Solutions for Exercise 1

Question 1.1 – Solution in text file 1_1.xq

```
declare function local:convertDate($date as xs:string?) as xs:string
{

if (compare(substring($date, 9, 3), 'Jan') = 0)
then concat(substring($date,6,2),'-01-',substring($date, 13, 4))
else
if (compare(substring($date, 9, 3), 'Feb') = 0)
then concat(substring($date,6,2),'-02-',substring($date, 13, 4))
else
if (compare(substring($date, 9, 3), 'Mar') = 0)
then concat(substring($date,6,2),'-03-',substring($date, 13, 4))
else
if (compare(substring($date, 9, 3), 'Apr') = 0)
then concat(substring($date,6,2),'-04-',substring($date, 13, 4))
else
if (compare(substring($date, 9, 3), 'May') = 0)
then concat(substring($date,6,2),'-05-',substring($date, 13, 4))
else
if (compare(substring($date, 9, 3), 'Jun') = 0)
then concat(substring($date,6,2),'-06-',substring($date, 13, 4))
else
if (compare(substring($date, 9, 3), 'Jul') = 0)
then concat(substring($date,6,2),'-07-',substring($date, 13, 4))
else
if (compare(substring($date, 9, 3), 'Aug') = 0)
then concat(substring($date,6,2),'-08-',substring($date, 13, 4))
else
if (compare(substring($date, 9, 3), 'Sep') = 0)
then concat(substring($date,6,2),'-09-',substring($date, 13, 4))
else
if (compare(substring($date, 9, 3), 'Oct') = 0)
then concat(substring($date,6,2),'-10-',substring($date, 13, 4))
else
if (compare(substring($date, 9, 3), 'Nov') = 0)
then concat(substring($date,6,2),'-11-',substring($date, 13, 4))
else
if (compare(substring($date, 9, 3), 'Dec') = 0)
then concat(substring($date,6,2),'-12-',substring($date, 13, 4))
else ()
} ;



declare function local:parseNews($rss as xs:string) {
let $categories := distinct-values( doc($rss)//item/category)

return <news>
{for $cat in $categories
return
<category name="{$cat}">
{
for $item in doc("DN-Ultimas.xml")//item[category=$cat]
let $date := $item/pubDate
order by $date descending
return
       <item       date="{local:convertDate($item/pubDate)}"       title="{$item/title}"
link="{$item/link}">
       {$item/description/text()}
       </item>
}
</category>
} </news>
};
```

```
local:parseNews("DN-Ultimas.xml");
```

Question 1.2 – Solution in text file 1_2.xq

Note: The function *parseNews* is the same of the exercise 1.1.

```
declare namespace p = "http://www.parlamento.pt"

declare function local:getSessionRelatedNews($parlament, $news, $n as xs:decimal) {
<related-news>
{
for $session in $parlament//p:session
let $sessionSpeeches := concat(for $speech in $session/p:speech
               return ($speech/text(), ' '))
return
       <session date="{$session/@date}">
       {
       for $news_item in $news//item
       let $news_item_copy := $news_item
       let    $result    :=    local:countCommonWords(concat($news_item/text(),    '    ',
$news_item/@title), $sessionSpeeches)
       return if(($result div count(distinct-values(tokenize($sessionSpeeches, '\W+')[. !
= '']))) >= ($n div 100))
               then <item title='{$news_item_copy/@title}' />
               else()
       }
       </session>
}
</related-news>
};

(: counts how many common words are between $arg1 and $arg2 :)
declare function local:countCommonWords($arg1, $arg2) {
let $arg1Words := distinct-values(tokenize(lower-case($arg1), '\W+')[. != ''])
let $arg2Words := distinct-values(tokenize(lower-case($arg2), '\W+')[. != ''])
return count(
       for $w in $arg1Words
       where $w = $arg2Words
       return $w)
};


local:getSessionRelatedNews(doc("Parlamento.xml"),     local:parseNews("DN-Ultimas.xml"),
33);
```

