Natural Language Processing

Unit 1: Preprocessing

August 2019

What is Preprocessing?

Preprocessing

Text preprocessing is a natural language task which process input text in order to make it predictable and analyzable for a natural language system performing one or many tasks.

Why is important?

- Text is often unestructured and is hard to process for a computational system

Why is important?

- Text is often unestructured and is hard to process for a computational system
- Natural text produced by humans has ortographic errors

Why is important?

- Text is often unestructured and is hard to process for a computational system
- Natural text produced by humans has ortographic errors
- Many words are not useful for some sets of specific NLP tasks

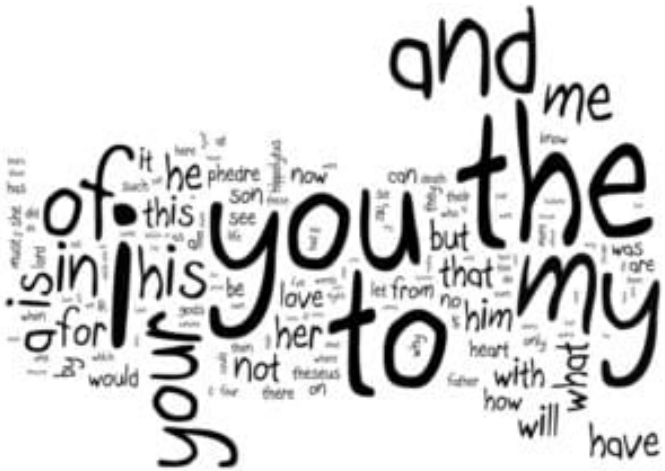
Why is important?

- Text is often unestructured and is hard to process for a computational system
- Natural text produced by humans has ortographic errors
- Many words are not useful for some sets of specific NLP tasks
- Not every verb conjugation and adjective form is useful for every task

Stop words removing

Stop words removal

Stop words are a set of commonly used words in a language. Examples of stop words in English are “a”, “the”, “is”, “are” and etc. The intuition behind using stop words is that, by removing low information words from text, we can focus on the important words instead.



Retrieved from https://webstrategies.com/stop-words-everything-you-should-know/

Stemming

Stemming

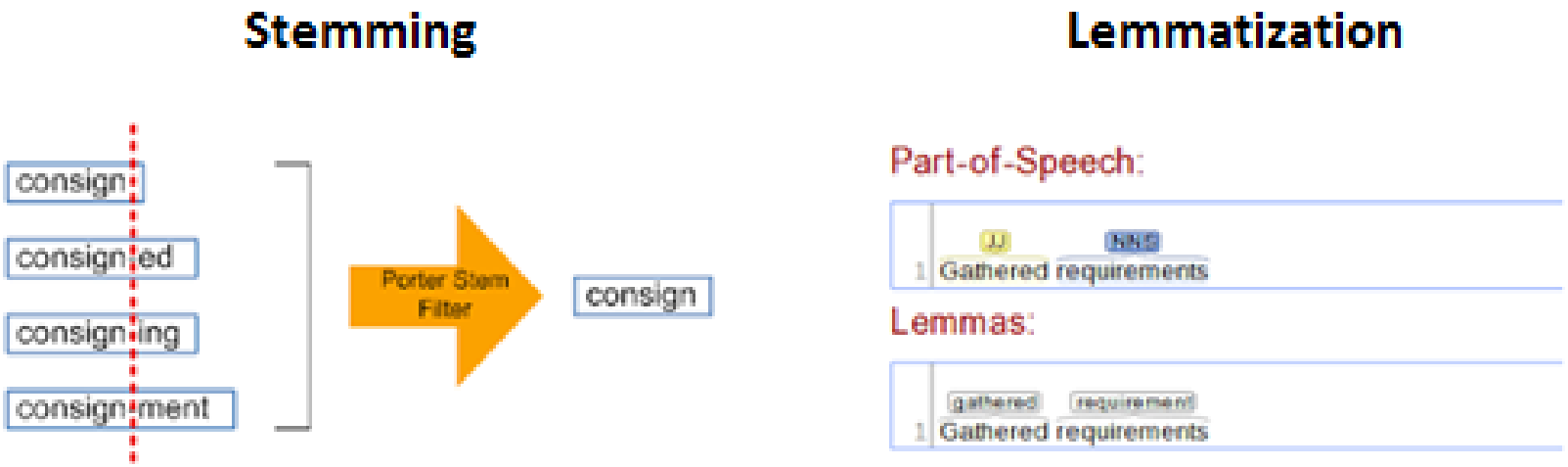
Stemming is the process of reducing inflection in words (e.g. troubled, troubles) to their root form (e.g. trouble). The "root" in this case may not be a real root word, but just a canonical form of the original word. Stemming uses a crude heuristic process that chops off the ends of words in the hope of correctly transforming words into its root form. So the words "trouble", "troubled" and "troubles" might actually be converted to troubl instead of trouble because the ends were just chopped off (ughh, how crude!). There are different algorithms for stemming. The most common algorithm, which is also known to be empirically effective for English, is Porters Algorithm.

