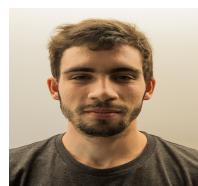


Processamento de Linguagens

Trabalho Prático Nº1 Processador / Sincronizador de legendas



Francisco Lira
A73909



Francisco Costa
A70922

25 de Março de 2018

Conteúdo

1	Contextualização	2
2	Enunciado	3
3	Descrição do Problema	4
3.1	Processador	4
3.2	Sincronizador	4
4	Decisões e Implementação	5
4.1	Processador	5
4.2	Sincronizador	6
5	Conclusão	8

1 Contextualização

Este relatório é o resultado do primeiro trabalho prático proposto e elaborado na unidade curricular de Processamento de Linguagens. O trabalho teve por base a utilização do sistema de produção para filtragem de texto GAWK sobre um ficheiro input previamente fornecido, de modo a responder a uma série de alíneas de um dado enunciado. Dos quatro enunciados apresentados, iremos resolver o terceiro enunciado, **Processador e Sincronizador de Legendas**, uma vez que :

$$(70922 \% 2) + 1 = 3$$

Este enunciado apresenta como input um ficheiro no formato *srt*, com um conjunto de legendas de um determinado filme. Deste modo, o objetivo prende-se com o desenvolvimento de um processador de texto com o auxílio do **GAWK** de forma a que fosse possível ler os ficheiros e transforma-los, obtendo assim os resultados pretendidos.

2 Enunciado

Num ficheiro *.srt*, as legendas possuem a seguinte organização.

- Identificador de legenda
- Intervalo de tempo da legenda
- Legenda

Por exemplo,

```
1  
00:00:48,344 --> 00:00:49,500  
Chamada: recebida  
  
2  
00:00:49,707 --> 00:00:53,128  
-Está tudo pronto?  
-Você não tinha de me substituir  
  
3  
00:00:53,328 --> 00:00:56,014  
Eu sei, mas quero fazer um turno  
  
4  
00:01:06,485 --> 00:01:08,943  
Você gosta dele, não?  
Gosta de observá-lo.
```

1. Construa um processador de srt que:

- retire os identificadores de legenda
- coloque as legendas numa única linha juntando-as com ”
- marque com traço horizontal os intervalos com mais de 2 segundos de silêncio

```
00:00:53,328 --> 00:00:56,014 Eu sei, mas quero fazer um turno.  
00:00:56,014 --> 00:01:06,485 ======  
00:01:06,485 --> 00:01:08,943 Você gosta dele, não?|Gosta de observá-lo.
```

2. Construa um sincronizador de legendas que recalcule os tempos das legendas de sub2.srt de modo que as legendas 12 e 1000 de sub1 fiquem sincronizadas respetivamente com as legendas 8 e 900 de sub2.

Exemplo de uso:

```
srtsync i1=12 i2=8 f1=1000 f2=900 sub1.srt sub2.srt > sub2-sync.srt
```

3 Descrição do Problema

Este trabalho prático, baseia-se na interpretação, filtragem e transformação de texto presente em ficheiros previamente fornecidos e devidamente formatadas. Neste caso concreto, estamos perante um ficheiro *.txt* que contem um conjunto de legendas numa dada linguagem, em que temos a seguinte organização:

- Identificador de legenda
- Intervalo de tempo da legenda
- Legenda(s)

Por exemplo,

```
1  
00:00:48,344 --> 00:00:49,500  
Chamada: recebida  
  
2  
00:00:49,707 --> 00:00:53,128  
-Está tudo pronto?  
-Você não tinha de me substituir  
  
3  
00:00:53,328 --> 00:00:56,014  
Eu sei, mas quero fazer um turno  
  
4  
00:01:06,485 --> 00:01:08,943  
Você gosta dele, não?  
Gosta de observá-lo.
```

3.1 Processador

O primeiro exercício diz respeito à criação de um processador capaz de transformar um ficheiro *.srt*, *SubRipText*, em algo mais simples em termos de visualização. Para tal será necessário, por um lado, retirar todos os identificadores de legenda, agrupar as legendas que aparecem no mesmo intervalo, separados por um "e" e identificar os intervalos de tempo, superiores a dois segundos, em que não existe diálogo, passos que serão à frente explicados mais ao pormenor.

