Text Mining

- Text Mining
  - 1.1. Overview
    - 1.1.1. Definition of Text Mining
    - 1.1.2. Applications
    - 1.1.3. Advantages
      - 1.1.3.1. Language Recognition
      - 1.1.3.2. Spam Filtering
      - 1.1.3.3. Sentiment Analysis
      - 1.1.3.4. E-commerce Personalization
  - 1.2. Natural Language Tool Kit (NLTK) Library
    - 1.2.1. Overview
    - 1.2.2. Significance
    - 1.2.3. Installation
    - 1.2.4. NLTK Environment Setup
    - 1.2.5. Reading NLTK corpora
    - 1.2.6. Exploring Brown corpus
  - 1.3. Text Extraction and Pre-processing
    - 1.3.1. Concept of Tokenization
    - 1.3.2. Concept of Stemming
    - 1.3.3. Concept of Lemmatization
    - 1.3.4. Concept of Part-of-Speech (POS) Tagging
    - 1.3.5. Stop Word Removal
    - 1.3.6. Stemming
    - 1.3.7. Lemmatization
    - 1.3.8. Part-Of-Speech (POS) Tagging
    - 1.3.9. Named Entity Recognition (NER)
    - 1.3.10. Text Parser Grandfather
    - 1.3.11. Verbs. Broken Down
    - 1.3.12. Noun Phrases Requiring Broken Classes
  - 1.4. Concluding Remarks
    - 1.4.1. Demo Brown Corpus
    - 1.4.2. Phrase Structure Rules
    - 1.4.3. Demo Text Parser Output
    - 1.4.4. Remaining Work Items
    - 1.4.5. Setting Up Environment Variables
    - 1.4.6. Stanford Parser Output
  - 1.5. Concluding Remarks
    - 1.5.1. Demo Brown Corpus
    - 1.5.2. Chunk Parsing
    - 1.5.3. Chunking Output Format
    - 1.5.4. Chunking Using Brown
    - 1.5.5. IOB Chunk Just Noun
    - 1.5.6. Chunk Just Noun Output
  - 1.6. Concluding Remarks
    - 1.6.1. Chunking Using Brown Corpus
    - 1.6.2. Corpus Code Parser
    - 1.6.3. Chunking Output
  - 1.7. Concluding Remarks
    - 1.7.1. Implementing POS
  - 1.8. Concluding Remarks
    - 1.8.1. Phrase Structure Rules

Text Mining

Overview

> Text mining is the technique of structuring large amounts of unstructured text data and analyzing it in order to extract patterns from the raw data.

It uses software that can identify concepts, patterns, topics, keywords and so on to flag

It uses computational techniques to extract high-quality information from text.

Flow of Text Mining

Text mining utilizes various techniques to extract and connected high-quality information from unstructured textual resources.

There are 5 techniques used in Text Mining System.

Significance

Applications

Speech Recognition

- This is the recognition and translation of spoken language into text and vice versa.
- Information retrieval from audio files.
- Automatic speech recognition (ASR) is a multi-faceted discipline with close links to other multimedia content.
- Speech recognition combined with natural language processing (NLP) can be used to analyze phone conversations.

Spam Filtering

- Textual information contained in an email can be analyzed using text mining.
- The email service can then bring to check the email for spam.
- If someone wan to launch the email, it is handed on spam and send to the spam folder.

Sentiment Analysis

- It determines if a given sentence expresses positive, negative or neutral sentiment.
- This includes deep analysis of the body of text to understand the opinion, mindset, and mood.
- It works best on text that has subjective context.

E-commerce Personalization

- Text mining is used to suggest products based on the user's interests.
- Individual customer attributes learn more about consumers by analyzing information, such as their tone and publication.
- Retailers can use text mining to identify individuals with personalized offers and discounts.
- Text mining can be used for product classifying, purchase patterns and products or attributes.

Natural Language Tool Kit (NLTK) Library

Overview

> NLTK is a set of open source Python modules used to work with human language data for applying statistical natural language processing (NLP).

Significance

NLTK Environment Setup

Step 1: Open the command line terminal.

Step 2: Type the command line command.

Reading NLTK corpora

Loading all texts from NLTK book module

Exploring Brown corpus

Exploring Brown corpus

- Some categories will give the different genre available.
- Check genre and corpus in a new variable.

Text Extraction and Pre-processing

Concept of Tokenization

Tokenization is basically chopping a body of text into words and sentences.

Concerned with the step by by word, it is also known as.

It works by separating words using spaces and punctuations.

Text Extraction and Processing: N-grams

N-gram is a simple language model that assigns probabilities to sequences of words and sentences.

