Processing Raw Text

Processing Raw Text

Introduction & Goals

- What is Raw Text? And Why does it need to be Processed?

- The Goal of this chapter is to…

Processing Raw Text

Introduction & Goals

- What is Raw Text? And Why does it need to be Processed?
  - In computing, Raw text(or Plain text) is only a linear sequence of chara cters with no representation
  - Yet in Natural Language, a series of characters has a graphical represe ntation of itself
  - Thus the Processing is needed to give the Raw text a graphical represe ntation by being segmented into linguistic units
- The Goal of this chapter is to…

Processing Raw Text

Introduction & Goals

- What is Raw Text? And Why does it need to be Processed?

- The Goal of this lecture is to…
  - To access text from local files and from the Web
    - for unlimited range of language material to access

  - Split documents up into individual words and punctuation symbols
    - In order to carry out same kinds of analysis' with other text collections

  - Write programs to produce formatted output and save it in a file.

Accessing Text

- From the web

```
>>> from urllib import urlopen
>>> url = "http://www.gutenberg.org/files/2554/2554.txt"
>>> raw = urlopen(url).read()
>>> type(raw)
<type 'str'>
>>> len(raw)
1176831
>>> raw[:75]
'The Project Gutenberg EBook of Crime and Punishment, by Fyodor Dostoevsky\r\n'
```

※ Note: The sample text is that from Project Gutenberg from the NLTK Corpus collection (http://www.gutenberg.org/catalog/)

Accessing Text

```
>>> from urllib import urlopen
>>> url = "http://www.gutenberg.org/files/2554/2554.txt"
>>> raw = urlopen(url).read()
>>> type(raw)
<type 'str'>
>>> len(raw)
1176831
>>> raw[:75]
'The Project Gutenberg EBook of Crime and Punishment, by Fyodor Dostoevsky\r\n'
```

The type of the text is String(str)
And it's length is 1,176,831 characters(ASCII)

※ Note: The sample text is that from Project Gutenberg from the NLTK Corpus collection (http://www.gutenberg.org/catalog/)

Accessing Text

```
>>> from urllib import urlopen
>>> url = "http://www.gutenberg.org/files/2554/2554.txt"
>>> raw = urlopen(url).read()
>>> type(raw)
<type 'str'>
>>> len(raw)
1176831
>>> raw[:75]
'The Project Gutenberg EBook of Crime and Punishment, by Fyodor Dostoevsky\r\n'
```

The 1,176,831 characters contains many unneeded details.
We need to break up the string into words and punctuation

Accessing Text

```
>>> from urllib import urlopen
>>> url = "http://www.gutenberg.org/files/2554/2554.txt"
>>> raw = urlopen(url).read()
>>> type(raw)
<type 'str'>
>>> len(raw)
1176831
>>> raw[:75]
'The Project Gutenberg EBook of Crime and Punishment, by Fyodor Dostoevsky\r\n'
```

The Process is called **Tokenization**

What is Tokenization?

- Definition of Tokenization
  - -> The Process of Breaking a stream of text up into tokens
  - For

    Information retrieval
    Information extraction
    Spell-checking
    Text-to-speech synthesis

- An early step of processing is to divide the input text into units called ***to kens*** where each is either a word or something else like a number or a punctuation mark

- Tokenization is generally considered as easy relative to other tasks in natural l anguage, and one of the more uninteresting tasks

So what counts as a Token?

- Token
  - Emerged as the first currency system
  - In modern Computer Science it substitutes a sensitive data element with a non-sensitive equivalent. Allowing more diverse uses
  - It is now used in Economics, Data security, Big Data References and…NLP

- A token is… Linguistically significant yet something simple

  Methodologically useful yet useless by its own

- In NLP a Token may be a word, number, or punctuation mark
  - Basically anything that may count as a unit of language inside a text.
  - Punctuation mark: Ex) period(.), comma(,), semicolon(;) etc.

So How is it Done?

- Tokenization usually relies on simple heuristics to maximize it's effect iveness.
  - Ex) Contiguous strings of characters are part of one token

- For example
  - Input: When life gives you lemons, don't make lemonade.
  - Output: [When][life][gives][you][lemons][,][don][‘][t][make][lemonade][.]
  - -> While the sequence of characters are isolated, unneeded detail still exits.

- Because of Language specific issues, Language Identification is reco mmended & highly effective.

Basic Steps in Tokenization

1. One must **Specify which Language** one will be Tokenizing.

2. The text should be Segmented into words, sentences and etc.
   - Ex) [When][life][gives][you][lemons][,][don][‘][t][make][lemonade][.]

3. Now we have to get rid and fix aberrations such as [.],[,],[‘],[“] etc.
   - Ex) Make [don][‘][t] into [don’t] and extinguish the unneeded [,],[.],[‘].

4. Finally we select the tokens from the isolated lists to use.

Language Identification

- Though many methods exist for Language Identification, tokeniz ation only requires what form the language is written.

