tchitulatula lyanga ovikundo" ukäe okussandjilia kussumwa vyovutima nambundi
tchika'
[INFO]: 2019-06-25 20:58:54,643 - [Kimbundu] Searching
'kamwanyu-kamwanyu erile o mukonda'
[INFO]: 2019-06-25 20:58:58,667 - [Kimbundu] Searching
'nginza ngabanze nzünza jimbéle mu'axaxi jitaau kuejié mukuozanga tubangietu
unzambi'
[INFO]: 2019-06-25 20:59:02,652 - [Kimbundu] Searching
'alenguluke balumuna njimba ngeri ngol'etu'
[INFO]: 2019-06-25 20:59:08,576 - [Kimbundu] Searching
'kamufufu ngó kebi sabulukenu nguzu rikanza ué utomba mukua-kijimba mundundu'
```

Figure 3.5: Execução do modulo *crawler.py*.

Com o output obtido, essa informação é adicionada ao ficheiro *queries.json*, que está dividido de forma a identificar cada frase que foi pesquisada para cada língua, assim como os resultados obtidos a partir dessa pesquisa. Este dicionário contém também um par (chave, valor) que representa o conjunto de *urls* obtidos, marcando cada um com uma *flag* que indica se já foi processado ou não. Esta informação é crucial para que no passo seguinte (*processor.py*) haja controlo sobre os *urls* que já foram visitados, e os que ainda são para processar. Na seguinte imagem, podemos verificar um exemplo do ficheiro *queries.json* com o formato descrito.

```

{
  "umbundu": {
    "amendukussole uk\u00e3
e ndandula suku ongongo osongelako tchitumba otchip\u00e3la kapuka kamenga"
: [
    {
      "name": "Umbundo - mwangol\u00e9
azulhttps://mwangoleazul.blogs.sapo.pt/18540.html",
      "link": "https://mwangoleazul.blogs.sapo.pt/18540.html",
      "google_link": null,
      "description": "28/09/2009 - Mulher \u2013 UK\u00c3E. Crian
\u00e7a \u2013 OMOL\u00c3. ",
      "thumb": null,
      "cached": null,
      "page": 0,
      "index": 0,
      "number_of_results": 3
    }
  ],
  "kimbundu": {
    "kuaxile ngabixila mbumba alenguluke": [
      {
        "name": "Os mist\u00e9rios e os segredos cultura bantu",
        "link":
"https://issuu.com/ceramicapadrao/docs/os_mist_rios_e_os_segredos_cultura_",
        "google_link": null,
        "description": "16/03/2018 - O m'bika'\u00e2
ualengele maz\u00e9",
        "thumb": null,
        "cached": null,
        "page": 0,
        "index": 0,
        "number_of_results": 2
      }
    ]
  },
  "urls":{
    "https://mwangoleazul.blogs.sapo.pt/18540.html": true,
    "https://issuu.com/ceramicapadrao/docs/os_mist_rios_e_os_segredos_cultura":
true
  }
}

```

Figure 3.6: Exemplo do ficheiro queries.json".

3.7 Processamento dos links obtidos

Para o processamento da informação dos links obtidos das pesquisas feitas no passo anterior (*crawler.py*), foi necessária a criação do modulo *processor.py* responsável por extrair documentos *Word*, *PDFs* e *Htmls* sendo estes convertidos para ficheiros texto.

