INTRODUCTION

In this module, brute force “transliteration” of consonants is used (alongside an algorithm for measuring the difference between two sequences*) in order to compare and match Tajik-Persian text strings:

```
>>>trans_words('Фориғ зи умеди раҳмату бими азоб', 'tg')
('фрғ', 'з', 'мд', 'рмтф', 'бм', 'зб')
>>>trans_words('فارغ ز امید رحمت و بیم عذاب', 'fa')
('фрғ', 'з', 'мд', 'рмт', 'ф', 'бм', 'зб')
```

Following notations are used:

- ‘tg’ – Tajik;
- ‘fa’ – Persian;
- ‘tg_comp’, ‘fa_comp’ – “translated” strings (as shown above) .

---

*Levenshtein ratio

TEXT COMPARISON

This module presents 3 functions to compare and match Tajik-Persian text strings:

1. `match_texts(tg_texts, fa_texts, min_ratio) -> list`

   This function returns a list of best-matched pairs between a set of text strings in Tajik (texts_tg) and a set of text strings in Persian (texts_fa) based on their “transliterations” similarity. Non-matched text strings are dropped.

2. `match_lines(tg_text, fa_text, min_ratio, window) -> list`

   This function returns a sequence of matching pairs between a sequence of text strings in Tajik (text_tg) and a sequence of text strings in Persian (text_fa) based on their “transliterations” similarity and relative order of text strings in sequences. Non-matched text strings are dropped. The parameter `window` determines the number of text strings compared at each step.

3. `match_words(tg_line, fa_line, window, seq_len) -> list`

   This function returns a sequence of matching pairs between a text string in Tajik (tg_line) and a text string in Persian (fa_line) and the corresponding sequences of “transliterations” based on their similarity and relative order of words in sequences. Non-matched parts of text strings are left in their relative positions. The parameter `window` determines the number of words compared at each step, while parameter `seq_len` sets the maximal possible length of a compound word.

MATCH MANIPULATION

This module presents a ParallelText class to manipulate Tajik-Persian matched pairs of text strings:

```
>>>tg = 'Фориғ зи умеди раҳмату бими азоб'
>>>fa = 'فارغ ز امید رحمت و بیم عذاب'
>>>match = ParallelText(match_words(tg, fa))
>>>match
-----------------------------------------
Фориғ | зи | умеди | раҳмату | бими | азоб
فارغ  | ز  | امید  | رحمت و  | بیم  | عذاب
 1.0  |1.0 |  1.0   |   1.0   | 1.0  | 1.0
-----------------------------------------

>>>match[1:3]
----------
зи | умеди
ز  | امید
1.0| 1.0
----------

>>>match.pop(2)
------
умеди
امید
1.0
-----

>>>match.find('fa', 'ز')
(1,)

>>>match.ratio()
1.0
```

```
>>>list(match)
['Фориғ зи умеди раҳмату бими азоб',
'فارغ ز امید رحمت و بیم عذاب',
'фрғзмдрмтфбмзб',
'фрғзмдрмтфбмзб',
1.0]

>>>match.split_by_ind((1, 4)):
[-----
Фориғ
فارغ
1.0
------,
---------------------
зи | раҳмату | бими
ز  | رحمت و  | بیم
1.0|   1.0   | 1.0
---------------------,
------
азоб
عذاب
1.0
------]

>>>match.split_by_size(2)
[--------------------
Фориғ | зи | раҳмату
فارغ  | ز  | رحمت و
 1.0  |1.0 |  1.0
---------------------,
------------
бими | азоб
بیم  | عذاب
1.0  | 1.0
------------]
```

```
>>>match2 = ParallelText([('ва',
'که',
>>>trans_words('که', 'tg'),
>>>trans_words('که', 'fa'))])
>>>match = add_ParallelTexts((match[:1], match2, match[1:]))
>>>match
---------------------------------------
Фориғ | ва | зи | раҳмату | бими | азоб
فارغ  | که | ز  | رحمت و  | بیم  | عذاب
1.0   |0.0 |1.0 |  1.0    | 1.0  | 1.0
---------------------------------------

>>>match.drop_by_ratio(1.0)
[-----
Фориғ
فارغ
1.0
------,
---------------------------
зи | раҳмату | бими | азоб
ز  | رحمت و  | بیم  | عذاب
1.0|   1.0   | 1.0  | 1.0
---------------------------]

>>>match.data['tags'] = ['START', '', '', '', '', 'END']
>>>match.names += ('tags',)
>>>match.show += ('tags',)
>>>match
---------------------------------------
Фориғ | ва | зи | раҳмату | бими | азоб
فارغ  | که | ز  | رحمت و  | بیم  | عذاب
1.0   |0.0 |1.0 |  1.0    | 1.0  | 1.0
START |    |    |         |      | END
---------------------------------------
```

TEXT PREPROCESSING

While this module doesn’t focus on text preprocessing, it does contain a few tools that can be used for that purpose:

```
>>>tg = 'Фориѓ зи умеди рањмату бими азоб'
>>>tg.translate(norm_tg)
'Фориғ зи умеди раҳмату бими азоб'


>>>tg = 'Фориғ зи умеди раҳмату бими азоб, Озод зи хоку бод в-аз оташу об!'
>>>remove_punctuation(tg)
'Фориғ зи умеди раҳмату бими азоб Озод зи хоку бод в-аз оташу об'


>>>tg = 'кӯдакону навҷавонони байни 6 то 15 сол ташкил медиҳад'
>>>replace_digits(tg, 'tg')
'кӯдакону навҷавонони байни ~ то ~ сол ташкил медиҳад'


>>>tg = 'Фориғ зи умеди раҳмату бими азоб, Озод зи хоку бод в-аз оташу об!'
>>>split_lines(tg)
('Фориғ зи умеди раҳмату бими азоб, Озод зи хоку бод в-аз оташу об',)
```