N-grams are combinations of adjacent words or letters of length n in the source text.

N-grams is very helpful in text mining, where it is required to extract patterns from the text.

It has several uses:

- Language Identification
- Keyword Extraction
- Sentiment Analysis
- Prediction of the next word or sentence
- Sentence Generation

See any of the sentences defined under NLTK book corpus in order to perform stemming as shown in the below example. All the words are converted to the corresponding stem or root word. Stemming is a kind of normalization for words.

Note: Stop words are language dependent.

Stemming

Stemming involves reducing a word to stem or base word form by removing Suffixes.

- Ex. Ideas, Ideally, Ideals, Ideas, Ideas, Ideas, Ideas, Ideas, Ideas, Ideas

The stemming process or algorithm is generally called as Stemmer

There are various stemming algorithms. Some of them are.

- Porter Stemmer
- Lancaster Stemmer
- Snowball Stemmer

See any of the sentences defined under NLTK book corpus in order to perform stemming as shown in the below example. All the words are converted to the corresponding stem or root word. Stemming is a kind of normalization for words.

Lemmatization

This is the method of grouping the various inflected forms of a word so that they can be analyzed as one item.

It uses a vocabulary list and morphological analysis (POS of a word) to get the root word.

- Lemmatization uses WordNet database and has English words listed together by their semantic relationships.

So we can see that the plural form of words, the different tenses and the different forms of words are converted to the root word.

rocks : rock	rocks : rock
corpora : corpus	corpora : corpus
better : good	better : good
Lemmatization without POS	Lemmatization with POS

Part-Of-Speech (POS) Tagging

A POS tagging tools assigns the most likely part-of-speech tag or a phrase tag based on its context and definition.

It is a part-of-text component such where in each word is given a category to indicate the part of speech and often also other grammatical categories such as tense, number (either plural or singular), case etc.

POS tags are used in text analysis tools and algorithms for training.

The great advantage is language independent whereas the same word in different part of speech, we will find an identification and retrieval of the word such that we can discuss the different POS tags. There are about 40 different tags in the Penn Treebank tagset.

POS tag is used for lemmatization in building Named Entity Recognition(NER) and extracting relationship between words.

Named Entity Recognition (NER)

NER helps to extract a set of relevant words from the text and it is now defined categories such as the names of persons, organizations, locations, etc.

Many use cases processing can be represented with the help of NER such as user get information, connections to companies or related. Does the lesser values the name of a person? Does the lesser consider the person's address? etc.

Text Mining Process
Text Mining Input
Information Retrieval
Natural Language Processing (NLP)
Information Extraction
Data Mining
Web Mining
Term Recognition
Named Entity Recognition (NER)
Co-reference resolution
Prior to the Named Entity Recognition (NER) task, we can get the tokens and the corresponding part-of-speech tags. Based on the part-of-speech tags, we can define a chunk parser to identify the named entities.
Text
Tokenization
Part-of-Speech Tagging
Chunking
Named Entity Recognition
Chunk Parsing
Chunk Parsing and Named Entity Recognition (NER)
Chunk Parsing
Named Entity Recognition (NER)
Chunk Parsing
Named Entity Recognition (NER)
Chunk Parsing

```
[illegible]
```


Приложение

```
[illegible]
```


```
[illegible]
```



```
[illegible]
```

Stop Word Removal

```
Stop 1. 1. 1.
```

```
Hence, convert the text into lower case, remove punctuations and numbers, tokenize the text, remove stopwords and stemming the text.

Use stemmer from nltk.stem.porter import PorterStemmer and create an instance of it.

Use stemmer.stem() to stem the text. Stemming should be done on words that are left after removing stopwords.
```

```
[nltk_data] Downloaded package stopwords to /root/nltk_data...
[nltk_data] Unzipped package corpora/stopwords.zip.
```

```
[nltk_data] Downloaded package punkt to /root/nltk_data...
[nltk_data] Unzipped package tokenizers/punkt.zip.
```

```
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()
```

```
[nltk_data] Downloaded package punkt_tokenizer to /root/nltk_data...
[nltk_data]   Unzipped package tokenizers/punkt_tokenizer.zip.
```

```
Lemmatization is an alternative approach to stemming. Use WordNetLemmatizer() from nltk.stem.wordnet import WordNetLemmatizer and create an instance of it.

Use lemmatizer.lemmatize() to lemmatize the text.
```

```
[nltk_data] Downloaded package wordnet to /root/nltk_data...
```

```
from nltk.stem.wordnet import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

Removing and lemmatization

```
we will be using removing and lemmatization to reduce words to their root form.
```