- Most written texts are usually one of…
  - Languages with inter-word spaces or similar
  - Languages written in Scriptio continua
  - Languages written in combinations of alphabets

- It is also important to know which direction to read
  - Many East Asian scripts can be written horizontally or vertically
  - The Arabic alphabet is written and read ‘right to left’.

Basic Steps in Tokenization

Language Identification

Languages with inter-word spaces

- Words are divided by space s or similar marks.
- Fairly straight-forward
- Most widely studied and kno wn

Ex) Latin based languages, Ko rean, English, French, German etc.

Written in Scriptio continua

- A style of writing without s paces or marks.
- Also lacks punctuations o r distinguished letter case s.
- Difficult to distinguish indi vidual words

Ex) Chinese, Japanese, Cla ssical Latin etc.

Combinations of Alph abets

- Combines alphabets to cr eate each syllable.
- Tend to have a large varie ty of syllables
- Alphabets looks may cha nge drastically.

Ex) Korean Hangul

※ Note that languages may fit in to more than one of these archetypes.

Text Segmentation

- As an extension of specifying language, text segmentation diver se greatly according to the language.
- Languages with inter-word spaces or similar:
  - Divide words with ‘spaces’, punctuation marks, stop words & etc.

  Ex) A piece of cake. -> [a][piece][of][cake][.]
- Languages written in Scriptio continua:
  - Usually use external lexicons which can be constituted with common w ord, synonym word, stop word etc.
  - Also may use spelling corrector to determine where the ‘spaces’ should be placed.

Handling Aberrations

- Aberrations are things that are unique or symbolic
  - Ex) N.Y. => New York,
  - Abbreviations and symbols are unique tokens considering they use periods or completely different characters apart from the normal alphabet of the accordin g language.

- Identifying <u>non-standard words</u> including numbers, abbreviations, and dates, and mapping any such tokens to a special vocabulary.

Handling Aberrations

- Methods vary to enlisting them beforehand or making countermeasures su ch as…
  - Lexical approaches; substituting non-frequent words to common ones
  - Semantic approaches; providing definitions for the unusual tokens
  - Multilingual approaches; using other languages to normalize the unusu al tokens by linking several multilingual definitions to it.

- One must be cautious to…
  - Make sure that the resulting token or substitute is a known word in a di ctionary
  - To properly distinguish the needed punctuation marks and unneeded o nes when removing them.

Basic Steps in Tokenization

Selecting Tokens : Normalization

Once we have all the tokens it is now the time to select tokens to form a vocabulary one could use. But before that it is essential to <u>Normalize</u>

<u>.</u>

- Normalization is done to index the tokens properly as well as allowing the t okens to represent words properly.
  - Lemmatization is used to reduce various form to base form

    Ex) (am, are, be, is) → [be] | (car, cars, car's, cars') → [car]
  - Stemming is also used to reduce terms to their "root" form

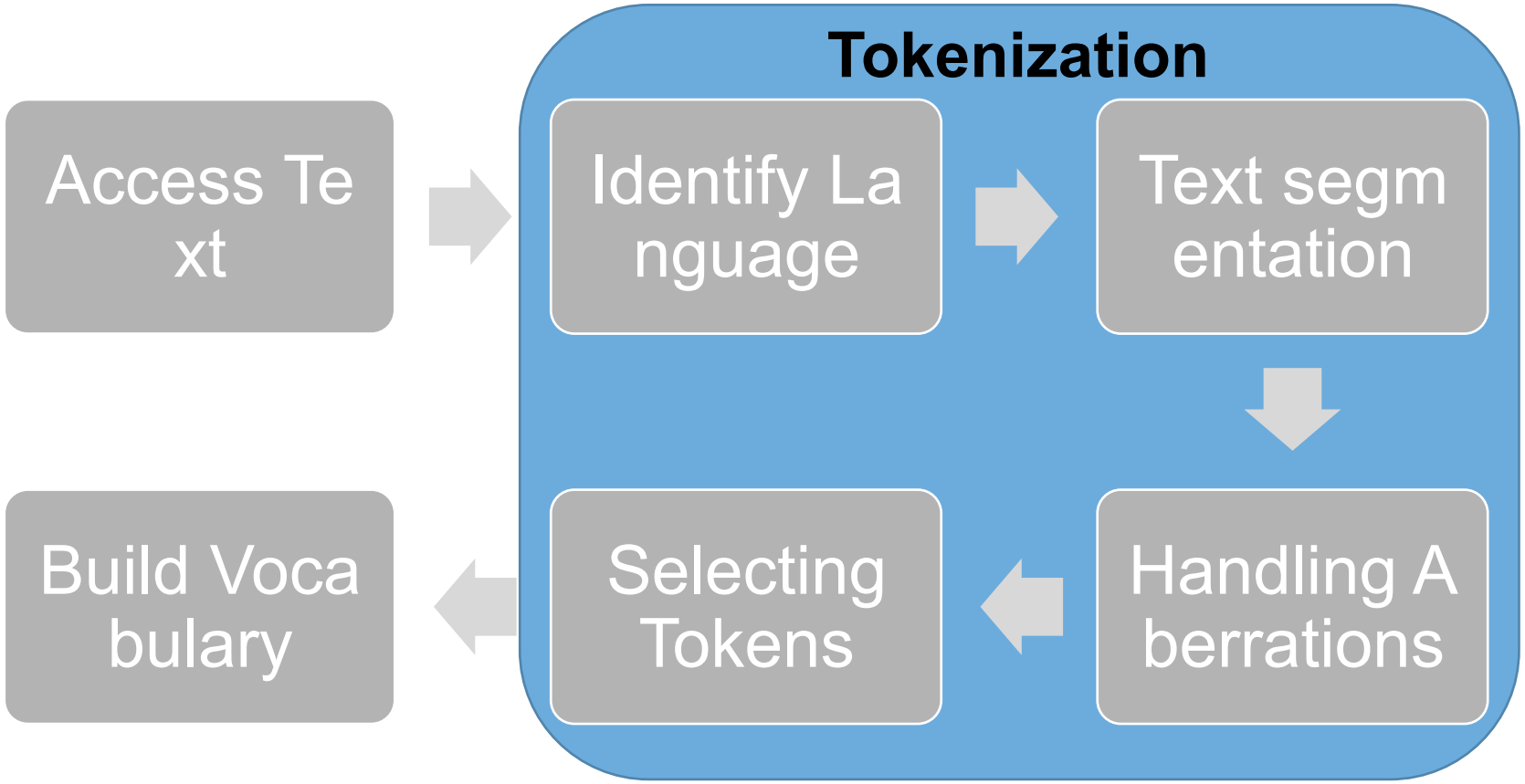
    Ex) (Compressed, Compression, Compressing …) → [Compress]

