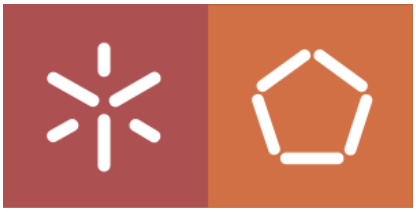

Universidade do Minho
Escola de Engenharia

Universidade do Minho
Mestrado Integrado em Engenharia Informática

Processamento de Linguagens

Machine Learning: datasets de treino

Grupo 4

Ana Afonso
(A85762)

Márcia Teixeira
(A80943)

Pedro Silva
(A82522)

Conteúdo

Lista de Figuras

1 Introdução

O presente projeto enquadra-se na unidade curricular de Processamento de Linguagens, na qual foi proposta o desenvolvimento de um programa em Pyhton que permite a extração de informação de um dataset de treino, apresentando-a em formato *HTML*, de modo a tornar-se visualmente mais apelativa ao ser humano.

Atualmente, a área de Machine Learning é extremamente conceituada e explorada, para além das suas tecnologias serem também utilizadas em diversas áreas. Como tal, a maior parte dos algoritmos concebidos para o efeito têm de ser treinados com um dataset especialmente anotado à mão e, posteriormente, testados sobre outro dataset anotado para ver se o que a máquina descobre é o mesmo que um ser humano faria à mão.

Deste modo, o desafio apresentado visa o cumprimento prático dos principais objetivos demarcados pelo enunciado, designadamente a escrita de Expressões Regulares (ER) para a descrição de padrões de frases dentro de textos, o consequente desenvolvimento de Processadores de Linguagens Regulares ou Filtros de Texto (FT) que filtrem/transformem textos com base no conceito de regras de produção Condição-Ação e a utilização do módulo 'er' (fazendo uso das suas funções) para implementar os FT pedidos.

A fim de garantir o sucesso na elaboração do programa prosposto recorreu-se não só à aprendizagem adquirida em contexto de aula, mas também a incessantes pesquisas que se revelaram fulcrais para a descoberta da melhor forma de cobrir pontos mais intrínsecos a determinados conteúdos, para assim, fornecer uma melhor resposta aos requisitos a cumprir.

2 Análise da problemática

2.1 Descrição detalhada do problema

A problemática residente no presente projeto visa permitir que a informação que até então se encontrava num formato apenas dedicado a cumprir um objetivo, designadamente o treino de algoritmos orientados a Machine Learning, possa ser transformada de modo a que seja passível de ser utilizada para outros possíveis fins. Apresentando-se a leitura facilitada como um exemplo destes, recorrendo aos conteúdos lecionados na unidade curricular até ao momento.

2.2 Decisões tomadas e Implementação

Com o objetivo de desenvolver as funcionalidades necessárias, de uma forma mais intuitiva, decidiu-se envergar numa abordagem ao problema por partes. Perante o supramencionado faseou-se esta tarefa nos seguintes tópicos:

- Desenvolvimento de Expressões Regulares responsáveis por capturar os padrões do dataset.
- Implementação de uma estrutura Condição - Ação para cada caso.
- Utilização dos módulos 'er' para implementar filtros de texto, de forma a tratar as incidências, com o objetivo final de compilar os resultados em ficheiros *HTML* executáveis no *browser*.
- Utilização de uma estrutura de dados para guardar a informação necessária para mais do que uma funcionalidade.

Seguindo o esquema delineado anteriormente, começou-se por desenvolver a Expressão Regular responsável por identificar que tipo de linha está a ser lida no momento, tendo em consideração que, dependendo desta, a ação a tomar será diferente.

Posteriormente, foi elaborada uma estrutura condicional, na qual foram inseridos os vários grupos de captura e consequentes ações a tomar para cada uma. A título de exemplo, salienta-se a existência de um grupo de captura responsável por detetar o início de uma nova categoria, demarcada por "B-Categoria", em que se inicia a alteração seguida do armazenamento da informação de uma nova categoria, no formato "String"(manipulada de forma a constituir uma linha de código *HTML*), que apenas decorrerá assim que a linha lida contenha apenas um "\n".