Lemmatization

Lemmatization

Lemmatization on the surface is very similar to stemming, where the goal is to remove inflections and map a word to its root form. The only difference is that, lemmatization tries to do it the proper way. It doesn't just chop things off, it actually transforms words to the actual root. For example, the word "better" would map to "good". It may use a dictionary such as WordNet for mappings or some special rule-based approaches.

Difference between stemming and lematizing



Retrieved from https://www.quora.com/What-is-difference-between-stemming-and-lemmatization

Case normalization

Case normalization

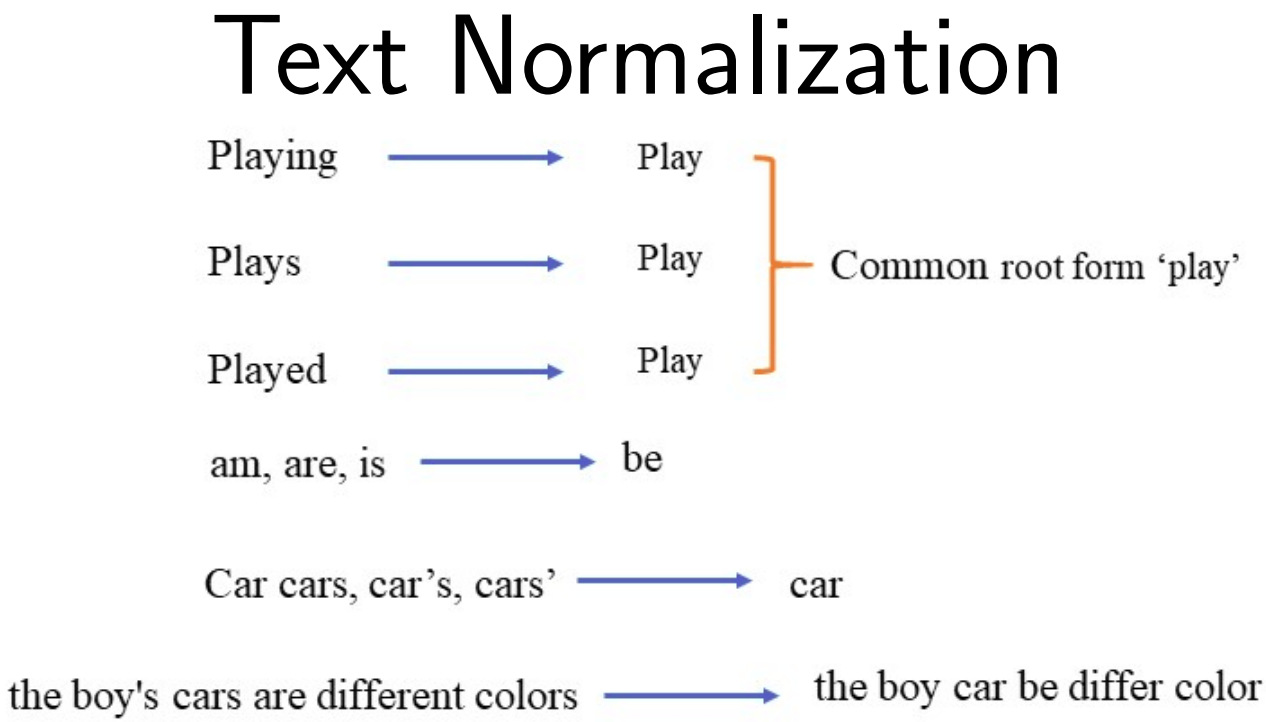
Normalizing ALL your text data, although commonly overlooked, is one of the simplest and most effective form of text preprocessing. It is applicable to most text mining and NLP problems and can help in cases where your dataset is not very large and significantly helps with consistency of expected output. Most of the time all data is represented as lower case text.