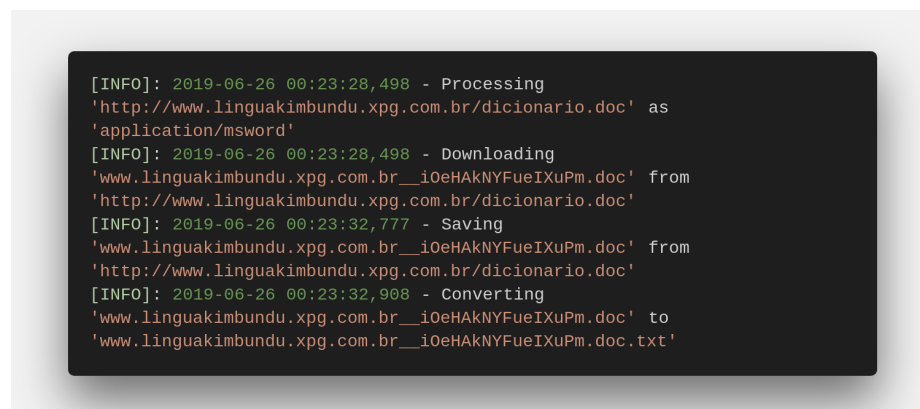
Para facilitar a vida ao utilizador, e também de modo a guardar mais informação, é escrito o link de onde foi extraído no início do ficheiro, no ficheiro

de texto já convertido. Assim, se esse ficheiro contiver informação útil para uma dada língua, o utilizador poderá visitar o link. Para a conversão para ficheiros de texto simples, dos ditos formatos, criamos uma *class* comum para os vários formatos, que foi herdada pelo processador de cada tipo de ficheiro, sendo que estes processadores só tinham de implementar um método que extraísse o texto do ficheiro original. E para isso, utilizamos ferramentas externas, como descritas nas próximas secções

3.7.1 Documentos Word

Para a conversão dos documentos *Word*, foi utilizada a biblioteca *fulltext*. Escolhemos esta biblioteca, tendo em conta o facto de ser das mais recentemente utilizadas, ter suporte *Python3* e o que pesou mais, foi o facto de tratar tanto a extensão *.doc*, assim como a extensão *.docx*.

Um exemplo de extração de um documento *Word*.

A terminal window with a dark background and light green text. It shows a series of log messages with timestamps and status indicators. The messages describe the process of downloading a Word document from a URL, saving it locally, and then converting it to a text file. The URLs and file names are truncated for brevity.

```
[INFO]: 2019-06-26 00:23:28,498 - Processing
'http://www.linguakimbundu.xpg.com.br/dicionario.doc' as
'application/msword'
[INFO]: 2019-06-26 00:23:28,498 - Downloading
'www.linguakimbundu.xpg.com.br__i0eHAKNYFueIXuPm.doc' from
'http://www.linguakimbundu.xpg.com.br/dicionario.doc'
[INFO]: 2019-06-26 00:23:32,777 - Saving
'www.linguakimbundu.xpg.com.br__i0eHAKNYFueIXuPm.doc' from
'http://www.linguakimbundu.xpg.com.br/dicionario.doc'
[INFO]: 2019-06-26 00:23:32,908 - Converting
'www.linguakimbundu.xpg.com.br__i0eHAKNYFueIXuPm.doc' to
'www.linguakimbundu.xpg.com.br__i0eHAKNYFueIXuPm.doc.txt'
```

Figure 3.7: Extração de conteúdo Word”.

3.7.2 Documentos PDF

No caso dos documentos *PDF*, o recurso que utilizamos foi uma chamada ao sistema, que invoca o comando *pdftotext*, extraíndo o resultado dessa execução, para a diretoria *'/tmp'* do sistema.

Posteriormente, a essa invocação é lido o conteúdo deste ficheiro temporário de modo a conseguirmos retornar o conteúdo em texto simples, para a classe base, de onde foi herdada toda a lógica de processamento do ficheiro que vai para além da simples extração do texto.

Temos aqui um pequeno exemplo da execução do *processor.py* com um documento *PDF*.