Estas strings são manipuladas para seguir o padrão de uma página *HTML*, permitindo que o resultado final possa ser visualizado no *browser*.

Para além da conceção dos pontos propostos no enunciado, tomou-se a iniciativa de desenvolver uma página denominada "extras.html", onde a informação foi manipulada de forma a apresentar os dados no formato descrito na segunda figura do enunciado.

```
GENRE:        science fiction films
DIRECTOR:        steven spielberg
ACTOR:        frank sinatra
```

Figura 2.1: Exemplo seguido para o desenvolvimento da página de *HTML* extra

Após a elaboração de todas as funcionalidades pedidas, e dada ambiguidade passível de gerar variadas interpretações, entendeu-se que a contagem dos elementos de cada categoria poderia ser de todos os elementos ou de elementos únicos. A decisão recaiu sobre a apresentação de ambas as informações na página denominada "output.html". Esta página alberga também as hiperligações existentes em cada categoria, que encaminham para cada uma das páginas desenvolvidas para o efeito (por exemplo, para "ACTOR.HTML").

Adicionalmente, foi criado um botão, ainda na página inicial que permite o redirecionamento para a página "extras.html".

Por fim, e tal como pedido, foi também implementado um botão que permite regressar das páginas alusivas às categorias à página inicial ("output.html").

Tendo por base, que o conceito de evolução nos é magnético, foi instintivo a procura de uma nova solução, isto é, uma resposta que fosse capaz de utilizar mais funções subjacentes ao módulo 'er'. Desta forma, tratando-se o processo de desenvolvimento de um projeto uma componente essencial para a aprendizagem e aprimoramento de conceitos, este projeto foi naturalmente um processo iterativo, tendo passado por mudanças que justificaram a construção de uma segunda versão da solução para a problemática.

Esta nova versão debruçou-se sobre o objetivo de desenvolver código que para além dos conceitos falados anteriormente se focasse em outros também lecionados, passando por abranger mais intrinsecamente os módulos 'er'. Mantiveram-se as funcionalidades já desenvolvidas na primeira versão, e foi feita a fragmentação da expressão *regex* de modo a adaptar o código à introdução da utilização de outras funções pertinentes nos módulos supramencionados. Deste modo, ao invés de apenas se utilizar a função *re.match()*, foram utilizadas também as funções *re.sub()* e *re.search()*.

Considera-se, portanto, que esta seja a versão final, por fazer a junção dos conceitos que ainda que pertencentes a passos de aprendizagem distintos, foram lecionados iterativamente em contexto de aula.

3 Codificação

3.1 Alternativas e Problemas de Implementação

Como mencionado em cima foram desenvolvidas duas versões para o enunciado atribuído. A primeira, na qual se usava apenas a função *re.match()* com uma Expressão Regular, que capturava as informações essenciais em grupos de captura. E uma segunda, que recorre às funções *re.match()*, *re.search()* e *re.sub()*, esta última para substituir diretamente nas linhas o escrito na expressão regular.

No que diz respeito, a possíveis contratempos subjacentes à implementação de todo o projeto, compreende-se que todos foram ultrapassados com a dedicação e empenho necessários, o que permitiu à resolução dos mesmos com relativa rapidez e facilidade.