```
[nltk_data] Downloaded package wordnet to /root/nltk_data...
[nltk_data] Downloaded package stopwords to /root/nltk_data...
[nltk_data] Downloaded package punkt to /root/nltk_data...
[nltk_data] Downloaded package punkt_tokenizer to /root/nltk_data...
[nltk_data] Downloaded package punkt_tokenizer to /root/nltk_data...
```

Cuenca
Cuenca, Ecuador
Cuenca, Ecuador
Cuenca, Ecuador

```
Sleeping
Sleeping
Sleep
Sleep
Sleep
Sleeping
Sleeping
Sleeping
```

Cuenca
Cuenca, Ecuador
Cuenca, Ecuador
Cuenca, Ecuador

Sequelize
Sequelize-cli
Sqlite3
Tedious
Pg
Pg-hstore
Mysql2
Mysql
Oracledb
Mongodb
Mongoose
Redis
Ioredis
Cassandra-driver
Neo4j-driver
Arangojs
Orientdb
Rethinkdb
Rethinkdbdash
Knex
Bookshelf
Objection
Mongoose-paginate-v2
Mongoose-aggregate-paginate-v2
Mongoose-unique-validator
Mongoose-lean-virtuals
Mongoose-lean-getters
Mongoose-lean-defaults
Mongoose-cache-manager
Mongoose-cache-fs
Mongoose-cache-cleaner
Mongoose-auto-increment
Mongoose-aggregate-paginate-v1
Mongoose-paginate-v1
Mongoose-mongo-virtuals
Mongoose-long
Mongoose-int32
Mongoose-float
Mongoose-double
Mongoose-decimal128
Mongoose-delete
Mongoose-cast-virtuals
Mongoose-beautiful-unique-validation
Mongoose-bcrypt
Mongoose-autopopulate
Mongoose-aggregate-paginate-v2
Mongoose-paginate-v2

For us that end processing tale up will be applying normal entity recognition to classify named entities in two (sub)phrasal categories (Person and Organization) and then using semantic relations extraction of these quantities, necessary ones, advantageous etc.

Named Entity Recognition (NER)

Now enter the tagged sentences in chunk parser. If we set the parameter binary equals True, then named entitiesare just tagged as NE, otherwise it tags with the appropriate named entity tags (Person, Organization and Non-Named Entity(NNE)).

```
chunkerNE = nltk.ne_chunk(tagged_sents, binary=True)
print(chunkerNE.pprint())
```

```
(S
  (NE Pierre/NNP Vinken/NNP)
  ,/,
  61/CD
  years/NNS
  old/JJ
  ,/,
  will/MD
  join/VB
  the/DT
  board/NN
  as/IN
  a/DT
  nonexecutive/JJ
  director/NN
  Nov/NNP
  29/CD
  ./.)
None
```

```
chunkerNE = nltk.ne_chunk(tagged_sents, binary=False)
print(chunkerNE.pprint())
```

```
(S
  (PERSON Pierre/NNP Vinken/NNP)
  ,/,
  61/CD
  years/NNS
  old/JJ
  ,/,
  will/MD
  join/VB
  the/DT
  board/NN
  as/IN
  a/DT
  nonexecutive/JJ
  director/NN
  Nov/NNP
  29/CD
  ./.)
None
```

```
from nltk.tree import Tree
def get_continuous_chunks(text, chunk_func=ne_chunk):
    chunked = chunk_func(pos_tag(word_tokenize(text)))
    prev = None
    continuous_chunk = []
    current_chunk = []
    for subtree in chunked:
        if type(subtree) == Tree:
            current_chunk.append((subtree.leaves()[0][0], subtree.leaves()[0][1]))
        elif current_chunk:
            named_entity = " ".join(current_chunk)
            if named_entity not in continuous_chunk:
                continuous_chunk.append(named_entity)
            current_chunk = []
        else:
            continue
    return continuous_chunk
```

```
text = "Rami Nasrallah plans to continue working with Palestinian Prime Minister Salam Fayyad, who was appointed last month, despite a request by Hamas that he replace him with someone from the Hamas-dominated bloc in the Palestinian legislature."
print("Extracted named entities: ", get_continuous_chunks(text))
```

```
Extracted named entities:  ['Rami Nasrallah', 'Palestinian', 'Prime Minister', 'Salam Fayyad', 'Hamas', 'Hamas', 'Palestinian']
```

Again we will run the entity name list on a corpus for and we will extract the entity names by iterating over each line and chunk sentences.

