## Contents

## 1 Lesson Outline

- The INTs (HUMINT, ELINT, GEOINT, OSINT, etc.)
- Text Analytics - the process of analyzing unstructured text, extracting relevant information, and transforming it into useful intelligence.
- Text to be analyzed: a list of songs by the Beatles
  - How many different words did they use?
  - Which words were used frequently?
  - Which word was used the most?
- Morse Code example
- Linux stuff: >, <, &, |, wget, tr, vi, wc, sed, cat, less, more, sort, uniq

## 2 Step-by-step:

1. Make a directory to work in and move into it
```
mkdir songs
cd songs
```
2. Get a list of Beatles songs from the Internet and look at it several different ways
```
wget https://raw.githubusercontent.com/matt-jacobs/modules/master/text-analytics/lesson-01/beatles-songs.txt
cat beatles-songs.txt # cat is short for catenate
less beatles-songs.txt # less shows the file a page at a time
more beatles-songs.txt # is a newer version of less
vi beatles-songs.txt  # vi is a text editor.  ESC!q exits without saving changes
nano beatles-songs.txt & nano beatles-songs.txt # this time without the ampersand
```
3. How many songs did the Beatles write?
```
wc -l < beatles-songs.txt # wc counts the number of lines in a file
wc -c < beatles-songs.txt # wc counts the number of characters in a file
wc -w < beatles-songs.txt # wc counts the number of "words" in a file
```
4. Use the **man** command to see how a **word** is defined
```
man wc
```
5. To get each word on its own line, change spaces to newlines
```
tr ' ' '\n' < beatles-songs.txt
tr ' ' '\n' < beatles-songs.txt | more
tr ' ' '\n' < beatles-songs.txt | sort | more
tr ' ' '\n' < beatles-songs.txt | sort | uniq | more
```
6. To get rid of parentheses, change them to spaces BEFORE changing the spaces to new lines
```
tr '(' ' ' < beatles-songs.txt |  tr ' ' '\n'|  sort | uniq | more
tr '(' ' ' < beatles-songs.txt |  tr ')' ' ' | tr ' ' '\n'|  sort | uniq | more
sed -e 's?[()]? ?g' < beatles-songs.txt  | tr ' ' '\n'|  sort | uniq | more
```

7. To get rid of other characters, also change them to spaces BEFORE changing the spaces to new lines
```
sed -e 's?[(),/!.?]? ?g' < beatles-songs.txt | tr ' ' '\n' | sort | uniq | more
```
8. Are uppercase and lowercase versions of words different?
```
sed -e 's?[(),/!.?]? ?g' < beatles-songs.txt | tr ' ' '\n' | tr [:upper:] [:lower:] | sed '/^\s*$/d' | sort
| uniq | more
```
9. Get rid of blank lines
```
sed -e 's?[(),/!.?]? ?g' < beatles-songs.txt | tr ' ' '\n' | tr [:upper:] [:lower:] | sed '/^\s*$/d' | sort
| uniq | more
```

10. Now we can count words
```
sed -e 's?[(),/!.?]? ?g' < beatles-songs.txt | tr ' ' '\n' | tr [:upper:] [:lower:] | sed '/^\s*$/d' | sort
| uniq -c | more
sed -e 's?[(),/!.?]? ?g' < beatles-songs.txt | tr ' ' '\n' | tr [:upper:] [:lower:] | sed '/^\s*$/d' | sort
| uniq -c |sort -bnr > counts.txt
more counts.txt
```
11. Test some words to see if the counts are correct
```
grep -i goodbye beatles-songs.txt
```