```
[INFO]: 2019-06-25 21:03:25,838 - Processing
'http://abmanacional.com.br/wp-content/uploads/2017/06/36-4-Erythrina-mulungu
1.pdf'
as 'application/pdf'
[INFO]: 2019-06-25 21:03:25,838 - Downloading
'abmanacional.com.br__eYcaowoLWxuGtYDH.pdf' from
'http://abmanacional.com.br/wp-content/uploads/2017/06/36-4-Erythrina-mulungu
1.pdf'
[INFO]: 2019-06-25 21:03:27,672 - Saving
'abmanacional.com.br__eYcaowoLWxuGtYDH.pdf' from
'http://abmanacional.com.br/wp-content/uploads/2017/06/36-4-Erythrina-mulungu
1.pdf'
[INFO]: 2019-06-25 21:03:27,748 - Converting
'abmanacional.com.br__eYcaowoLWxuGtYDH.pdf' to
'abmanacional.com.br__eYcaowoLWxuGtYDH.pdf.txt'
```

Figure 3.8: Extração de conteúdo PDF”.

3.7.3 Documentos de texto simples

Para ficheiros simples de texto, simplesmente é feito a sua extração, para posteriormente ser tratado. Como já é um ficheiro simples de texto, não é necessário nenhuma conversão.

3.7.4 Páginas WEB

A biblioteca que estamos a utilizar para a extração de texto de páginas *Html* é o *BeautifulSoup*. Para esta escolha, pesou o facto de dar para escolher quais são as *tags html* de onde queremos extrair texto.

O controlo das *tags* é importante dado que, as páginas *html* têm formatos variados, que fogem ao nosso controlo, o que facilmente levaria a que fosse extraído muito conteúdo que não era relevante para o projeto, logo, optamos por extrair apenas o conteúdo dentro de *tags p, h1, h2, h3, h4, h5, h6*. E desta forma, o controlo sobre o texto extraído é relativamente maior.

Esta abordagem não impede que seja extraído conteúdo que não é relevante, mas é, dentro da nossa ótica uma solução muito razoável para o problema apresentado.

Na imagem seguinte, é possível verificar a execução da extração de informação dos links obtidos.

```
[INFO]: 2019-06-25 14:57:32,007 - Processing
'https://maisfutebol.iol.pt/fredy/angola/24-0-18529' as 'text/html'
[INFO]: 2019-06-25 14:57:32,007 - Downloading
'maisfutebol.iol.pt__pmuFOFKot6ITmIVa.html' from
'https://maisfutebol.iol.pt/fredy/angola/24-0-18529'
[INFO]: 2019-06-25 14:57:32,205 - Saving
'maisfutebol.iol.pt__pmuFOFKot6ITmIVa.html' from
'https://maisfutebol.iol.pt/fredy/angola/24-0-18529'
[INFO]: 2019-06-25 14:57:32,295 - Converting
'maisfutebol.iol.pt__pmuFOFKot6ITmIVa.html' to
'maisfutebol.iol.pt__pmuFOFKot6ITmIVa.html.txt'
[INFO]: 2019-06-25 14:57:32,555 - Processing
'https://www.record.pt/futebol/futebol-nacional/liga-nos/belenenses/detalhe/f
redy-e-o-unico-portugues-em-angola-900874'
as 'text/html'
```

Figure 3.9: Extração de conteúdo Html”.

3.8 Identificação da Língua

Para a classificação da língua e extração de dados, foi criado o modulo *identifier.py* que é responsável por verificar em cada ficheiro extraído no passo anterior (*processor.py*), qual é a língua dominante, fazendo esta avaliação paragrafo a paragrafo e guardando esses parágrafos na pasta de documentos com a classificação 'provável' da linguagem predominante, seguindo certos critérios.

O critério utilizado para classificar um texto como pertencente a uma língua, assumindo que o número de palavras do texto é representado por 'comprimento', e o número de palavras desse mesmo texto que já existem no dicionário da língua, onde está a ser feita a identificação, representado por 'contador', seguimos a seguinte fórmula:

$$\text{contador} * 100 / \max(\text{comprimento}, 10)$$

A escolha desta fórmula, foi resultante de vários testes para tentar encontrar um ponto de equilíbrio em que os falsos positivos fossem o menor possível, sem castigar a possibilidade de encontrar novas palavras ara cada língua. Quando o texto é classificado como pertencente a uma língua, é movido para a diretoria 'docs/identified/language_slug/', e as palavras desse texto que não existem no dicionário da língua, são adicionada a um ficheiro 'languages/languages_slug_probable/dictionary.json', para mais tarde serem avaliadas por alguém experiente na língua.