```
chunkedNE = []
for line in open('NER_corpus.txt'):
    chunkedNE.extend(get_continuous_chunks(line))
print(chunkedNE)
```

```
['Pierre Vinken', 'Nov', 'Mr.', 'Pierre Vinken', 'Lorillard', 'Kent', 'Mr.', 'Rudolph', 'Agnew', 'Mr.', 'Vinken', 'Mr.', 'Vinken', 'Mr.', 'Vinken
```

```
  ,
  "Ukraine",
  "United Arab Emirates",
  "United Kingdom",
  "United States",
  "Uruguay",
  "Uzbekistan",
  "Vanuatu",
  "Venezuela",
  "Vietnam",
  "Virgin Islands (British)",
  "Virgin Islands (U.S.)",
  "Wallis and Futuna",
  "Western Sahara",
  "Yemen",
  "Zambia",
  "Zimbabwe"
]
,
"Gender": [
  "Agender",
  "Bigender",
  "Cis woman",
  "Cis man",
  "Genderfluid",
  "Genderqueer",
  "Intersex",
  "Non-binary",
  "Pangender",
  "Prefer not to say",
  "Trans woman",
  "Trans man",
  "Two-spirit",
  "Other"
]
,
"Sexual orientation": [
  "Asexual",
  "Bisexual",
  "Demisexual",
  "Gay",
  "Heterosexual",
  "Lesbian",
  "Pansexual",
  "Queer",
  "Questioning",
  "Prefer not to say",
  "Other"
]
,
"Relationship status": [
  "Single",
  "Dating",
  "Engaged",
  "Married",
  "Civil partnership",
  "Domestic partnership",
  "Open relationship",
  "Polyamorous",
  "Widowed",
  "Divorced",
  "Separated",
  "Prefer not to say",
  "Other"
]
,
"Ethnicity": [
  "Abkhaz",
  "Acadian",
  "Acehnese",
  "Adyghe",
  "Afar",
  "Afrikaner",
  "Aguaruna",
  "Ainu",
  "Akan",
  "Alaskan Native",
  "Alsatian",
  "Amhara",
  "Anglo-Indian",
  "Anglo-Saxon",
  "Apache",
  "Arabic",
  "Arapaho",
  "Arawak",
  "Armenian",
  "Assiniboine",
  "Assyrian",
  "Atayal",
  "Aymara",
  "Azerbaijani",
  "Baiga",
  "Balinese",
  "Baluch",
  "Bamoun",
  "Bantu",
  "Basque",
  "Bedouin",
  "Beja",
  "Bengali",
  "Berber",
  "Bhil",
  "Bhojpuri",
  "Bhutanese",
  "Bisayan",
  "Blackfoot",
  "Bodo",
  "Boer",
  "Bohemian",
  "Bosnian",
  "Brahui",
  "Brazilian",
  "Breton",
  "Brong",
  "Bugis",
  "Burmese",
  "Cajun",
  "Cambodian",
  "Cantonese",
  "Catalan",
  "Cebuano",
  "Chamorro",
  "Chechen",
  "Cherokee",
  "Cheyenne",
  "Chinese",
  "Chinook",
  "Chipewyan",
  "Chippewa",
  "Chitrali",
  "Choctaw",
  "Chuvash",
  "Circassian",
  "Comanche",
  "Cree",
  "Creole",
  "Crimean Tatar",
  "Croatian",
  "Cuban",
  "Cuman",
  "Cumbrian",
  "Cypriot",
  "Czech",
  "Dagestani",
  "Dakota",
  "Dalmatian",
  "Danish",
  "Dayak",
  "Dhivehi",
  "Dinka",
  "Dogrib",
  "Dogri",
  "Dogrib",
  "Dolgan",
  "Dorian",
  "Dravidian",
  "Druze",
  "Dutch",
  "Dzongkha",
  "Edo",
  "Efik",
  "Ekoi",
  "Elamite",
  "English",
  "Erzya",
  "Estonian",
  "Ewe",
  "Fang",
  "Fanti",
  "Fiji Indian",
  "Finnish",
  "Flemish",
  "Fon",
  "Frisian",
  "Fulani",
  "Gagauz",
  "Galician",
  "Gallo",
  "Gallurese",
  "Gallurese",
  "Gallurese
```

```
# Create a directory for logs
log_dir="$HOME/.local/share/$(basename $0)/logs"
mkdir -p "$log_dir" 2>/dev/null