3.2 Sincronizador

O segundo exercício, um pouco mais complexo que o primeiro requer a criação de um sincronizador que faça a sincronização de dois ficheiros de legendas de forma a que duas legendas presentes num ficheiro fiquem sincronizadas com duas determinadas legendas presentes no outro ficheiro, sendo necessário ajustar o intervalo de legendas que seja afetado.

4 Decisões e Implementação

4.1 Processador

O primeiro ponto do processador, pede apenas para retirar os identificadores de legenda. Para isso apenas foi deixar de fora qualquer expressão que fosse incluir o mesmo, como esta linha só tem um número seguido de "\n", foi simples não ter nenhuma Expressão Regular tão abrangente. Para respeitar o segundo ponto, também não tivemos dificuldades. Criamos uma variável, "line", que caso seja na linha anterior tenha sido legenda a variável tem o valor1, se formos imprimir outra linha de legenda verificamos o valor da variável e imprimimos, caso não seja outra linha de legenda deixamos o valor da variável a 0. Já no último ponto, encontramos algumas dificuldades, e com o objetivo de não tornar este exercício algo de difícil percepção, optamos por escrever um pré-processador que altera o estilo em que os tempos são guardados. Depois de passar as legendas no pré-processador, este é o formato em que elas se passam a encontrar:

```
00:03:19:680:00:03:21:360
Deveria saber que você estaria aqui
00:03:22:800:00:03:24:280
Professora McGornigle
00:03:35:360:00:03:38:600
Boa noite Professor Dumbledore
00:03:40:880:00:03:43:560
Os rumores são verdadeiros?
```

Desta maneira torna-se muito mais simples comparar tempos e usar os seus valores.

```
gawk -f 1c.gawk subs.srt > subs_alt.srt
```

Precisamos então de gerar um novo ficheiro, o *subs_alt* que será o ficheiro que o nosso processador final irá receber. Fica então fácil de aceder a todos os campos indicativos de horas, usando apenas \$i. A partir daqui é então simples, o primeiro tempo definido é 0:0:0:0 e este é o valor que está nas variáveis. Também para facilitar os cálculos passamos o valor de tempo para milissegundos e passamos a testar se passaram 2000 milissegundos, o equivalente a 2 segundos.

Este é então o código final em *awk*:

```
BEGIN      {RS="\n";FS=":";ORS=" ";
    timeh=0;timem=0;times=0;timems=0;relms=0;}
/[0-9]{2}:[0-9]{2}/      {line = 0;
    if ($1*3600000+$2*60000+$3*1000+$4>relms+2000){
        print "\n" timeh ":" timem ":" 
                times "," timems " -->";
        print $1 ":" $2 ":" $3 "," $4
                " _____";
        print "\n" $1 ":" $2 ":" $3 "," $4 " -->
                " $5 ":" $6 ":" $7 "," $8;
        timeh=$5;timem=$6;times=$7;timems=$8;
        relms=timeh*3600000+timem*60000+times*1000+timems}
    /[A-Za-z]+/   {if (line==1) print "|";line = 1;print}
END {}
```

4.2 Sincronizador

Relativamente ao Sincronizador, tal como foi apresentado no enunciado, pretende-se percorrer duas vezes o ficheiro *sub2.srt*, uma primeira para guardar os tempos em que aparecem as legendas indicadas, e uma segunda para fazer a sincronização, ou seja, o cálculo dos novos tempos de visualização da legenda.

Antes de passar a qualquer um destes dois passos, procedemos inicialmente à transformação da legenda, onde cada linha do ficheiro será constituída pelo identificador, intervalo de tempo, legendas, de forma a simplificar o processo de análise.

