BRAND ANALYSIS USING NLTK

RVA DATA HACKERS

ROBERT LEE

6/17/14

TABLE OF CONTENTS

WHAT IS PYTHON

- Python is a programming language that is…
  - High level and human readable
  - Interpreted, not compiled
  - Object-oriented yet very functional
  - Well suited to text analysis and NLP

WHAT IS NLTK

- The **N**atural **L**anguage **T**ool**k**it (NLTK)
  - Contains a large library of modules designed for natural language processing
  - Modules range from text processing to advanced classification algorithms for text analysis
  - Modules included that we won't cover:
    - Semantic analysis: tagging, chunking, grammars
    - Machine Learning for classification
    - Information extraction: relational, structured data

GOALS OF THE WORKSHOP

- In the next forty minutes, you will be able to:
  - Read, write, and understand basic python syntax
  - Run an interactive Python session from the CLI
  - Read in raw text and manipulate it in Python
  - Use many of NLTK's basic python functions
  - Understand tokenization and token frequency
  - Utilize this knowledge to process employee survey responses to analyze internal branding

TWO WAYS TO USE PYTHON

- In the Python Interpreter
  - Type each line at the command prompt (>>>)
  - Python interprets each line as it's entered
- Running a standalone script
  - Write up your program in a plain-text file
  - Save it with the extension .py
  - Execute it on the command line: python script.py
- Speed and convenience versus complexity

‘HUMAN-READABLE’ PYTHON CODE

Our Python program

```
1 >>> for line in open("file.txt"):
2 >>>     for word in line.split():
3 >>>         if word.endswith("ing"):
4 >>>             print word
5 ...
```

- How to read this Python program
  - For each line in the text file *file.txt*
  - For each word in the line (split into a list of words)
  - If the word ends with *–ing*
  - Print the word

DEFINING PYTHON FUNCTIONS

Functions

A function is a way of packaging and reusing program code.

```
1 >>> def repeat(message):
2 ...     return message + message
3 >>> monty = "Monty Python"
4 >>> repeat(monty)
5 "Monty Python Monty Python"
```

- Line 1 defines a function named repeat that takes in one argument named ***message***
- Note that types are not declared for variables
- We define a variable ***monty*** and then call the function with ***monty*** as its argument

PYTHON DATA TYPES

A number stores a numeric value.

```
>>> variable_1 = 10
```

A string is a continuous sequence of characters in quotation marks.

```
>>> variable_2 = "John Smith" # or 'John Smith'
```

A list contains items separated by commas and enclosed in square brackets. Items can be of different data types, and lists can be modified.

```
>>> variable_3 = [10, "John Smith", ["another", "list"]]
```

A tuple is like a list, but it is immutable (i.e., tuples are read-only).

```
>>> variable_4 = (10, "John Smith")
```

A dictionary contains key-value pairs, and is enclosed in curly braces.

```
>>> variable_5 = {"name": "John Smith", "department": "marketing"}
```

TABLE OF CONTENTS

WHAT IS A TOKEN

- A ***token*** is a sequence of characters that are treated as a group
- Usually synonymous with “word” of sentence
- Some examples of tokens:
  - brand
  - Robert
  - hasn’t
  - state-of-the-art
  - ,

PROCESSING RAW TEXT

- Processing raw text data takes several steps
  - Reading in the data into a raw string
  - Splitting raw string into distinct tokens
  - Removing meaningless tokens and punctuation
  - Normalizing text to be case-insensitive
  - Correcting tokens for spelling mistakes (optional)
- Processing pipeline allows for standardization of text for analysis

USING THE NLTK TOKENIZER

The NLTK word tokenizer

```
1 >>> tokens = nltk.word_tokenize(raw)
```

`word_tokenize()` is the NLTK's default tokenizer function. It's possible to write your own, but `word_tokenize()` is usually appropriate in most situations.

- What does this tokenizer treat as the boundary between tokens?