Text normalization

Text normalization

A highly overlooked preprocessing step is text normalization. Text normalization is the process of transforming text into a canonical (standard) form. For example, the word “gooood” and “gud” can be transformed to “good”, its canonical form. Another example is mapping of near identical words such as “stopwords”, “stop-words” and “stop words” to just “stopwords”. Text normalization is important for noisy texts such as social media comments, text messages and comments to blog posts where abbreviations, misspellings and use of out-of-vocabulary words (oov) are prevalent.

Text normalization

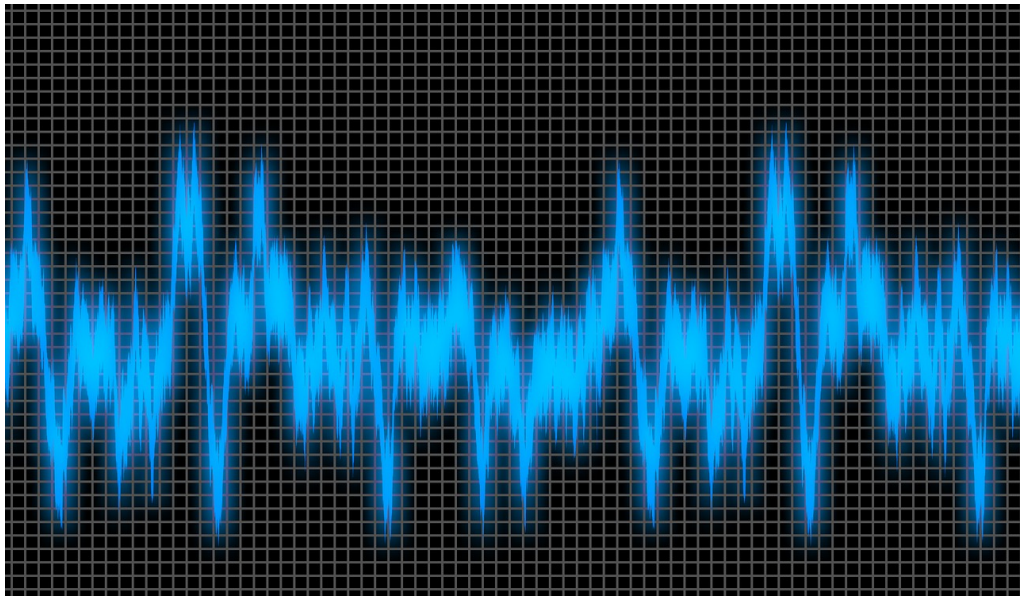


Retrieved from https://www.datacamp.com/community/tutorials/stemming-lemmatization-python

Noise removal

Noise removal

Noise removal is about removing characters digits and pieces of text that can interfere with your text analysis. Noise removal is one of the most essential text preprocessing steps. It is also highly domain dependent.



Text enrichment

Text enrichment

Text enrichment involves augmenting your original text data with information that you did not previously have. Text enrichment provides more semantics to your original text, thereby improving its predictive power and the depth of analysis you can perform on your data.

Which tasks you should do?

- Must Do:
  - Noise removal
  - Case normalization
- Should Do:
  - Simple normalization
- Task dependant:
  - Advanced normalization (e.g. addressing out-of-vocabulary words)
  - Stop-word removal
  - Stemming / lemmatization
  - Text enrichment / augmentation

SOLDAI RESEARCH

Tokenization

Tokenization

Tokenization is the task to extract functional units or tokens from an information string in order to perform sintactic and lexical based tasks. The process of segmenting running text into words and sentences. Electronic text is a linear sequence of symbols (characters or words or phrases). Naturally, before any real text processing is to be done, text needs to be segmented into linguistic units such as words, punctuation, numbers, alpha-numerics, etc. This process is called tokenization.