Question 1.3 – Solution in text file 1_3.xq

Note: The function *parseNews* is the same of the exercise 1.1.

```
declare namespace p = "http://www.parlamento.pt"

declare function local:getSessionRelatedNews($parlament, $news) {
<related-news>
{
for $session in $parlament//p:session
return
       <session date="{$session/@date}">
       {
       let $politicians := distinct-values(for $politician in $parlament//p:politician
                              where $politician/@code = $session//p:speech/@politician
                              return $politician)
       for $news_item in $news//item
       for $p in $politicians
       return if(local:countCommonWords($news_item, $p) >= (local:wordCount($p) div 2))
       then <item title='{$news_item/@title}' />
       else()
       }
       </session>
}
</related-news>
};

(: counts how many common words are between $arg1 and $arg2 :)
declare function local:countCommonWords($arg1, $arg2) {
let $arg1Words := distinct-values(tokenize(lower-case($arg1), '\W+')[. != ''])
let $arg2Words := distinct-values(tokenize(lower-case($arg2), '\W+')[. != ''])

return count(
       for $w in $arg1Words
       where $w = $arg2Words
       return $w)
};

declare function local:wordCount($arg as xs:string?)  as xs:integer {
   count(tokenize($arg, '\W+')[. != ''])
};


local:getSessionRelatedNews(doc("Parlamento.xml"), local:parseNews("DN-Ultimas.xml"));
```

Solutions for Exercise 2

Question 2.1 – Solution in text file 2_1.txt

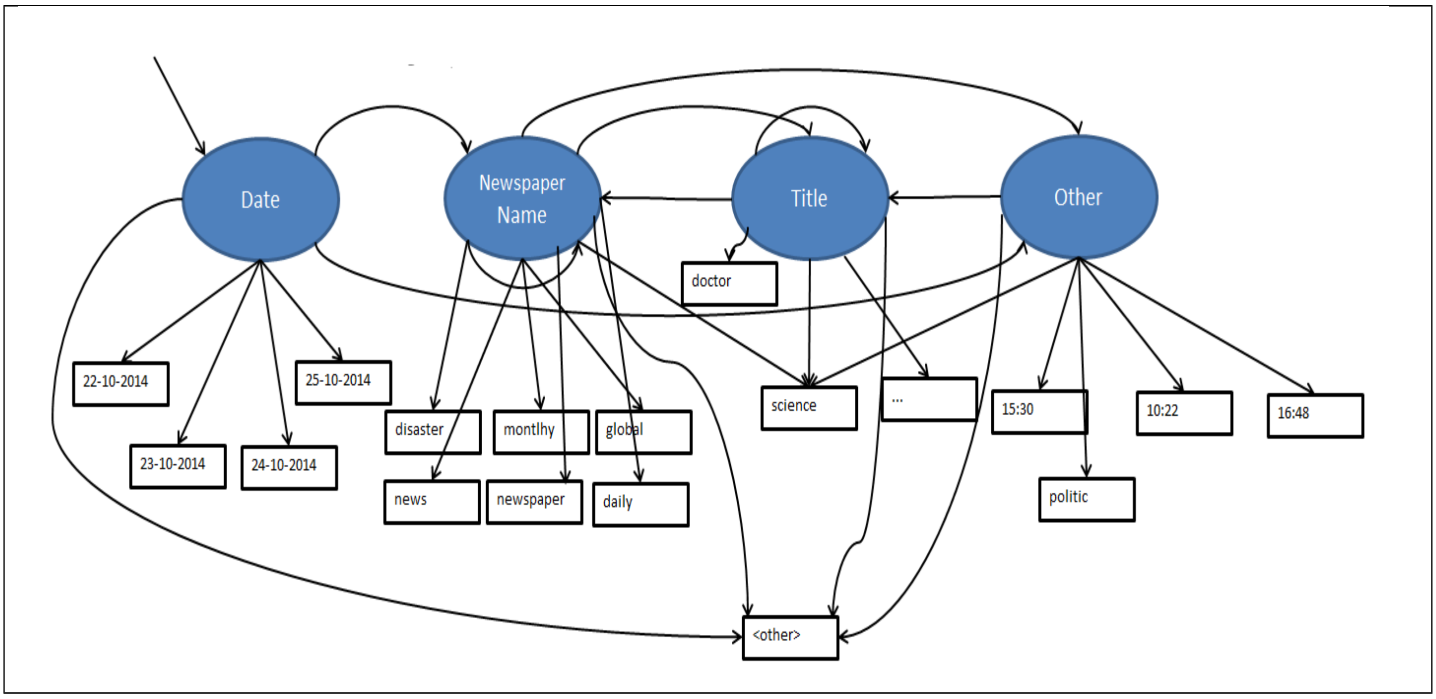
```
<div>
        <span>
                <p>#pcdata</p>
                <p>#pcdata</p>
                <strong>#pcdata</strong>
                <strong>#pcdata</strong>
                <em>#pcdata</em>
        </span>
        (<span>
                <p>#pcdata</p>
                <p>#pcdata</p>
                <strong>#pcdata</strong>
                <em>#pcdata</em>
        </span>)?
</div>
```