Querying the type of tokens

```
1 >>> type(tokens)
2 <type "list">
```

`tokens` is of the `type` *list*.

STANDARDIZATION OF TOKENS

Defining an NLTK text

```
>>> text = nltk.Text(tokens)
```

Calling the `nltk.Text()` module on `tokens` defines an NLTK Text, which allows us to call more sophisticated text analysis methods on it.

Querying the type of text

```
>>> type(text)
<class "nltk.text.Text">
```

The `type` of `text` is not a list, but a custom object defined in the NLTK.

REMOVING STOP WORDS

Finally, we could remove the stop words

- Stop words are words like *the*, *to*, *by*, and *also* that have little semantic content
- We usually want to remove these words from a text before further processing
- Stop words are highly frequent in most texts, so their presence doesn't tell us much about this text specifically

The NLTK includes lists of stop words for several languages

```
1 >>> from nltk.corpus import stopwords
2 >>> stopwords = stopwords.words("english")
```

- Consider removing words of the organization

CORRECTING FOR SPELLING

- Python contains a module called Enchant
- Enchant can spell check for multiple languages
- Suggestions are ordered by likelihood

```
Python 2.7.6 (default, Mar 26 2014, 15:27:46)
[GCC 4.2.1 Compatible Apple LLVM 5.1 (clang-503.0.38)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import enchant
>>> dictionary = enchant.Dict("en_US")
>>> dictionary.check("spelling")
True
>>> dictionary.check("speling")
False
>>> dictionary.suggest("speling")
['spelling', 'spieling', 'sapling', 'spilling', 'spoiling', 'spooling', 'spline', 'spewing', 'spellings',
'pealing', 'peeling', 'sealing', 'selling', 'soling', 'spleen', 'sling', 'speckling', 'paling', 'piling',
'poling', 'puling', 'splint', "spelling's"]
```

TABLE OF CONTENTS

MOTIVATION OF WORKSHOP

- How can we leverage NLTK to extract meaning
- Discover how text, when analyzed in quantity, can provide useful and actionable information
- Given a list of employee survey responses about our company (*Cyberdyne)*, can we discover an internal branding?
- Generalize the practical exercise to allow similar analysis to be done in different fields

TABLE OF CONTENTS

FREQUENCY DISTRIBUTIONS

- FreqDist creates a dictionary in which keys are tokens occurring in the text and the values are the corresponding frequencies
- Can assist in quickly discovering words associated with the organization
- Provides an easy, big picture look at branding

TABLE OF CONTENTS

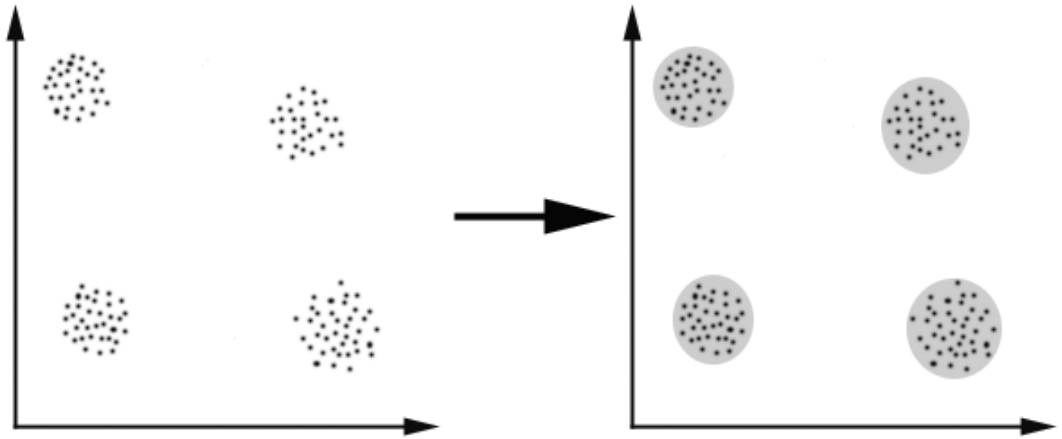
PURPOSE OF STEMMING WORDS

- Stemming reduces words to their stem (morphological root)
- Allows for the grouping of common words
  - Cats, catlike, catty all are reduced to stem cat
  - Fishing, fisher, fished all are reduced to stem fish
- Reduces the total set of words to allow for quicker and blander analysis