SOLDAI RESEARCH

Tokenization

English and spanish words are often separated from each other by whitespace, but whitespace is not always sufficient:

- New York is treated as a large word despite it contain spaces
- We need to separate I'm into the words I and am
- To process tweets we need to tokenize emoticons like :)
- Chinese don't have white spaces. How we tokenize? (jieba https://github.com/fxsjy/jieba)

SOLDAI RESEARCH

Regular expressions

Regular expressions are expressions written in an special format in order to find specific patterns in text corpus:

- Emails (mario.campos@upy.edu.mx)
- Numbers (1245.32)
- Company names (SoldAI SAPI S.A de C.V)
- Phone numbers (+529994263828)

Regular expressions

Simple regular expressions

RE	Example Patterns Matched
/woodchucks/	"interesting links to woodchucks and lemurs"
/a/	"Mary Ann stopped by Mona's"
/!/	"You've left the burglar behind again!" said Nori

RE	Match	Example Patterns
/[wW]oodchuck/	Woodchuck or woodchuck	"Woodchuck"
/[abc]/	'a', 'b', *or* 'c'	"In uomini, in soldati"
/[1234567890]/	any digit	"plenty of 7 to 5"

RE	Match	Example Patterns Matched
/[A-Z]/	an upper case letter	"we should call it 'Drenched Blossoms' "
/[a-z]/	a lower case letter	"my beans were impatient to be hoed!"
/[0-9]/	a single digit	"Chapter 1: Down the Rabbit Hole"

Regular expressions

Aliases

RE	Expansion	Match	First Matches
\d	[0-9]	any digit	Party␣of␣5
\D	[^0-9]	any non-digit	Blue␣moon
\w	[a-zA-Z0-9_]	any alphanumeric/underscore	Daiyu
\W	[^\w]	a non-alphanumeric	!!!!
\s	[␣\r\t\n\f]	whitespace (space, tab)	
\S	[^\s]	Non-whitespace	in␣Concord