3.2 Resultados obtidos

De seguida apresentam-se os resultados obtidos (visualmente mais intuitivos), a partir dos ficheiros *HTML* gerados, que seguem os requisitos propostos pela equipa docente, bem como uma funcionalidade extra que se considerou pertinente.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset='UTF-8'/>
        <title>Categorias</title>
    </head>
    <body style='background-color: #e6e6ff;'>
        <h2 style='color: #633b97'>Machine Learning: datasets de treino</h2>
        <h3 style='color: #633b97'>Categorias:</h3>
        <ul>
            <li><strong><a href='ACTOR.html' style='color: #535353'>ACTOR</a>: </strong>3220 elementos (1338 únicos)<p></li>
            <li><strong><a href='YEAR.html' style='color: #535353'>YEAR</a>: </strong>2858 elementos (204 únicos)<p></li>
            <li><strong><a href='TITLE.html' style='color: #535353'>TITLE</a>: </strong>2376 elementos (1615 únicos)<p></li>
            <li><strong><a href='GENRE.html' style='color: #535353'>GENRE</a>: </strong>4354 elementos (313 únicos)<p></li>
            <li><strong><a href='DIRECTOR.html' style='color: #535353'>DIRECTOR</a>: </strong>1720 elementos (948 únicos)<p></li>
            <li><strong><a href='SONG.html' style='color: #535353'>SONG</a>: </strong>245 elementos (173 únicos)<p></li>
            <li><strong><a href='PLOT.html' style='color: #535353'>PLOT</a>: </strong>1927 elementos (1219 únicos)<p></li>
            <li><strong><a href='REVIEW.html' style='color: #535353'>REVIEW</a>: </strong>221 elementos (102 únicos)<p></li>
            <li><strong><a href='CHARACTER.html' style='color: #535353'>CHARACTER</a>: </strong>385 elementos (257 únicos)<p></li>
            <li><strong><a href='RATING.html' style='color: #535353'>RATING</a>: </strong>2007 elementos (48 únicos)<p></li>
            <li><strong><a href='RATINGS_AVERAGE.html' style='color: #535353'>RATINGS_AVERAGE</a>: </strong>1869 elementos(167 únicos)<p></li>
            <li><strong><a href='TRAILER.html' style='color: #535353'>TRAILER</a>: </strong>113 elementos (19 únicos)<p></li>
        </ul>
        <button><a href='extra.html'>Extras</a></button>
    </body>
</html>
```

Figura 3.1: Código *HTML* produzido para a página inicial

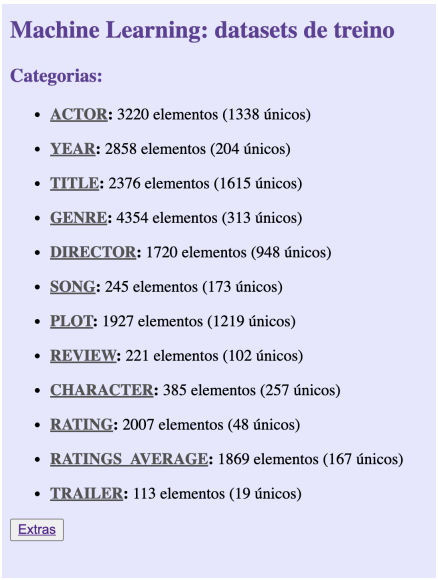

Figura 3.2: Página inicial gerada a partir do código *HTML*

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset='UTF-8'/>
        <title>ACTOR</title>
    </head>
    <body style='background-color: #e6e6ff;'>
        <h2 style='color: #633b97'>Categoria: ACTOR</h2>
        <button><a href='output.html'>Home</a></button>
        <p> </p>
        <ul>
            <li><p>raymond massey  (linhas: 90830, 90831)<p></li>
            <li><p>robert picardo  (linhas: 66002, 66003)<p></li>
            <li><p>lois maxwells  (linhas: 7617, 7618)<p></li>
            <li><p>sean penn  (linhas: 29346, 29347)<p></li>
            <li><p>victoria principle  (linhas: 79599, 79600)<p></li>
            <li><p>sean penn  (linhas: 63478, 63479)<p></li>
            <li><p>john wayne  (linhas: 36616, 36617)<p></li>
            <li><p>betty white  (linhas: 9403, 9404)<p></li>
            <li><p>leonardo decaprio  (linhas: 31429, 31430)<p></li>
            <li><p>kari wuhrer  (linhas: 101148, 101149)<p></li>
            <li><p>danny devito  (linhas: 22923, 22924)<p></li>
            <li><p>elizabeth berkley  (linhas: 55968, 55969)<p></li>
            <li><p>denis leary  (linhas: 44379, 44380)<p></li>
            <li><p>hilary duff  (linhas: 3042, 3043)<p></li>
            <li><p>bob hoskins  (linhas: 7957, 7958)<p></li>
            <li><p>anthony hopkins  (linhas: 8180, 8181)<p></li>
```

