Technical Report

Author:	Bogdan Babych	***Doc ID:***	TRE-2001-Ieper-Babych
Revision:	R1.00	***Date:***	29-Бер-01

Keywords: language recognition; research for pp_tn.dps; letter combination frequency

Title: **Language identification algorithm for disambiguating English and Ukrainian URL and e-mail tokens**

1 About this document

In this report we propose a statistical technique for disambiguating English and Ukrainian tokens in URLs and e-mails

1.1 Abstract

Ukrainian and English tokens in e-mail and URLs have different rules of pronunciation in UKU. We propose to solve the problem of their distinction by calculating relative frequencies of 2-character combinations in Ukrainian and English text corpora and applying these numbers on-line for every 2-character combination in a token under consideration. If the average frequency of all 2-character combinations in the token is higher for the calculations based on the Ukrainian data, then the decision is made that the token should be read according to UKU G2P rules; otherwise, it is sent to the ENU G2P module.

The proposed approach was implemented as an AWK script and tested on the material of about 2000 tokens derived from a 2,6 MB corpus of Ukrainian e-mails and URLs. 771 tokens consisted of actual English and Ukrainian words, for which making the distinction was critical (others were random letter combinations).

The algorithm worked with an error rate **2.6%** for all 2000 tokens, and with the error rate **6.74%** for the 771 tokens consisting of actual words.

1.2 Status

☑ Work document

☐ Accepted by reviewers

1.3 References

[1] Manning, Christopher D., Schuetze, Hinrich. 1999. Foundations of statistical language processing. MIT Press, Cambridge, MA, London, England.

[2] Markov, Andrei A. 1913. An example of statistical investigation in the text of 'Eugene Onyegin' illustrating coupling of 'tests' in chains. In *Procedings of the Academy of Sciences, St. Petersbourg,* vol. 7 of VI, pp. 153-162.

1.4 Contents

2 Introduction

Ukrainian uses Cyrillic character set in all domains, with one exception – the use of Cyrillic characters is not allowed in URLs and E-mail addresses. But Ukrainian words often appear as e-mail and URL tokens in Latin transliteration, though there are no stable rules for this transliteration.

The following problem arises in the development process of the UKU **pp_tn.dps** grammar: URL and e-mail tokens may have different rules of pronunciation depending on their origin.

- The tokens which come from **English** should be read according to ENU G2P rules, that is their **transcription** is mapped to Ukrainian sound system; E.g.: the token *book* should be pronounced as ['bUk], but not as ['bO.Ok]. The token *unknown* should be pronounced as [an-'nOUn], but not as ['un.knOvn].
- The tokens which come from **Ukrainian** are pronounced according to UKU G2P rules, that is their **transliteration** is processed with UKU G2P. E.g.: the token *spozhivach* (Ukr. споживач – consumer) has to be pronounced as [spO.ZI.'vAt&S], but not as ['spaZIvæt&S]. The token *oksamyt* (Ukr. оксамит – velvet) should be pronounced as [O.ksA.'mIt], but not as ['aks$-mIt]. The token *marchuk* (Ukr. Марчук – a family name of a person) should be pronounced as [mA.'rt&Suk] but not as ['mar-k^k] or ['mar-t&S^k].

In the experimental corpus of e-mails downloaded from Ukrainian newsgroups (2.6 MB) there are about 2000 different tokens used in e-mail and URLs. Among them there are 771 (about 40%) tokens that are words and word combinations of either English or Ukrainian language[1]. In order to pronounce these tokens correctly, we need to be able to recognize the language to which a token belongs. The use of dictionaries for this recognition is limited, since tokens often consist of word combinations and of "letter + word" combinations, of proper names and their combinations, etc.

3 The technique of language recognition for disambiguating English words and Ukrainian transliterations

The technique is based on calculating probabilities of 2-letter combinations for English words from ENU text corpus and transliterated Ukrainian words from UKU text corpus. It can be seen as an application of Markov chains for modeling letter sequences[2] of English and Ukrainian [1, p. 317].

