Text Analytics

Introduction

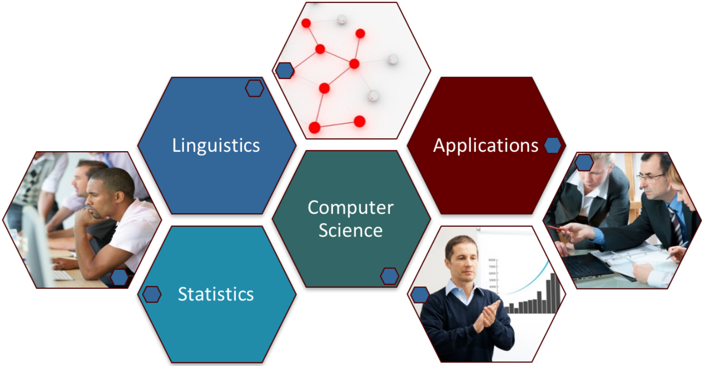
Edward Jones

Dept. of Statistics
Texas A&M University

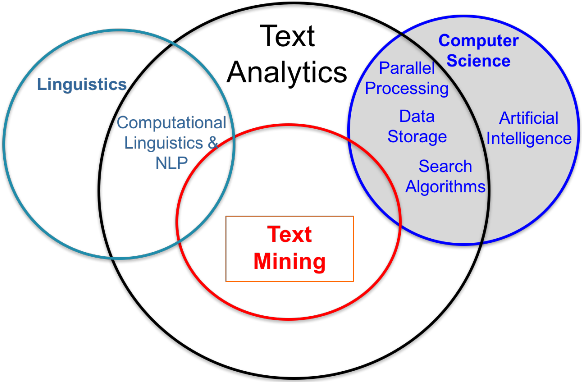
July 9, 2020

Outline

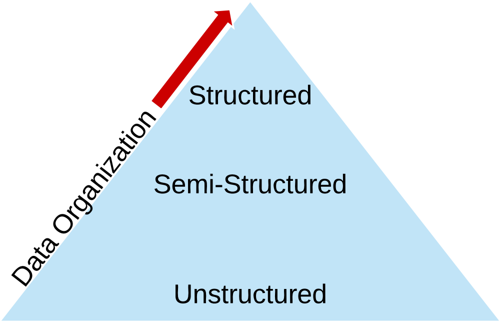
Text Analytics



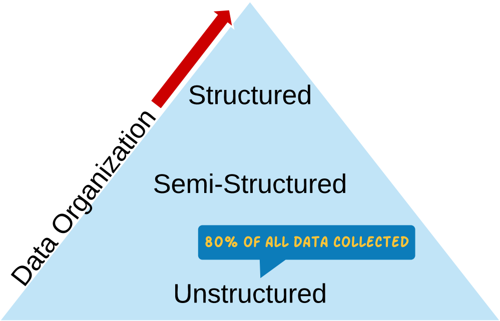
Text Analytics



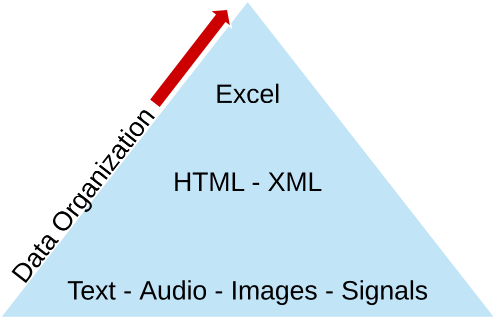
Data Structure



Unstructured Data



Data Structure Examples



Text Data

Text data are referred to as *Natural Language* (NL) communications.

NL data consist of all written records such as:

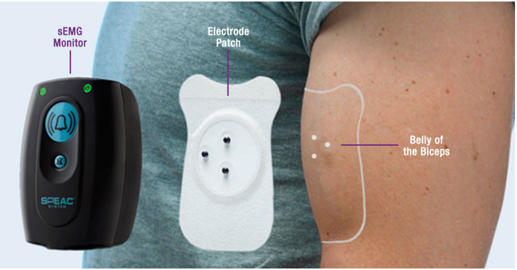
- Books
- Email & Transcripts
- Reviews & Opinions
- Customer Complaints
- Maintenance Reports
- Accident Reports

Other Unstructured Data

Other Unstructured Data Include:

