REPORT
NLP COURSE - WEEK 1

Group 1:

- Nguyen Thac Thong - K57CA
- Phan Xuan Tien - K57CA
- Nguyen Tung Lam - K57CA
- Nguyen Xuan Nam - K57CA
- Truong Quoc Tuan - K57CA
- Le Van Giap - K57CA
- Nguyen Van Giap - K57CA
- Nguyen Tuan Phong - K57CA

1. Regex description

Regex is the abbreviation for *regular expression*. A regex defines a search pattern for strings. The search pattern can be anything from a simple character, a fixed string or a complex expression containing special characters describing the pattern. The pattern defined by the regex may match one or several times or not at all for a given string.

We defined some following simple regexes:

```
URL = "(https?://)?[\\w-~]+(\\.[\\w-~]+)+(:\\d{1,5})?(/[\\w-~]*)*";

EMAIL = "([\\w\\d_\\.-]+)@(([\\d\\w-]+)\\.)*([\\d\\w-]+)";

PHONE_NUMBER = "(\\(+\\d+\\))?[\\d\\.\\-]+";

NUMBER = "[-+]?\\d+([\\.,]\\d+)*(%)?";

DATE = "(0?[1-9]|[12]\\d|3[01])(\\/|-|.)(0?[1-9]|1[0-2])((\\/|-
                                                |.)((19|20)\\d\\d))?";
```

2. Algorithms

a. Word tokenization

We defined a list of special characters as below:

```
specialChar = {";", "…", "...", "&", "?", "!", "_", "=", "\"", "'",
      "\\", "<", ">", "[", "]", "{", "}", "‘", "’", "“", "”", "*",
      "~","^", "|"}
```

We also defined other special charaters:

```
otherSpecialChar = {"+", "-", ")", "(", "/", ":", ".", ","}
```

Here is the pseudo code for this algorithm:

```
procedure WordTokenization (input: line)
      if line has special charaters
            add 2 whitespaces before and after each special charater
      words = empty list
      tokens = tokenize line by whitespace
      for each token in tokens
            if token matches regexes
                  add token to words
            else
```

```
                    add 2 whitespaces before and after each special
                                                          charater
        return words
```

b. Sentence segmentation

```
    procedure SentenceSegmentation (input: words)
        sentences = empty list
        sentence = empty list
        for each word in words
            add word to sentence
            if word is “end of string” // in {".", "?", "!", "EOS"}
                normalize sentence to a string
                add sentence to sentences
                clear sentence to prepare for next round
        return sentences
```

Exceptions of Sentence Segmentation algorithm are listed as below:

String	Example
.	Dưới đây là những thực tế về sex sau khi kết hôn mà chị em nào cũng biết: 1.

3. Implementation

We implemented the algorithms in Java. Our source contains three major classes: *Main.java*, *WordTokenizer.java* and *SentenceTokenizer.java*.

WordTokenizer.java and *SentenceTokenizer.java* implements the two algorithms above.

The class *Main.java* handles the command line argument. Firstly, it takes the path of input text file from the argument and then adds all the lines in the file to a list. Secondly, it calls the method tokenize of class *WordTokenizer* to tokenize each line of the list and then adds all the words to another list. Finally, the class *SentenceTokenizer* helps to build the list of sentences from the list of words above. The list of words and the list of sentences will be saved to file immediately.

Besides, there are also some helper classes such as *Regex.java* (manages the regexes), *Dictionary.java* (manages the exceptions), etc.

4. Experience

Data source: We crawled articles from website http://www.chudu24.com/ and stored them in file *chudu24.txt* whose size is approximately 10Mb.

The source code of our program is available at this github repository: https://github.com/tuantq57/nlp-group1.

Result:

- Total lines: 23337
- Total sentences: 92156
- Total tokens: 47970
- Total words: 1925155

Runtime:

- Word tokenization time: 24776ms
- Sentence segmentation time: 1673ms
- Total runtime: 31442ms

We also have a function to compute frequency for each word. Here are some top-rated words:

Word	Frequency
,	87306
.	80386
có	28745
và	26768
khách	24784
sạn	24520
phòng	23292
không	21985
rất	20841
tôi	19644