Figura 3.3: Amostra do código *HTML* produzido para a página associada à categoria

Figura 3.4: Página associada à categoria gerada a partir do código *HTML*

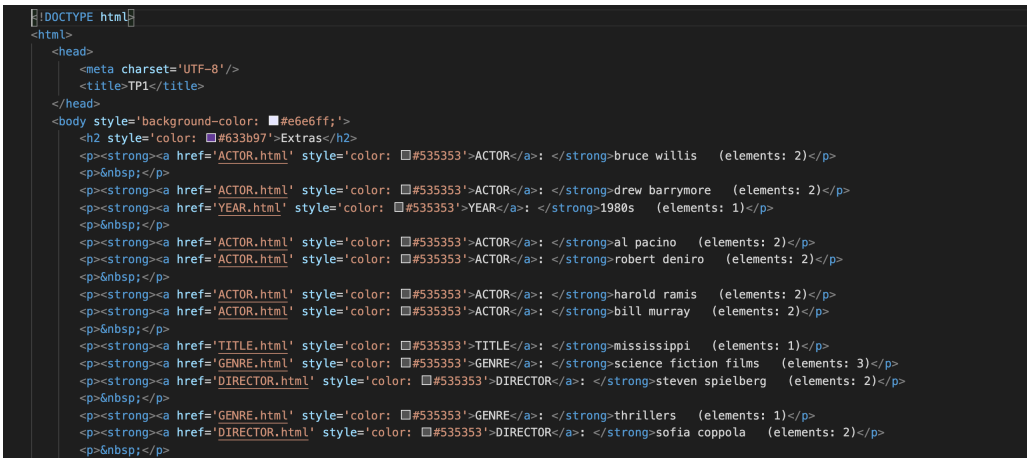

Figura 3.5: Amostra do código *HTML* produzido para a página das informações extra

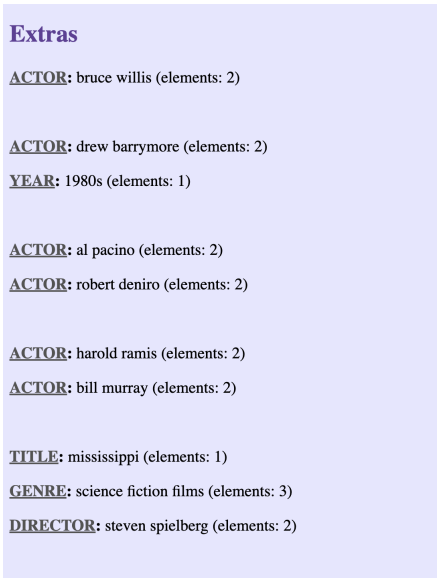

Figura 3.6: Página com informações extra gerada a partir do código *HTML*

4 Conclusão

Em virtude do que foi mencionado, o grupo considera que o projeto desenvolvido vai de encontro ao perspetivado aquando do início minuciosamente ponderado do mesmo.

Durante a realização do projeto, teve-se em consideração o principal objetivo que visa o aprofundamento e experimentação de conceitos lecionados à priori, nomeadamente elaboração de Expressões Regulares e utilizações de funções, como match() e sub(), para implementar os Filtros de Texto pedidos, que transformem textos com base no conceito de regra produção. E, nesse sentido, atingiram-se todos os objetivos definidos inicialmente.

Este projeto revelou-se bastante enriquecedor, uma vez que incentivou não só a aprendizagem, exploração e manuseamento relativo a Expressões Regulares, como também mitigou a capacidade de trabalhar com a linguagem Python.