- Audio Recording
- Photographs
- Video Recordings
- Signal Processing

Signal Processing Example: Seizure Monitor



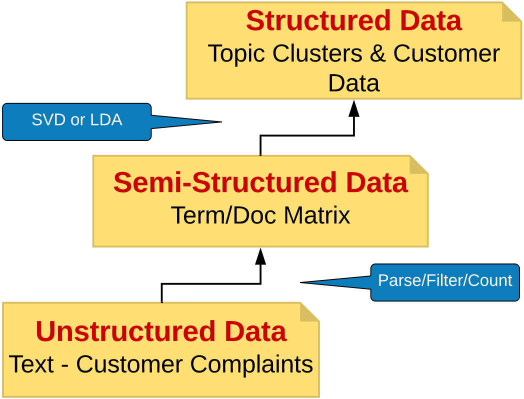
Signal Processing Example

Structured Data
Signal Moving Avg/Sec

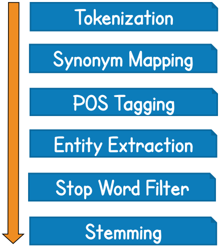
Semi-Structured Data
Wavelet Transformation

Unstructured Data
Signal Every 1 ms

Text Processing Example



Preparing Term-Doc Matrix



$$T^{t \times d} = \begin{pmatrix} f_{1,1} & f_{1,2} & f_{1,3} & \cdots & f_{1,d} \\ f_{2,1} & f_{2,2} & f_{2,3} & \cdots & f_{2,d} \\ f_{3,1} & f_{3,2} & f_{3,3} & \cdots & f_{3,d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{t,1} & f_{t,2} & f_{t,3} & \cdots & f_{t,d} \end{pmatrix}$$

Parse

Parsing is the most time consuming step in Text Analytics. Its objective is to create a list of terms and term groups in the corpus.

1. **Tokenization:** Extracting individual words, numbers and punctuation
2. **Synonym Mapping:** Replacing words with their synonym
3. **POS Tagging:** (optional) Assigning Parts of Speech to Terms
4. **Entity Extraction:** (optional) Identify Noun Groups & Entities

Filter

Parsing is followed by filtering. The purpose of filtering is to remove unwanted words and to stem words to their root form..

6. **Stop Word Filter:** Remove stop words
7. **Filter Other Words:** (optional) Remove words found in most documents and words found in only a few.
8. **Stemming:** (optional) Map words to their root form

Frequency Form of Term-Document Matrix

$$T^{t \times d} = \begin{pmatrix} f_{1,1} & f_{1,2} & f_{1,3} & \cdots & f_{1,d} \\ f_{2,1} & f_{2,2} & f_{2,3} & \cdots & f_{2,d} \\ f_{3,1} & f_{3,2} & f_{3,3} & \cdots & f_{3,d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{t,1} & f_{t,2} & f_{t,3} & \cdots & f_{t,d} \end{pmatrix}$$

where:
t is the number of terms in the corpus (document collection)
d is the number of documents in the corpus
$f_{i,j}$ is the number of times the ith term appears in the jth document.

Example Term-Document Matrix

$$T^{5\times 6} = \begin{pmatrix} 0 & 0 & 0 & 0 & 4 & 0 \\ 5 & 2 & 8 & 4 & 9 & 0 \\ 2 & 0 & 0 & 0 & 5 & 0 \\ 0 & 4 & 3 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

What does this tell us about topics in this corpus?

- Documents d_1 and d_5 share terms t_3 and t_5.
- Documents d_2 and d_3 share term t_4.
- Term t_1 appears unique to d_5.
- Term t_2 is common to most documents.
- Document d_6 is blank, or only contains filtered terms.

Binary Form of Term-Document Matrix

$$T^{t\times d} = \begin{pmatrix} \delta_{1,1} & \delta_{1,2} & \delta_{1,3} & \cdots & \delta_{1,d} \\ \delta_{2,1} & \delta_{2,2} & \delta_{2,3} & \cdots & \delta_{2,d} \\ \delta_{3,1} & \delta_{3,2} & \delta_{3,3} & \cdots & \delta_{3,d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \delta_{t,1} & \delta_{t,2} & \delta_{t,3} & \cdots & \delta_{t,d} \end{pmatrix} \text{ where } \delta_{i,j} = \begin{cases} 0 & \text{for} \quad f_{i,j} = 0 \\ 1 & \text{for} \quad f_{i,j} > 0 \end{cases}$$