0. The preparatory stage consisted of:

- Frequency lists were created from UKU and ENU text corpora.
- The words in the UKU text corpus were transliterated (using one common way of transliteration).
- Non-frequent words very filtered out (to get rid of trash in the frequency lists).
- The "*" symbol was inserted at the beginning and at the end of every word to mark the word boundary.

1. For matrices of all possible two-letter combinations in UKU and ENU two parameters were calculated: (a) relative frequencies of each possible letter combination **in dictionary** and (b)

[1] The other 60% of tokens in the corpus are abbreviations, acronyms and other letter combinations that can be pronounced OK either as English transcriptions or Ukrainian transliterations.

[2] Actually, Andrei A. Markov developed his model in 1913 for similar task – modeling the letter sequences in works of Russian literature [2]; later this model was successfully applied in a different field for speech recognition tasks.

relative frequencies of each possible letter combination **in text** (the sum of frequencies of words where this combination was found). Further we worked only with the first parameter, though for other purposes the second parameter should also be taken into account. (The first parameter is more appropriate for modeling the frequency of letter combinations in URL domain names, where very rare words and proper names may be used).

Word boundary was considered a separate character, so the frequencies of each letter in word initial or word final position were uniformly represented as frequencies of combinations "***L**" and "**L***" (where L stands for all Latin alphabetic characters).

The matrices were created with an AWK script "graphemeComb_ENUv2_0.awk" (see Annex).

2. The frequency matrices of possible letter combinations were used for on-line language recognition of tokens from the UKU e-mail corpus. The language recognition uses the following procedure:

(a) For a token the sum of relative frequencies and an average frequency of letter combinations was calculated on the basis of the ENU letter combination table and on the basis of the UKU letter combination table

(b) The average frequencies for UKU and ENU were compared with each other. If the number for UKU table is bigger, the decision is made that the token represents a Ukrainian word, if the number for ENU table is bigger, the decision is that the token is an English word

An AWK script "CalcWordProbability.awk" (see Annex) performs the on-line recognition of tokens.

4 The results of language recognition for e-mail and URL tokens

The list of about 2000 recognized e-mail and URL tokens was manually checked. The following results were obtained:

- 52 recognition mistakes were found, that means:
- The error rate of the algorithm is **2.6%** for all 2000 tokens
- The error rate of the algorithm is **6.74%** for the 771 tokens that consist of actual English or Ukrainian words

It is interesting to note that the error rate is different for "English" and "Ukrainian" decisions made by the algorithm.

Among 1326 records recognized as **English** there are 526 tokens consisting of actual words.

- 15 mistakes were made for these words; it means:
- the general error rate for "English" decision is *1.13%* for all tokens;
- the error rate is *2.85%* for actual words.

Among 669 records recognized as **Ukrainian** there are 244 tokens consisting of actual words

- 37 mistakes were made; it means:
- the general error rate for "Ukrainian" decision is *5.53%*, for all tokens;
- the error rate is *15.16%* for actual words.

5 Ways of improving the algorithm

The possible directions of research for improving the performance of the algorithm may be:

- Taking into account sequences of 3 or 4 letters (using 2nd or higher order Markov models [1, p.320])
- Calculating *probabilities* of letter combinations and letter sequences (words and tokens), instead of calculating relative frequencies of letter combinations (and the average relative frequency for a letter sequence).

6 Implementation

The algorithm is currently implemented as an AWK script. It needs to be implemented in DEPES as an external C-function (since the current version of DEPES does not support arithmetic operations with integers and real numbers).

It is preferable that the frequency tables were independent from the function and were presented in a text format that is easy to edit and generate with AWK scripts. The implementation of the C-function in this case will be language-independent. The function will work for any pair of languages, for which frequency tables of letter combinations are generated.