Não obstantes a futuros aprimoramentos, concebeu-se código de fácil compreensão para promover a vontade e ousadia necessária a realização de posteriores alterações.

Em suma, compreende-se a necessidade da existência de sistemas inteligentes gerados por Expressões Regulares que são capazes de encontrar padrões num texto, sendo estes um gigante e permanente apoio na vida do ser humano em imensas vertentes. Como um ilustre cientista e pensador uma vez mencionou "A informação não é conhecimento. A única fonte de conhecimento é a experiência - Albert Einstein"e como do desenvolvimento vem a experiência, a elaboração deste projeto garantiu-nos esse tão ansiado conhecimento.

A Código do Programa

Lista-se a seguir o código inerente ao programa que foi desenvolvido.

Listing A.1: Código relativo à primeira versão

```
import re

file = open('train.txt', 'r')

num_elems = 0
cat = ""
category = ""
sentence = ""
in_category = 0

line = 0
string_linha = ""

my_categories = dict(set())

my_categories_unrepeated = dict(set())

#### Helper functions ####
def update_categories(category):
    if(category not in my_categories.keys()):
        vals = set()
        my_categories[category] = vals
        vals2 = set()
        my_categories_unrepeated[category] = vals2

def update_in_category(category, value, unrepeated):
    if (category in my_categories.keys()):
        if (unrepeated == 0):
            my_categories.get(category).add(value)
        else:
            my_categories_unrepeated.get(category).add(value)

def showCategory(category):
    cat = my_categories.get(category)
    return cat

def createOutput():
    file = open('output.html', 'w')

    file.write("<!DOCTYPE html>\n")
    file.write("<html>\n")
    file.write("    <head>\n")
    file.write("        <meta charset='UTF-8'/>\n")
    file.write("        <title>Categorias</title>\n")
    file.write("    </head>\n")
    file.write("    <body style='background-color: #e6e6ff;'>\n")
    file.write("        <h2 style='color: #633b97'>Machine Learning: datasets de treino</h2>\n")
```

```
    file.write("        <h3 style='color: #633b97'>Categorias:</h3>\n")
    file.write("        <ul>\n")
    for (cat,val) in my_categories.items():
        unrepeated = str(len(my_categories_unrepeated.get(cat)))
        file.write(f"            <li><strong><a href='{cat}.html' style='color: #535353'>" + cat + "</a>" + ": </strong>" +
       str(len(val)) + " elementos (" + unrepeated + " nicos)<p></li>\n")
    file.write("        </ul>\n")
    file.write("        <button><a href='extra.html'>Extras</a></button>\n")
    file.write("    </body>\n")
    file.write("</html>")
    file.close()

def createFiles():
    for (cat,val) in my_categories.items():
        file = open(cat + '.html', 'w')

        file.write("<!DOCTYPE html>\n")
        file.write("<html>\n")
        file.write("    <head>\n")
        file.write("        <meta charset='UTF-8'/>\n")
        file.write("        <title>" + cat + "</title>\n")
        file.write("    </head>\n")
        file.write("    <body style='background-color: #e6e6ff;'>\n")
        file.write("        <h2 style='color: #633b97'>Categoria: " + cat + "</h2>\n")