Example Binary Term-Document Matrix

Frequency & Binary Forms:

$$T^{5\times 6} = \begin{pmatrix} 0 & 0 & 0 & 0 & 4 & 0 \\ 5 & 2 & 8 & 4 & 9 & 0 \\ 2 & 0 & 0 & 0 & 5 & 0 \\ 0 & 4 & 3 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \overset{\delta(f)}{\Rightarrow} \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Log-Frequency Form of Term-Document Matrix

$$T^{t \times d} = \begin{pmatrix} \lambda_{1,1} & \lambda_{1,2} & \lambda_{1,3} & \cdots & \lambda_{1,d} \\ \lambda_{2,1} & \lambda_{2,2} & \lambda_{2,3} & \cdots & \lambda_{2,d} \\ \lambda_{3,1} & \lambda_{3,2} & \lambda_{3,3} & \cdots & \lambda_{3,d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \lambda_{t,1} & \lambda_{t,2} & \lambda_{t,3} & \cdots & \lambda_{t,d} \end{pmatrix} \text{ where } \lambda_{i,j} = \ln(f_{i,j} + 1)$$

Example Log-Frequency Form

Frequency & Log Forms:

$$T^{5\times 6} = \begin{pmatrix} 0 & 0 & 0 & 0 & 4 & 0 \\ 5 & 2 & 8 & 4 & 9 & 0 \\ 2 & 0 & 0 & 0 & 5 & 0 \\ 0 & 4 & 3 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \overset{\ln(f+1)}{\Rightarrow} \begin{pmatrix} 0 & 0 & 0 & 0 & 1.61 & 0 \\ 1.79 & 1.10 & 2.20 & 1.61 & 2.30 & 0 \\ 1.10 & 0 & 0 & 0 & 1.79 & 0 \\ 0 & 1.61 & 1.39 & 0 & 0 & 0 \\ 0.69 & 0 & 0 & 0 & 0.69 & 0 \end{pmatrix}$$

Term-Document Table

Terms	d_1	d_2	d_3	$\cdots$	d_d
t_1	$f_{1,1}$	$f_{1,2}$	$f_{1,3}$	$\cdots$	$f_{1,d}$
t_2	$f_{2,1}$	$f_{2,2}$	$f_{2,3}$	$\cdots$	$f_{2,d}$
t_3	$f_{3,1}$	$f_{3,2}$	$f_{3,3}$	$\cdots$	$f_{3,d}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
t_n	$f_{n,1}$	$f_{n,2}$	$f_{3,3}$	$\cdots$	$f_{n,d}$

Create Term-Frequency Matrix

```
import sys, math, operator
import pandas as pd
import numpy  as np
from AdvancedAnalytics.Text import text_analysis, text_plot
# Packages from Sci-Learn
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.decomposition import TruncatedSVD
from sklearn.decomposition import NMF

import matplotlib.pyplot as plt
from wordcloud   import WordCloud, STOPWORDS
from collections import Counter
from PIL         import Image
```

```
ta = text_analysis(synonyms=None, stop_words=None, pos=True, stem=True )
cv = CountVectorizer(max_df=0.95, min_df=0.05, max_features=None,
                     binary=False, analyzer=ta.analyzer)
tf    = cv.fit_transform(reviews)
terms = cv.get_feature_names()
```

Apply TDIDF to Term-Frequency Matrix

```
# Show Word Cloud based on TFIDF weighting
tfidf = True
if tfidf == True:
    # Construct the TF/IDF matrix from the data
    print("\nConducting Term/Frequency Matrix using TF-IDF")
    # Default for norm is 'l2', use norm=None to supress
    tfidf_vect = TfidfTransformer(norm=None, use_idf=True)
    # tf matrix is (n_reviews)x(m_features)
    tf = tfidf_vect.fit_transform(tf)
```

Apply LDA to Term-Frequency Matrix

```
uv = LatentDirichletAllocation(n_components=n_topics)
U  = uv.fit_transform(tf)
text_analysis.display_topics(uv, terms, n_terms=15, word_cloud=True)
```

Display LDA Topics

```
# Turn the Term/Frequency matrix into a dictionary
td = text_plot.term_dic(tf, terms, scores=None)
# Display the top 20 terms
k  = Counter(td)
top_terms = k.most_common(20)
if type(top_terms[0][1]) == np.float64:
    for t in top_terms:
        print("{:10s}{:>8.2f}".format(t[0], t[1]))
else:
    for t in top_terms:
        print("{:10s}{:>8d}".format(t[0], t[1]))

text_plot.word_cloud_dic(td, mask=None, random=12345, bg_color="maroon",
                         max_words=30, size=(400,200))
```