TABLE OF CONTENTS

WHAT IS CLUSTERING

- Clustering is an unsupervised learning algorithm that attempts to find structure in a collection of unlabeled data
- We want to “cluster” common sentences which might give rise to a common theme



HOW TO CLUSTER TEXT

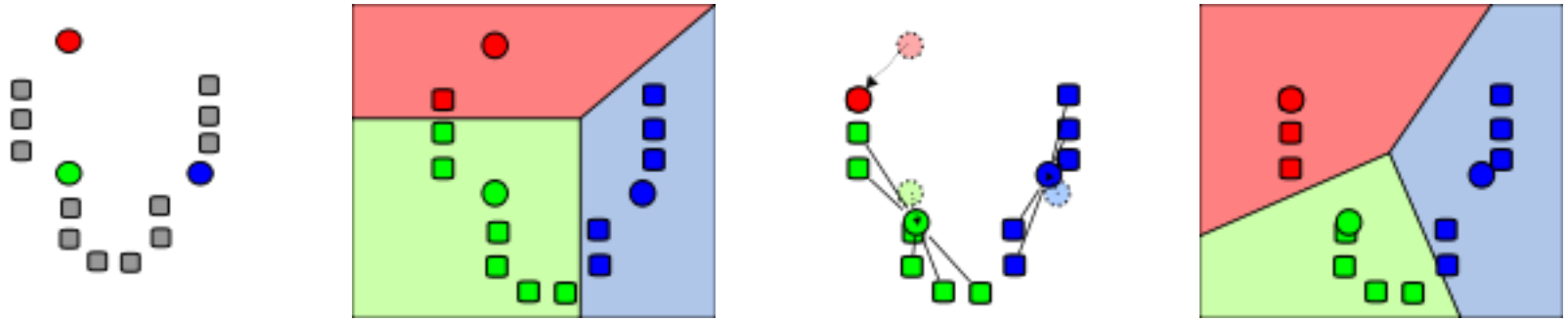
- To categorize sentences, we must build a quantitative metric to compare like sentences
- Sentences are composed of words (or stems)
- Sentences that share meaningful words (or stems) tend to share a common meaning
- Build a "**feature vector**" that is the length of all the words (or stems) mentioned in survey
- Compute the vector for each response and cluster the responses into common themes

FEATURE VECTORS CONTINUED

- Consider the fake survey of two responses:
  - “I like ice cream”
  - “I hate ice cream”
- Word set: [I, like, ice, cream, hate]
- The feature vectors for each sentence:
  - “I like ice cream” [1, 1, 1, 1, 0]
  - “I hate ice cream” [1, 0, 1, 1, 1]

K-MEANS CLUSTERING

- Given the vectors for each response, find “K” clusters that minimizes the vector distance for all the responses in the survey.



- Shown: iterative process of assigning vectors (responses) into three clusters (or themes).

TABLE OF CONTENTS

FURTHER EXPLORATION

- Looking at word associations via word frequency matrix for common groupings
- Discover word groupings that contain many links to other groupings
- Check out supervised learning algorithms (Naïve-Bayes, Maximum Entropy, etc) to better classify a feature set

RESOURCES

- Bird, Steven, Ewan Klein, and Edward Loper. 2009. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. Sebastopol, CA: O'Reilly. (Available online at http://www.nltk.org/book/)
- Perkins, Jacob. 2010. Python Text Processing with NLTK 2.0 Cookbook. Birmingham: Packt.