Assim, na primeira parte do sincronizador, para o armazenamento dos tempos, utilizamos a seguinte Expressão Regular, para identificar o *Timestamp* da legenda, no formato

```
[horas]:[minutos]:[segundos]:[milissegundos]
/([0-9]+):([0-9]+):([0-9]+),([0-9]+)-->([0-9]+):([0-9]+):([0-9]+),([0-9]+)/
```

Feito o armazenamento num array *temp*, resta fazer a conversão das várias unidades de tempo para milissegundos e realizar a soma. As várias unidades de tempo estão armazenadas em índices distintos do array, conforme o exemplo.

```
ti1 = temp6[1] * hours_to_milliseconds + temp6[2] * minutes_to_milliseconds
+ temp6[3] * seconds_to_milliseconds + temp6[4];
```

Com os tempos devidamente convertidos, precedemos à segunda leitura do segundo ficheiro. Nesta leitura, é efetuado a leitura do tempo de cada legenda, para que este possa ser atualizado, uma vez que se pretende sincronizar com o primeiro ficheiro.

Assim, é feito o cálculo do novo tempo com recurso à função presente no enunciado:

```
new_ti1 = sync(str_i1,str_f1,str_i2,str_f2,ti1);
new_tf1 = sync(str_i1,str_f1,str_i2,str_f2,tf1);
```

onde,

```
function sync(i1,f1,i2,f2,t) {
    dur1 = f1 - i1;
    dur2 = f2 - i2;
    scale = dur1 / dur2;
    shift = (i2*scale) - i1;
    return ((t * scale) - shift);
}
```

Terminado cálculo do novo tempo, efetuamos a conversão para o formato de legenda [h]:[m]:[s]:[ms]. Para tal, verifica-se se o tempo está acima dos vários valores possíveis para existirem horas, minutos, ou segundos.

Com a legenda no formato correto, resta apenas voltar a colocar o ficheiro com o formato inicial, conforme o exemplo:

```
subtitle_id
HH:MM:SS:MS
Este é o terceiro enunciado do trabalho prático
```

Para tal,

```
time_format_new_ti1 = hi":"mi":"si","mli;
time_format_new_tf1 = hf":"mf":"sf","mlf;
timestamps[$1] = time_format_new_ti1" --> "time_format_new_tf1;
for(i=3;i<=NF;i++) {
    if(i==3) subtitles[$1] = $i"\n";
    else if(i==NF) subtitles[$1] = subtitles[$1]$i"\n\n";
    else subtitles[$1] = subtitles[$1]$i"\n";
}
```

Terminadas todas as alterações no segundo ficheiro, estas alterações são escritas num novo ficheiro quando,

```
END { for(i in subtitles) {
    print i"\n"timestamps[i]"\\n"subtitles[i];
}}
```

5 Conclusão

Durante a realização deste trabalho prático, o primeiro desta unidade curricular, várias foram as etapas que tivemos de passar para chegar ao resultado final.

Apesar da simplicidade e do baixo nível de dificuldade que o trabalho revelou, este foi bastante importante, na medida em que a sua elaboração nos permitiu melhorar a nossa capacidade em escrever Expressões Regulares(ER), e a partir destas desenvolver Processadores de Linguagens Regulares, tendo em vista a filtragem e transformação de textos. Para isto, foi necessário aprender e aprofundar um pouco a linguagem do sistema de produção para filtragem de texto *GAWK* e assim desenvolver as soluções necessárias para cada caso particular das diversas alíneas.

No exercício do processador, tal como já foi falado optamos por obrigar a passagem por um pre-processador e depois o processador. Deste modo ficamos com o trabalho mais simplificado e com o código mais percutível para que possa ser analisado mais facilmente.

Na parte do sincronizador, exercício dois, apesar de termos terminado, achamos que poderíamos ter realizado um melhor trabalho. Apesar de este estar a funcionar, não estamos satisfeitos com a falta de simplicidade da resolução apresentada.

Concluindo, no geral estamos satisfeitos com o trabalho desenvolvido, tendo este se revelado bastante útil no aperfeiçoamento dos nossos conhecimentos no que toca ao tema em questão, à medida que os fomos colocando em prática.