Tokenizer Report

This report is the summary of the 4 word tokenizers in Farsi

- Parsivar
- Hazm
- Farsi Veb Tokenizer
- SetPer

Parsivar

Normalization and Tokenization Problems

different character encodings

One of the main problems in Persian text processing is the existence of **different character encodings** in text
documents. For example, the word water (آب), might have different encodings in different documents.

To solve this problem, for each character we extracted all different encodings from a corpus of text documents with more than 2 million documents which gathered from Persian weblogs.

finding word boundaries

In Persian language, multi token words can be written in three formats

- completely separated by a space delimiter
- separated by half-space
- attached to each other

Therefore, determining word and phrase boundaries is a complicated task in Persian. The challenge in Persian is that the **space cannot be considered as the only delimiter in all cases**.

Solutions:

- Rule based space correction:
    - some certain rules have been defined in the first step using **regular expressions** .
    - There are still some words that do not match to the rules. These words usually consist of two or three parts which we can't extract a general rule for them such as "گفت و گو " (Conversation). To overcome this problem, we **construct a dictionary** containing such words and check their
      existence in the sentences.
- Space correction based on learning :
    - training set using 90% of **Bijan khan** corpus (**Bijan khan** et al. 2011).
    - tagged the multi token words using IOB tagging format.
    - 96.5% of F1-score

Tokenizer Code

```
class Tokenizer():

    def __init__(self):

        pass

    def tokenize_words(self, doc_string):

        token_list = doc_string.strip().split()

        token_list=[x.strip("\u200c") for x in token_list if
len(x.strip("\u200c"))!= 0]

        return token_list

    def tokenize_sentences(self, doc_string):

        pattern = r'([!\.\?؟:]+)[\n]*'

        doc_string = re.sub(pattern, '\t\t', doc_string)

        pattern = r'[\n]+'

        doc_string = re.sub(pattern, '\t\t', doc_string)

        doc_string = doc_string.split('\t\t')

        return doc_string
```

Hazm

WordTokenizer

this class inherits from python **NLTK** *TokenizerI* class and uses new Farsi *words_file* & *verbs_file* & after verb_ & *before verb* lists .

in tokenize functions it replace links, emojis,ids,emails,hashtags and numbers with right formats and then split the sentence by "spcace", it also has a feature for joining the verbs to represent complete verb in one single item of list.

Farsi verb Tokenizer

Tokenizer

it has some resources

- past stems
- present stems
- transformations [patterns of Farsi verbs]

by checking the all patterns in every sentences , for every pattern replace word(s) with new token.

samples:

input	output
هر روز از آنها پول خواسته نمی‌شده است.	هر روز از آنها پول خواسته_نمی‌شده _ است.

SeTPer

RegEx based Word Tokenizer using PERL

Word Tokenizer

```
#!/usr/bin/perl

binmode(STDIN,":utf8");
binmode(STDOUT,":utf8");

while (<>){

    # tokenization

    s/(\P{P})(\p{P}[\p{P}\s\N{U+200C}]|\p{P}\Z)/$1 $2/gs;
    s/(\P{P})(\p{P}[\p{P}\s\N{U+200C}]|\p{P}\Z)/$1 $2/gs;
    s/(\A\p{P}|[\p{P}\s\N{U+200C}]\p{P})(\P{P})/$1 $2/gs;
    s/(\p{P})(?!\1)/$1 $2/gs;
    s/[ \N{U+200C}]{2,}/ /gs;

    print;
}
```