Através da aplicação do algoritmo ACME, usando a fig.3 como wrapper e a fig.4 como sample, as primeiras duas tags (div e span) alinham. Em seguida apesar das tags <p> alinharem existe um "string mismatch" que leva a que seja criado o UFRE #pcdata, o mesmo se passa nas seguintes tags ate </span> (<p>,<strong>, <strong>, <em>), a tag </span> alinha com a do wrapper. Depois, existe uma "tag mismatch" que implica o Collapse under Mismatch que primeiro verifica se é uma lista para tentar fazer essa extração, contudo como a segunda span é diferente da primeira não é possível verificar o quadrado para fazer a generalização. Assim, é testado para verificar se é um campo opcional, o que neste caso se verifica e assim a segunda span é opcional. Com a aplicação do algoritmo obtivemos o wrapper generalizado para os segmentos apresentados na fig.3 e fig.4.

Solutions for Exercise 3

Question 3.1 – As palavras usadas para a pergunta 3 foram as encontradas no exemplo sem alteração (e.g., Unkown).

Os estados do modelo correspondem a informação que queremos obter, excepto o "other" que se refere a informação que não nos interessa. No estado "Date" incluímos toda a data e daí retiraríamos o dia e o mês.



Question 3.2 – Solution in text file 3.2

Probabilidades iniciais:

Iniciais	Date	Title	Name	Other
π(Esta-do)	1	0	0	0

Probabilidades de transição:

Transição	Date	Title	Name	Other
Date	0	0	0,4	0,6
Title	0	0,833333	0,1	0
Name	0	0,181818	0,5454545	0,272727273
Other	0	0,5	0	0

Probabilidades de emissão:

Emissão	22-10-2014	daily	disaster	doctor	show	that
Date	0,0416667	0,020833	0,0208333	0,020833333	0,020833333	0,020833333
Title	0,0136986	0,013699	0,0136986	0,02739726	0,02739726	0,02739726
Name	0,0185185	0,037037	0,037037	0,018518519	0,018518519	0,018518519
Other	0,0204082	0,020408	0,0204082	0,020408163	0,020408163	0,020408163

eating	cement	help	digestion	23-10-2014	15:30	unkown
0,020833333	0,020833333	0,02083333	0,020833333	0,0625	0,020833	0,020833
0,02739726	0,02739726	0,02739726	0,02739726	0,01369863	0,013699	0,027397
0,018518519	0,018518519	0,01851852	0,018518519	0,01851852	0,018519	0,018519
0,020408163	0,020408163	0,02040816	0,020408163	0,02040816	0,040816	0,020408

virus	cause	people	less	stupid	global	newspaper
0,020833	0,020833	0,020833	0,020833	0,020833	0,020833	0,020833
0,041096	0,027397	0,027397	0,027397	0,027397	0,013699	0,013699
0,018519	0,018519	0,018519	0,018519	0,018519	0,074074	0,037037
0,020408	0,020408	0,020408	0,020408	0,020408	0,020408	0,020408

science	16:48	cure	unknown	found	german	laboratory
0,020833	0,020833	0,020833	0,020833	0,020833	0,020833	0,020833
0,027397	0,013699	0,027397	0,027397	0,027397	0,027397	0,027397
0,037037	0,018519	0,018519	0,018519	0,018519	0,018519	0,018519
0,061224	0,040816	0,020408	0,020408	0,020408	0,020408	0,020408