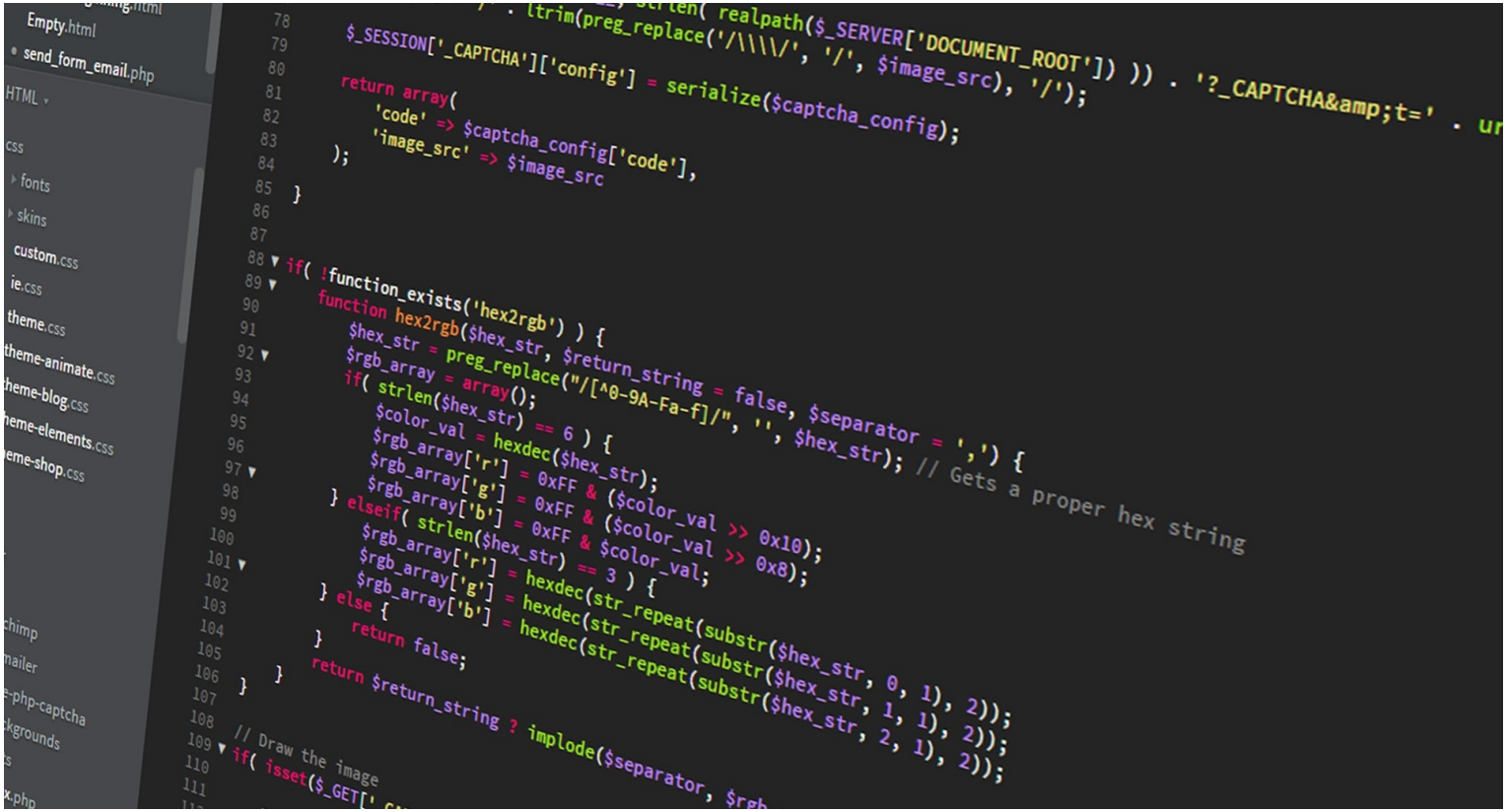
Regular expressions

Count operators

RE	Match
*	zero or more occurrences of the previous char or expression
+	one or more occurrences of the previous char or expression
?	exactly zero or one occurrence of the previous char or expression
{n}	*n* occurrences of the previous char or expression
{n,m}	from *n* to *m* occurrences of the previous char or expression
{n,}	at least *n* occurrences of the previous char or expression
{,m}	up to *m* occurrences of the previous char or expression

SOLDAI RESEARCH

Let's code


Empty.html
send_form_email.php
HTML
css
fonts
skins
custom.css
ie.css
theme.css
theme-animate.css
theme-blog.css
theme-elements.css
theme-shop.css
chimp
mailer
-php-captcha
kgrounds
x.php
$_SESSION['_CAPTCHA']['config'] = serialize($captcha_config);
return array(
'code' => $captcha_config['code'],
'image_src' => $image_src
);
}
if(!function_exists('hex2rgb')) {
function hex2rgb($hex_str, $return_string = false, $separator = ',') {
$hex_str = preg_replace("/[^0-9A-Fa-f]/", '', $hex_str); // Gets a proper hex string
$rgb_array = array();
if(strlen($hex_str) == 6) {
$color_val = hexdec($hex_str);
$rgb_array['r'] = 0xFF & ($color_val >> 0x10);
$rgb_array['g'] = 0xFF & ($color_val >> 0x8);
$rgb_array['b'] = 0xFF & $color_val;
} elseif(strlen($hex_str) == 3) {
$rgb_array['r'] = hexdec(str_repeat(substr($hex_str, 0, 1), 2));
$rgb_array['g'] = hexdec(str_repeat(substr($hex_str, 1, 1), 2));
$rgb_array['b'] = hexdec(str_repeat(substr($hex_str, 2, 1), 2));
} else {
return false;
}
return $return_string ? implode($separator, $rgb
// Draw the image
if(isset($_GET[

Assignments

Assignment 2: Write a jupyter notebook to perform the following NLP tasks defined as functions:

- Stemming
- Lemmatizing
- Cleansing
- Word Tokenize
- Phrase Tokenize
- Text normalization
- Regex entity extractor

Make a jupyter notebook explaining the code and loading the tweets database in order to pre process the text extracting hashtags and twitter user names as tokens

SOLDAI RESEARCH

References

[1] Jurafsky D., Martin J.: Speech and Language Processing 2nd. ed. (2009).

[2] Bird S., Klein E., Loper, E.: Natural Language Processing with Python, (2009). ISBN: 978-0-596-51649-9

[3] Indurkhya N., Damerau F. Handbook of Natural Language Processing, Second Edition (2010). ISBN: 978-1-4200-8593-8

[4] Kao, A., Poteet S. Natural Language Processing and Text Mining (2007). ISBN: 78-1-84628-175-4

[5] https://kavita-ganesan.com/text-preprocessing-tutorial

SOLDAI RESEARCH