7 Annex

7.1 The script “graphemeComb_ENUv2_0.awk” for calculating relative frequencies of letter combinations

```
# prints absolute and relative frequency of each letter combination

BEGIN
{
FS = ";"
ctrlArr[1] =  "a"
ctrlArr[2] =  "b"
ctrlArr[3] =  "c"
ctrlArr[4] =  "d"
ctrlArr[5] =  "e"
ctrlArr[6] =  "f"
ctrlArr[7] =  "g"
ctrlArr[8] =  "h"
ctrlArr[9] =  "i"
ctrlArr[10] = "j"
ctrlArr[11] = "k"
ctrlArr[12] = "l"
ctrlArr[13] = "m"
ctrlArr[14] = "n"
ctrlArr[15] = "o"
ctrlArr[16] = "p"
ctrlArr[17] = "q"
ctrlArr[18] = "r"
ctrlArr[19] = "s"
ctrlArr[20] = "t"
ctrlArr[21] = "u"
ctrlArr[22] = "v"
ctrlArr[23] = "w"
ctrlArr[24] = "x"
ctrlArr[25] = "y"
ctrlArr[26] = "z"
ctrlArr[27] = "*" # - word boundary

# creating the data structure -- 2D array of 2 letter combinations:
for (ltr1i = 1; ltr1i <= 27; ltr1i ++)
        {
        ltr1 = ctrlArr[ltr1i]
        for (ltr2i = 1; ltr2i <= 27; ltr2i ++)
                {
                ltr2 = ctrlArr[ltr2i]
                lArr[ltr1][ltr2] = 0
                dArr[ltr1][ltr2] = 0
                }
        }
}

{
FrqQ = $2

for (i=1; i < length($3); i++)
        {
        chLtr1 = substr($3, i, 1)
        chLtr2 = substr($3, i+1, 1)

        if (chLtr1 in lArr && chLtr2 in lArr[chLtr1])
                {
                # increasing counter for the appropriate element in array
                AllFrq = AllFrq + FrqQ
                AllFrqD++

                lArr[chLtr1][chLtr2] = lArr[chLtr1][chLtr2] + FrqQ
                dArr[chLtr1][chLtr2]++
                }
        else printf "%s%s\n", chLtr1, chLtr2 > "err.log"
        }
}

END
{
for (ch1 in lArr)
        {
        for (ch2 in lArr[ch1])
                {
                RelFrq = lArr[ch1][ch2] / AllFrq
                RelFrqD = dArr[ch1][ch2] / AllFrqD
                printf  "%s%s;%d;%f;%d;%f\n",  ch1,  ch2,  lArr[ch1][ch2],  RelFrq,  dArr[ch1][ch2],
RelFrqD

                }
        }
}
```

7.2 The script “CalcWordProbability.awk” for on-line language recognition of UKU and ENU e-mail and URL tokens

```
BEGIN
{
FS = ";"
while (getline<"uku_Frq_table.txt">0)
        {
        FTabUKU[$1] = $5
        }
while (getline<"enu_Frq_table.txt">0)
        {
        FTabENU[$1] = $5
        }
}
{
WordProbUKU = 0
WordProbENU = 0
for (i=1; i<length($1); i++)
        {
```

```
        Bi = substr($1, i, 2)
        BiProbUKU = FTabUKU[Bi]
        BiProbENU = FTabENU[Bi]
        WordProbUKU = WordProbUKU + BiProbUKU
        WordProbENU = WordProbENU + BiProbENU
        }

AvBiProbUKU = WordProbUKU / i
AvBiProbENU = WordProbENU / i

SumAvBiProb =  AvBiProbENU + AvBiProbUKU
UKUrelProb = AvBiProbUKU / SumAvBiProb * 100

if (UKUrelProb > 50)
        {
        DES = "Ukr"
        }
else {DES = "Eng"}

printf "%s;%f;%f;%f;%f;%f;%s\n", $1, WordProbUKU, AvBiProbUKU, WordProbENU, AvBiProbENU,
UKUrelProb, DES
}
```

7.3 Other material

Frequency tables of character combinations, the frequency list, testing material and service AWK scripts could be found in a shared directory on \\LIBNA:

\\LIBNA\PROJETS\RS_pptn.dps\M_Chain\