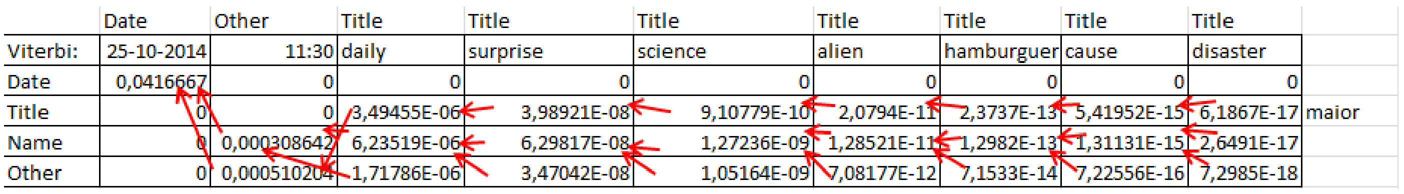
24-10-2014	news	monthly	this	rocket	state	scientist
0,0416667	0,020833	0,020833	0,020833	0,020833	0,0208333	0,020833
0,0136986	0,013699	0,013699	0,027397	0,041096	0,0273973	0,027397
0,0185185	0,037037	0,037037	0,018519	0,018519	0,0185185	0,018519

0,0204082	0,020408	0,020408	0,020408	0,020408	0,0204082	0,020408

25-10-2014	10:22	ready	alien	invasion	minister	defense
0,0416667	0,020833	0,020833	0,020833	0,020833	0,020833	0,020833
0,0136986	0,013699	0,027397	0,027397	0,027397	0,027397	0,027397
0,0185185	0,018519	0,018519	0,018519	0,018519	0,018519	0,018519
0,0204082	0,040816	0,020408	0,020408	0,020408	0,020408	0,020408

politic	<other>
0,020833	0,020833
0,013699	0,013699
0,018519	0,018519
0,040816	0,020408

Question 3.3



	Date	Other	Title	Title	Title	Title	Title	Title	Title	
Viterbi:	25-10-2014	11:30	daily	surprise	science	alien	hamburguer	cause	disaster	
Date	0,0416667	0	0	0	0	0	0	0	0	
Title	0	0	3,49455E-06	3,98921E-08	9,10779E-10	2,0794E-11	2,3737E-13	5,41952E-15	6,1867E-17	maior
Name	0	0,000308642	6,23519E-06	6,29817E-08	1,27236E-09	1,28521E-11	1,2982E-13	1,31131E-15	2,6491E-17	
Other	0	0,000510204	1,71786E-06	3,47042E-08	1,05164E-09	7,08177E-12	7,1533E-14	7,22556E-16	7,2985E-18	