- In the case of English or other Alphabet based letters, it is common to conv ert texts into lowercases beforehand to reduce Aberrations as it reduces th e 52 characters into 26.

Building a Vocabulary

- Once the tokenization is done, it is produced as a list of strings(words).
  - It is essential that we create a Vocabulary so we can sort it.
- In our list of Tokens we will find a number of tokens recurring again and again. It is now time to get rid of the excess material.
  - How one get rids of the excess material and how he deals with the unique tokens left behinds will greatly affect the built vocabulary usefulness and purpose.
  - If the purpose is to train, it is important to mark the more recurring tokens as well as mark the more important tokens as well
  - For tasks that only need the tokens such as language identification, it is not need ed.

Conclusion : Our Journey so Far…



❖Note that text preprocessing differs greatly according to the pro grammer.

Extra: Spelling Correction

- Spelling Correction or spell checker is a program used to flag words in a document that may not be spelled correctly.

- There are 2 basic principles for most spelling correction algorithms
  - Of various alternative correct spellings for a misspelled query, choose the ‘near est one’.
  - When two correctly spelled queries are tied (or nearly tied), select the one that i s more common.

Forms of Spelling Correction

We focus on 2 specific forms of spelling correction

- Isolated-Term Correction
  - Attempt to correct a single query term at a time.

- Context-Sensitive Spelling Correction
  - An algorithm that are capable of recognizing a misspelled word based on the c ontext of the surrounding words.

Isolated-Term Correction

Isolated-Term Correction usually consists of these steps

1. Get misspelled error string.
2. Extract words from vocabulary ‘near’ to the misspelled error string.
3. Find the nearest word for spelling correction.

While step 2 differs greatly among programmers,

edit distance is widely used for step 3

Extra: Spelling Correction

Isolated-Term Correction: Edit Distance

Through edit distance, once the error is detected, it is used to change the error to a word ‘nearest’ to it.

→ Given two character strings S1 and S2, the **<u>edit distance</u>** is the minimum number of edit operations required to transform S1 into S2.

- The usual edit operations are…
  1) Insert a character into a string
  2) Delete a character from a string
  3) Replace a character of a string by another character

Extra: Spelling Correction

Isolated-Term Correction: Edit Distance

The following is a programming algorithm for computing the edit distance between strings S 1 & S2.

```
EDITDISTANCE(s1, s2)
int m[i, j] = 0
for i ← 1 to |s1|
do m[i, 0] = i
for j ← 1 to |s2|
do m[0, j] = j
for i ← 1 to |s1|
do for j ← 1 to |s2|
    do m[i, j] = min{m[i − 1, j − 1] + if (s1[i] = s2[j]) then 0 else 1fi,
                     m[i − 1, j] + 1,
                     m[i, j − 1] + 1}
return m[|s1|, |s2|]
```

※ Note that simply comparing the error string with every other word is very expensive, theref ore in spelling correction additional algorithms should be added to compensate for that.

Homework: Context-Sensitive Spelling Correction

- As said above, there are various methods for Context-Sensitive Spelling Correction. Find & introduce one.
  - The report should be more or less 1 page long.
  - Font size should be 11, File type should be ‘doc’.
  - It should mainly be about its name, history, how it works, pros and cons , and current status of the method.
  - Leave references.
  - Paper should be in English.
- Your paper shall be graded according to
  - How well you have presented your topic on your report.
  - How accurate your paper was.
  - Whether you made mistakes(misspellings, type-o’s etc.)