        file.write("        <button><a href='output.html'>Home</a></button>\n")
        file.write("        <p> </p>\n")

        file.write("        <ul>\n")
        for v in val:
            file.write("            <li><p>" + v + "<p></li>\n")
        file.write("        </ul>\n")
        file.write("    </body>\n")
        file.write("</html>")
        file.close()


dest_file = open('extra.html', 'w')

dest_file.write("<!DOCTYPE html>\n")
dest_file.write("<html>\n")
dest_file.write("    <head>\n")
dest_file.write("        <meta charset='UTF-8'/>\n")
dest_file.write("        <title>TP1</title>\n")
dest_file.write("    </head>\n")
dest_file.write("    <body style='background-color: #e6e6ff;'>\n")
dest_file.write("        <h2 style='color: #633b97'>Extras</h2>\n")

for linha in file:
    y = re.match(r'(B|I)-([a-zA-Z_]+)(\t| )+([a-zA-Z0-9]+)|(O).+', linha)
    if y:
        if (y.group(1) == 'B'):
            if (in_category == 1):
                if (num_elems != 0):
                    dest_file.write("        " + "<p>" + cat + sentence + "  (elements: " + str(num_elems) + ")</p>\n")
                    update_in_category(category, sentence, 1)
                    sentence += " (linhas: " + string_linha + ")"
                    update_in_category(category, sentence, 0)
                    num_elems = 0
                    category = ""
                    cat = ""
                    sentence = ""
            in_category = 1
            num_elems += 1
            category = y.group(2)
```

```
            cat = f'<strong><a href='{category}.html' style='color: #535353'>" + category + "</a>" + ": </strong>"
            update_categories(category)
            sentence = y.group(4) + ' '
            string_linha = str(line)
        elif (y.group(1) == 'I'):
            num_elems += 1
            sentence += y.group(4) + ' '
            string_linha += ', ' + str(line)
        elif (y.group(5) == 'O'):
            if (in_category == 1):
                if(num_elems != 0):
                    dest_file.write("        " + "<p>" + cat + sentence + "  (elements: " + str(num_elems) + ")</p>\n")
                    update_in_category(category, sentence, 1)
                    sentence += " (linhas: " + string_linha + ")"
                    update_in_category(category, sentence, 0)
                    num_elems = 0
                    category = ""
                    cat = ""
                    sentence = ""
                    in_category = 0
    else:
        if (num_elems != 0):
            dest_file.write("        " + "<p>" + cat + sentence + "  (elements: " + str(num_elems) + ")</p>\n")
            update_in_category(category, sentence, 1)
            sentence += " (linhas: " + string_linha + ")"
            update_in_category(category, sentence, 0)
            num_elems = 0
            category = ""
            cat = ""
            sentence = ""
            in_category = 0
            dest_file.write("        <p> </p>\n")
    line += 1
dest_file.write("    </body>\n")
dest_file.write("</html>")
dest_file.close()

createOutput()
createFiles()
```