```
[INFO]: 2019-06-25 13:21:09,010 - Checking probabilities for
'docs/to_process/inzotumbansi.org__yZ0jkvdTqYqHsuVb.html.txt'
[INFO]: 2019-06-25 13:21:09,028 - Saving text in 'kimbundu'
probable docs folder as 'inzotumbansi.org__yZ0jkvdTqYqHsuVb.html.txt.0'
[INFO]: 2019-06-25 13:21:09,061 - Saving text in 'kimbundu'
probable docs folder as 'inzotumbansi.org__yZ0jkvdTqYqHsuVb.html.txt.1'
[INFO]: 2019-06-25 13:21:09,079 - Saving text in 'kimbundu'
probable docs folder as 'inzotumbansi.org__yZ0jkvdTqYqHsuVb.html.txt.2'
[INFO]: 2019-06-25 13:21:09,086 - Saving text in 'kimbundu'
probable docs folder as 'inzotumbansi.org__yZ0jkvdTqYqHsuVb.html.txt.3'
[INFO]: 2019-06-25 13:21:09,097 - Saving text in 'kimbundu'
probable docs folder as 'inzotumbansi.org__yZ0jkvdTqYqHsuVb.html.txt.4'
[INFO]: 2019-06-25 13:21:09,103 - Saving text in 'kimbundu'
probable docs folder as 'inzotumbansi.org__yZ0jkvdTqYqHsuVb.html.txt.5'
```

Figure 3.10: Execução do modulo identifier.py”.

Chapter 4

Especificação técnica

Para a execução deste programa, primeiro é necessário a língua de programação *Python3*, como das suas seguintes bibliotecas:

- `git+https://github.com/abenassi/Google-Search-API`
- `fulltext`
- `beautifulsoup4`
- `flake8`

Para uma boa execução deste programa, devemos executar de acordo com a seguinte ordem:

- Primeiro, deve conter textos base na pasta 'bootstrap/', para o *bootstrap.py* poder criar a base de funcionamento do programa (os dicionário base).
- Executar `bootstrap.py`.
- Executar `crawler.py`
- Executar `processor.py`.
- Executar `identify.py`

No fim, todos os ficheiros de texto encontrados e classificados para cada uma das línguas, é guardado em 'docs/identified/language_slug' para mais tarde serem tratados pelo utilizador. É também criado um dicionário com as palavras descobertas em 'language/language_slug' chamado probable-dictionary, onde o utilizador poderá verificar, e adicionar ao textos base do bootstrap, para assim poder aumentar o dicionário base e obter melhores resultados.

Para demonstrar a estrutura geral do nosso projeto, foi criada a seguinte imagem, de forma a explicar facilitar a percepção dos seus componentes.