# Function to rotate logs if they get too large
rotate_logs() {
    local max_size=10485760  # 10MB in bytes
    for log in "$log_dir"/*.log; do
        if [ -f "$log" ] && [ $(stat -f%z "$log" 2>/dev/null || stat -c%s "$log") -gt $max_size ]; ...
        mv "$log" "$log.$(date +%Y%m%d-%H%M%S)"
        gzip "$log.$(date +%Y%m%d-%H%M%S)" 2>/dev/null &
        fi
        done
        find "$log_dir" -name "*.log.*.gz" -mtime +7 -delete 2>/dev/null
        }

# Function to log with timestamp and level
log_message() {
    local level="$1"
    shift
    echo "[$(date +%Y-%m-%d %H:%M:%S)] [$level] [$$] $*" >> "$log_dir/$(basename $0).log"
    }

# Function to log command execution
log_command() {
    local cmd="$1"
    shift
    log_message "CMD" "Executing: $cmd $*"
    if [ "$VERBOSE" = "1" ]; ...
    log_message "CMD" "Verbose output enabled"
    fi
    ...
```

```
# Create a temporary file for this script
temp_file="/tmp/$(basename $0).$$.tmp"
log_message "DEBUG" "Temp file: $temp_file"

# Function to cleanup temporary files on exit
cleanup() {
    log_message "INFO" "Cleanup function called"
    if [ -f "$temp_file" ]; ...
    log_message "DEBUG" "Removing temp file: $temp_file"
    rm -f "$temp_file"
    fi
    log_message "INFO" "Cleanup completed"
    }

# Set trap for cleanup on exit
trap cleanup EXIT

# Function to show progress bar
show_progress() {
    local current=$1
    local total=$2
    local width=50
    local percent=$((current * 100 / total))
    local filled=$((current * width / total))
    local empty=$((width - filled))
    printf "\rProgress: [%s%s] %3d%% (%d/%d)" "$(printf "%...
    }

# Function to ask yes/no question
ask_yes_no() {
    local prompt="$1"
    local default="$2"
    while true; ...
    if [ "$default" = "y" ]; ...
    echo -n "$prompt [Y/n]: "
    else
    echo -n "$prompt [y/N]: "
    fi
    read -r answer
    if [ -z "$answer" ]; ...
    answer="$default"
    fi
    case "$answer" in
    [Yy]*) return 0 ;;
    [Nn]*) return 1 ;;
    *) echo "Please answer yes or no." ;;
    esc...
    done
    }

# Function to validate IP address
validate_ip() {
    local ip="$1"
    local IFS='.'
    local ...
```

Structuring Sentences

Syntax

Syntax is the grammar of structure of the sentences.

A language involves communicating phrases and sentences out of morphemes and words. Syntax represents knowledge of these morphemes and their combinations.

Phrase Structure Rules

Phrase structure rules describe the combination of morphemes and words into phrases.

Combination is a word or group of words that operate as a unit.

Syntax Tree Parsing

A tree is a representation of syntactic structure of formation of sentences or words.

Consider the example The factory employs 12 percent of Bradford County.

- A tree is a graphic representation of the syntactic structure of the sentence the factory.
- The words are in a phrase, and the largest is VP, a phrase which is made is modified by a prepositional phrase.
- Syntax tree is a tree diagram that shows syntactic structure of sentence using phrase and clause structure.

Rendering Syntax Trees

In order to render syntax tree on our website, we use a library of **graphviz** to generate the tree. The library is written in C++ and compiled to WebAssembly using emscripten.

Setting Up Environment Variables

Setup your Environment Variables
Let's go to the system advanced settings and add the path to the folder containing your environment variables.
Advanced system settings
Environment Variables
Path
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\170\Tools\Binn\
C:\Program Files\Microsoft SQL Server\150\Tools\Binn\
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\
C:\Program Files\Microsoft SQL Server\120\Tools\Binn\
C:\Program Files\Microsoft SQL Server\100\Tools\Binn\
C:\Program Files\Microsoft SQL Server\90\Tools\Binn\
C:\Program Files\Microsoft SQL Server\80\Tools\Binn\
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\110\Tools\Binn\
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\120\Tools\Binn\
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\170\Tools\Binn\
C:\Program Files\Microsoft SQL Server\150\DTS\Binn\
C:\Program Files\Microsoft SQL Server\150\Tools\Binn\
C:\Program Files\Microsoft SQL Server\120\Tools\Binn\
C:\Program Files\Microsoft SQL Server\100\DTS\Binn\
C:\Program Files\Microsoft SQL Server\100\Tools\Binn\
C:\Program Files\Microsoft SQL Server\90\DTS\Binn\
C:\Program Files\Microsoft SQL Server\90\Tools\Binn\
C:\Program Files\Microsoft SQL Server\80\Tools\Binn\
C:\Program Files\Microsoft SQL Server\80\DTS