Listing A.2: Código relativo à versão final

```
import re

file = open('train.txt', 'r')

num_elems = 0
cat = ""
category = ""
sentence = ""
in_category = 0

line = 0
string_linha = ""

my_categories = dict(set())

my_categories_unrepeated = dict(set())

#### Helper functions ####
def update_categories(category):
    if(category not in my_categories.keys()):
        vals = set()
        my_categories[category] = vals
```

```
        vals2 = set()
        my_categories_unrepeated[category] = vals2

def update_in_category(category, value, unrepeated):
    if (category in my_categories.keys()):
        if (unrepeated == 0):
            my_categories.get(category).add(value)
        else:
            my_categories_unrepeated.get(category).add(value)

def showCategory(category):
    cat = my_categories.get(category)
    return cat

def createOutput():
    file = open('output.html', 'w')

    file.write("<!DOCTYPE html>\n")
    file.write("<html>\n")
    file.write("    <head>\n")
    file.write("        <meta charset='UTF-8'/>\n")
    file.write("        <title>Categorias</title>\n")
    file.write("    </head>\n")
    file.write("    <body style='background-color: #e6e6ff;'>\n")
    file.write("        <h2 style='color: #633b97'>Machine Learning: datasets de treino</h2>\n")
    file.write("        <h3 style='color: #633b97'>Categorias:</h3>\n")
    file.write("        <ul>\n")
    for (cat,val) in my_categories.items():
        unrepeated = str(len(my_categories_unrepeated.get(cat)))
        file.write(f"            <li><strong><a href='{cat}.html' style='color: #535353'>" + cat + "</a>" + ": </strong>" +
       str(len(val)) + " elementos (" + unrepeated + " nicos)<p></li>\n")
    file.write("        </ul>\n")
    file.write("        <button><a href='extra.html'>Extras</a></button>\n")
    file.write("    </body>\n")
    file.write("</html>")
    file.close()

def createFiles():
    for (cat,val) in my_categories.items():
        file = open(cat + '.html', 'w')

        file.write("<!DOCTYPE html>\n")
        file.write("<html>\n")
        file.write("    <head>\n")
        file.write("        <meta charset='UTF-8'/>\n")
        file.write("        <title>" + cat + "</title>\n")
        file.write("    </head>\n")
        file.write("    <body style='background-color: #e6e6ff;'>\n")
        file.write("        <h2 style='color: #633b97'>Categoria: " + cat + "</h2>\n")
        file.write("        <button><a href='output.html'>Home</a></button>\n")
        file.write("        <p> </p>\n")

        file.write("        <ul>\n")
        for v in val:
            file.write("            <li><p>" + v + "<p></li>\n")
        file.write("        </ul>\n")
        file.write("    </body>\n")
        file.write("</html>")
        file.close()

dest_file = open('extra.html', 'w')
dest_file.write("<!DOCTYPE html>\n")
dest_file.write("<html>\n")
dest_file.write("    <head>\n")
```

```
dest_file.write("        <meta charset='UTF-8'/>\n")
dest_file.write("        <title>TP1</title>\n")
dest_file.write("    </head>\n")
dest_file.write("    <body style='background-color: #e6e6ff;'>\n")
dest_file.write("        <h2 style='color: #633b97'>Extras</h2>\n")

for linha in file:
    y = re.match(r'(B|I|O)', linha)
    if y:
        if(y.group(1) == 'B'):
            if (in_category == 1):
                if (num_elems != 0):
                    dest_file.write("        " + "<p>" + sentence + "  (elements: " + str(num_elems) + ")</p>\n")
                    sentence = re.sub(r'.+:.+ (.+)', r'\1', sentence)
                    update_in_category(category, sentence, 1)
                    sentence += " (linhas: " + string_linha + ")"
                    update_in_category(category, sentence, 0)
                    num_elems = 0
                    category = ""
                    sentence = ""
            sentence = re.sub(r'B-([a-zA-Z_]+)(\t| )+([a-zA-Z0-9]+)', r"<strong><a href='\1.html' style='color:
        #535353'>\1:</a></strong> \3", linha)
            in_category = 1
            num_elems += 1
            y = re.search(r'([a-zA-Z_]+):[a-z\/\<\>]+ ([a-zA-Z0-9]+)', sentence)
            if (y):
                category = y.group(1)
                update_categories(category)
            string_linha = str(line)
        elif(y.group(1) == 'I'):
            linha = re.sub(r'I-([a-zA-Z_]+)(\t| )+([a-zA-Z0-9]+)', r' \3', linha)
            sentence += linha
            num_elems += 1
            string_linha += ', ' + str(line)
        else:
            if (in_category == 1):
                if(num_elems != 0):
                    dest_file.write("        " + "<p>" + sentence + "  (elements: " + str(num_elems) + ")</p>\n")
                    sentence = re.sub(r'.+:.+ (.+)', r'\1', sentence)
                    update_in_category(category, sentence, 1)
                    sentence += " (linhas: " + string_linha + ")"
                    update_in_category(category, sentence, 0)
                    num_elems = 0
                    category = ""
                    sentence = ""
                    in_category = 0
    else:
        if (num_elems != 0):
            dest_file.write("        " + "<p>" + sentence + "  (elements: " + str(num_elems) + ")</p>\n")
            sentence = re.sub(r'.+:.+ (.+)', r'\1', sentence)
            update_in_category(category, sentence, 1)
            sentence += " (linhas: " + string_linha + ")"
            update_in_category(category, sentence, 0)
            num_elems = 0
            category = ""
            sentence = ""
            in_category = 0
            dest_file.write("        <p> </p>\n")
    line += 1
dest_file.write("    </body>\n")
dest_file.write("</html>")
dest_file.close()
createOutput()
createFiles()
```