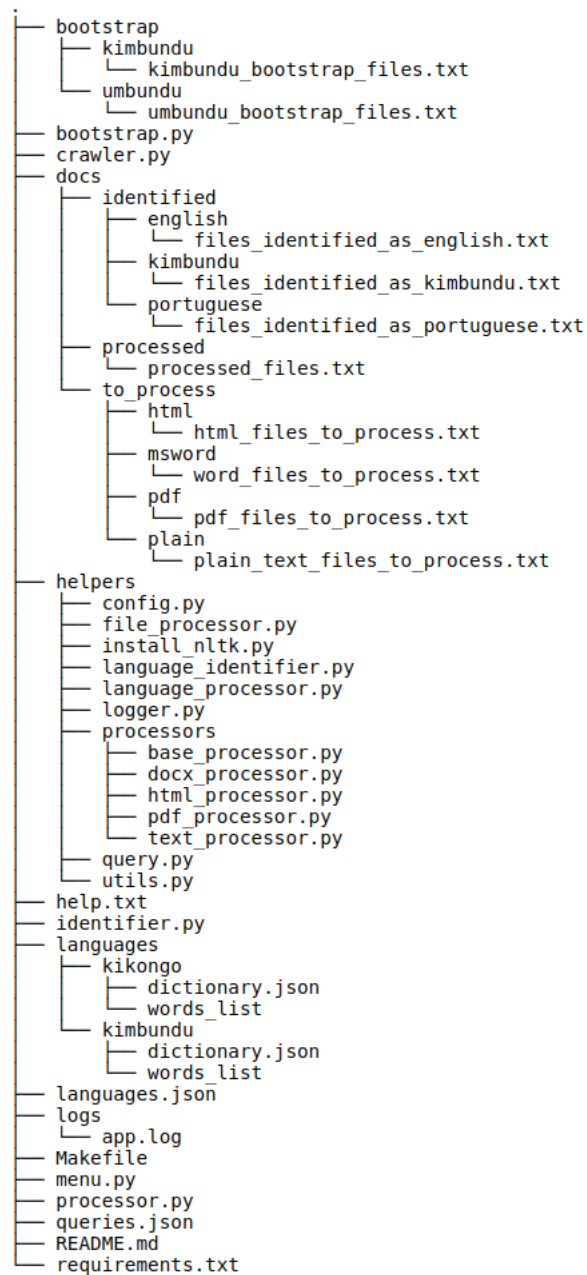


Figure 4.1: Estrutura completa do programa.

Chapter 5

Dificuldades

Um dos grandes problemas para a obtenção de bons resultados, foi a falta de domínio básico das línguas de minorias, pois a obtenção de informação correta e útil é complicada, porque não temos como verificar se os textos que encontramos são realmente de uma dada língua. Isto pode levar à criação uma base de informação errada. A falta de variedade de textos úteis como base, leva a que os resultados do programa, vão geralmente encontrar esses mesmos.

Chapter 6

Próximos passos

Ao analisar o resultados das nossas pesquisas, deparamos com ficheiros de áudio e de vídeo, o que poderá ser uma boa via a explorar o processamento e extração de texto dessas fontes.

Uma boa implementação futura poderá ser uma melhoria no algoritmo de identificação de língua.

Outra boa implementação será adaptar o projeto para correr em *docker*, de modo a ser agnóstico ao sistema operativo da maquina onde está a correr, e não ser preciso a instalação de recursos externos, dado que a imagem *docker* já estaria preparada com esses recursos.

Chapter 7

Conclusão

Com este projeto, era pretendida a criação de um programa, que facilite a procura de informação sobre línguas de minorias, utilizando recursos web. Inicialmente, foram apresentadas as ferramentas e conceitos que iriam ser necessários para a realização do projeto, como o modo da sua implementação. Desta forma, concluímos que o projeto foi realizado de forma satisfatória podendo continuar-se o seu desenvolvimento, como referido na secção "Próximos passos".

Chapter 8

Referências

- <http://natura.di.uminho.pt/~jj/pensador/>
- <http://observinguaportuguesa.org/angola-portugues-e-falado-por-7115-de-angolanos/>
- <https://github.com/abenassi/Google-Search-API>
- <https://www.dicio.com.br/lista-de-palavras/>
- <https://www.ef.com/wwen/english-resources/english-vocabulary/top-3000-words/>
- <http://jornalcultura.sapo.ao/artes/poemas-kimbunduportugues?fbclid=IwAR2IEqE1T6fDwvaqCnu4CQ>
- <https://mweloweto.com/tag/proverbios-em-umbundu/?fbclid=IwAR3lj6TaPE4MdP59nkQipOZ69zySYFsaaimr4zDxU1l09q6qiRNITbSdw>