Solutions for Exercise 4

Question 4.1 – Solution in text file 4.1.xq

```
declare namespace ns = "http://www.parlamento.pt"

declare function local:model( $doc ) {

    (:get all partys:)
    let $partys := distinct-values( $doc//ns:politician/data(@party) )



    (:number of interventions:)
    let $interventions_of_party_members := (for $party in $partys
                                     let $number_interventions := count(     for
$speech in $doc//ns:speech, $politician in $doc//ns:politician
                                                                          where
$politician[@party = $party] and $speech[@politician=$politician/@code]
                                                                          return   $speech
        )
                                     return
                                     <party
                                       name="{$party}"
                                       size="{$number_interventions}"
                                     >
                                     </party>   )


   (:get all words:)
```

```
    let $all_words := (      for $speech in $doc//ns:speech
                      let $words := fn:tokenize($speech/text(), "(\.|\!|\?|\,|\:|[ ]+)")
                      return $words  )

        let  $words_normalized  :=  fn:distinct-values(for  $word  in  $all_words  return
if(string($word) = '') then () else fn:lower-case($word))

    (:party with words:)
    let $word_tokens := (     for $word in $words_normalized
                             let $party_word_count := (for $party in $partys
                                                     return <party
                                                            name="{$party}">
                                                              {count(    for  $speech  in
$doc//ns:speech, $politician in $doc//ns:politician
                                                                         where
$politician[@party = $party]
                                                                         and
$speech[@politician = $politician/@code]
                                                                         and
fn:matches(fn:lower-case($speech/text()), fn:concat('\b', $word, '\b'))
                                                                         return $speech
                                                                    )}
                                                            </party>
                                                     )
                             return <word
                                    token="{$word}"
                                    >
                                    {      for $party_word in $party_word_count
                                           return $party_word}
                                    </word>
                      )

    return
         <model>
           {   for $interventions_of_party_member in $interventions_of_party_members
               return  $interventions_of_party_member
           }
           {   for $word_token in $word_tokens
               return  $word_token
           }
         </model>

};


(:local:model(doc("file:///home/antonio/GTI/Proj1/Parlamento.xml")):)
local:model(doc("file:///afs/ist.utl.pt/users/2/1/ist173721/GTI/Proj1/Parlamento.xml"))
```

Question 4.2 – Solution in text file 4.2.xq

```
declare function local:multiplytail($seq, $i, $res) {
  if ($i le 0) then $res
  else local:multiplytail($seq, $i - 1, $res*number($seq[$i]))
};

declare function local:multiply($seq) {
  local:multiplytail($seq, count($seq), 1)
};

declare function local:naive-bayes
  ( $model, $speech as xs:string ) {


  (:vocab size:)
  let $vocab_size := count(   for $word_token in $model//word
                              return $word_token
                        )




   (:each word in speech:)

   let $all_words_in_speech := fn:tokenize($speech, "(\.|\!|\?|\,|\:|[ ]+)")

   let $words_normalized_in_speech := ( for $word in $all_words_in_speech
                                     return
                                       if(string($word) = '')
                                       then
                                         ()
                                       else
                                       fn:lower-case($word) )


    (:naive-bayes:)

    (:party prob:)
    let $total_partys_intervention := fn:sum( for $party in $model/model/party
                                             return $party/@size )
    let $party_probs := (     for $party in $model/model/party
                              return
                              <party_prob
                                party="{$party/@name}"
                                prob="{($party/@size div $total_partys_intervention)}"
                              >
                              </party_prob>
                      )


    (:words prob:)
    let $word_probs := (      for $word_in_speech in $words_normalized_in_speech
                              let $number_of_occ_in_model := fn:count(     for   $word   in
$model//word
                                                                          where
$word[@token = $word_in_speech]
                                                                          return     $word
       )
                              return
                              if($number_of_occ_in_model > 0)
                              then
                                           (for   $word   in   $model//word,   $party   in
$word//party
                                           where $word[@token = $word_in_speech]
                                     let    $total_occ_word    :=    sum(for    $party_2    in
$word//party return number($party_2/text()))
                                     return
                                     <word_prob
                                       token="{$word_in_speech}"
                                       party="{$party/@name}"
```

```
                                        occ="{$party/text() + 1}"
                                        totalocc="{$total_occ_word}"
                                        prob="{($party/text() + 1) div ($total_occ_word +
$vocab_size)}"
                                      >
                                      </word_prob>)
                              else
                                      (for $party in $model/model/party
                                      return
                                      <word_prob
                                        token="{$word_in_speech}"
                                        party="{$party/@name}"
                                        occ="0"
                                        totalocc="0"
                                        prob="{(1 div $vocab_size)}"
                                      >
                                      </word_prob>)
                       )



    (:calc each party prob:)
    let $party_naive_bayes_probs := (       for $party_prob in $party_probs
                                      let   $prob_words_party   :=   (for   $word_prob   in
$word_probs
                                                                 where  $word_prob/@party  =
$party_prob/@party
                                                                return $word_prob/@prob)
                                      return
                                      <naive_bayes            party="{$party_prob/@party}"
prob="{$party_prob/@prob * local:multiply($prob_words_party)}" />
                                  )

   (:return max:)


   let $max := fn:max( for $prob in $party_naive_bayes_probs//@prob return $prob )

         return   (for   $party_naive_bayes_prob   in   $party_naive_bayes_probs   where
$party_naive_bayes_prob[@prob = $max] return $party_naive_bayes_prob)

};

local:naive-bayes(doc("file:///afs/ist.utl.pt/users/2/1/ist173721/GTI/Proj2/model.xml"),
"Reply to the previous reply.")
```

Solutions for Exercise 5

Question 5.1 – Solution in text file 5.1.xq

```
let   $hmm_out   :=   doc("D:\Francisco\IST\Mestrado\1Ano-1Semestre\Gestao   e   Tratamento   de
Informação\Projecto\Parte 2\Exemplo_out_hmm.xml")
let   $ex1_out   :=   doc("D:\Francisco\IST\Mestrado\1Ano-1Semestre\Gestao   e   Tratamento   de
Informação\Projecto\Parte 2\Exemplo_out_ex1.xml")

let $items := ($ex1_out//category,
               <category name="Outros">
                       {for $item in $hmm_out//item
                        let $date := $item/date
                        order by $date descending
                                 return         (<item         date="{$date/day}-{$date/month}"
title="{$item/title}"></item>)
                       }
               </category>)

return (       <news>
               {$items}
       </news> )
```

Question 5.2.1 – Solution in text file 5.2.1.xq

```
declare function local:extract-month( $date as xs:string ) as xs:string {
       let $sub_mes_ano := substring-after($date, '-')
       let $sub_mes := substring-before($sub_mes_ano, '-')
       return (       if($sub_mes = '')
               then $sub_mes_ano
               else $sub_mes)
};

let  $ex1_out  :=  doc("D:\Francisco\IST\Mestrado\1Ano-1Semestre\Gestao  e  Tratamento  de
Informação\Projecto\Parte 2\Exemplo_out_ex1.xml")
let $months := distinct-values (
       for $date in $ex1_out//item/@date
       return local:extract-month($date))

for $month in $months
return  (
               <result month="{$month}">
                       {count( for $date in $ex1_out//item/@date
                               where local:extract-month($date) = $month
                               return $date)}
               </result>
       )
```

Question 5.2.2 – Solution in text file 5.2.2.xq

```
declare function local:extract-month( $date as xs:string ) as xs:string {
       let $sub_mes_ano := substring-after($date, '-')
       let $sub_mes := substring-before($sub_mes_ano, '-')
       return (       if($sub_mes = '')
               then $sub_mes_ano
               else $sub_mes)
};

let  $ex1_out  :=  doc("D:\Francisco\IST\Mestrado\1Ano-1Semestre\Gestao  e  Tratamento  de
Informação\Projecto\Parte 2\Exemplo_out_ex1.xml")
let  $hmm_out  :=  doc("D:\Francisco\IST\Mestrado\1Ano-1Semestre\Gestao  e  Tratamento  de
Informação\Projecto\Parte 2\Exemplo_out_hmm.xml")

let $months := distinct-values (
       (for $date in $ex1_out//item/@date
       return local:extract-month($date),
       for $item in $hmm_out//item
       return $item/date/month))

for $month in $months
return  (
               <result month="{$month}">
                       {count( for $date in (
                                             for $item in $hmm_out//item
                                             let  $item_date  :=  concat($item/date/day,'-',
$item/date/month)
                                             return $item_date,
                                             for $date in$ex1_out//item/@date
                                             return string($date))
                              where local:extract-month($date) = $month
                              return $date)}
               </result>
        )
```