

CS500-Data Science Tools and Technique

Introduction to Text Analysis

Bahar Ali
PhD Scholar,
National University Of Computer and Emerging Sciences,
Peshawar.

Introduction to Text Analysis

- Text analytics is an artificial intelligence (AI) technology that uses natural language processing (NLP) to transform the unstructured text in documents and databases into normalized, structured data suitable for analysis or to drive machine learning (ML) algorithms.
- Text Analytics is the process of deriving meaningful information from the natural language text using NLP
- NLP is the part of computer science and artificial intelligence which deal with the human languages.

Basic Techniques used in NLP

- **Tokenization:**

- Identifying the language units (words)
- E.g., White spaces

- **Stemming:**

- Stripping suffixes from the end of a word e.g., identify, identification, identifiable, identifying

- **Lemmatization:**

- Determining that the two words are having the same root, despite their surface differences
- E.g., song, sung, sang etc.

Tokenization

- For computer to understand any text, we need to break down it in a way that our machine can understand.
- Tokenization is a way of separating a piece of text into smaller units called tokens.
- Tokens can be either words, characters, or subwords, hence can be broadly classified into 3 types:

1. Word Tokenization:

- The most common way of forming tokens is based on space.
- For example, consider the sentence: “He is happy”.
- Assuming space as a delimiter, the tokenization of the sentence results in 3 tokens i.e. “He”, “is” and “happy”.

Tokenization

2. Character Tokenization:

- Similarly, tokens can either be characters.
- For example, let us consider text “Happy”
- Character tokens are; “H”, “a”, “p”, “p”, and “y”

3. Subword Tokenization:

- Subword Tokenization based on n-gram characters
- For example, let consider text “smartest”
- The subwords token using 4-gram window are; “smar” and “test”.

Stemming

- Stemming is the process of reducing inflection in words to their root form such as mapping a group of words to the stem.
- It does not keep a lookup table for the actual stem of the words, but apply algorithmic rules to generate stems
- Might not be an actual language word.
- Prefixes and suffix stripping
- For example, identification, identifiable, identifying are mapped to their stem “identify”

Stemming

- Steps to stem a document
 - Take a document as input
 - Read the document line by line
 - Tokenize the line
 - Stem the words
 - Output the stemmed words

Popular stemmer algorithms

- **Porter stemmer (1979):**
 - English Stemmer
 - The older one and widely used stemmer
 - Suffix Stripping
 - 5 rules for different cases that are applied in phases to generate stems (Step by step)
 - Python library: NLTK

Popular stemmer algorithms

- **Lancaster stemmer (1990):**
 - English Stemmer
 - This one is the most aggressive stemming algorithm
 - Iterative algorithm with rules saved externally
 - Due to iterations and over-stemming may occur.
 - Stemmed words are not intuitive for shorter words
 - Python library: NLTK
 - Using NLTK, you can easily add custom rules to this algorithm

Popular stemmer algorithms

- **Snowball stemmer (2001):**
 - English and Non-English stemmer
 - Improved version of porter stemmer
 - One can generate its own set of rules for any language
 - Has Implementation for many languages (Danish, Dutch, English, French, German, Hungarian, Italian, Norwegian, Porter, Portuguese, Romanian, Russian, Spanish, Swedish)
 - Python library: NLTK
- There are other Non-English stemmer like **ISRISemmer** (Arabic stemmer) and **RSLPStemmer** (Portuguese Stemmer)

Stemming Example

WORD	PORTER	LANCASTER	SNOWBALL
table	tabl	tabl	tabl
probably	probabl	prob	probabl
wolves	wolv	wolv	wolv
playing	play	play	play
is	is	is	is
dog	dog	dog	dog
the	the	the	the
beaches	beach	beach	beach
grounded	ground	ground	ground
dreamt	dreamt	dreamt	dreamt
envision	envis	envid	envis

Lemmatization

- Lemmatization groups together different inflected forms of a word so they can be analyzed as a single item, identified by the word's lemma, or dictionary form
- Like stemming it maps several words into one common root
- Unlike stemming it keeps a lookup table for the actual stem of the word (Uses WordNet corpus)
- Output of lemmatization is always a proper word
- Example: A lemmatizer will map “go”, “went”, “gone”, and “goes” to a common word i.e. “go”

Applications of Tokenization, Stemming and Lemmatization

- Sentiment Analysis
- Document Clustering
- Information retrieval

Bag of Words

- A simplifying representation used in natural language processing and information retrieval.
- A text is represented as the bag of its words, disregarding grammar and even word order but keeping multiplicity.
- Any information about the order or structure of words in the document is discarded.
- Stop words will be removed
- Need lemmatization and stemming
- Creating vectors for each document/ sentence
- Python Library: NLTK

Binary Bag of Words

Lowering words

Stemming

Lemmatization

Stop words removal etc.

Bag of Words

Sentence 1: He is a good boy

Sentence 2: She is an intelligent girl

Sentence 3: He is a good boy and smart boy

Sentence 1: good boy

Sentence 2: intelligent girl

Sentence 3: good boy smart boy

**Binary
Vectors** →

	good	boy	intelligent	girl	smart	Output
Sentence 1	1	1	0	0	0	?
Sentence 2	0	0	1	1	0	?
Sentence 3	1	1	0	0	1	?

Bag of Words

Lowering words

Stemming

Lemmatization

Stop words removal etc.

Bag of Words

Sentence 1: He is a good boy

Sentence 2: She is an intelligent girl

Sentence 3: He is a good boy and smart boy

Sentence 1: good boy

Sentence 2: intelligent girl

Sentence 3: good boy smart boy

Vectors →

	good	boy	intelligent	girl	smart	Output
Sentence 1	1	1	0	0	0	?
Sentence 2	0	0	1	1	0	?
Sentence 3	1	2	0	0	1	?

Bag of Words: Disadvantages

- In sentiment analysis, we must know which word is semantically more significant in a specific context
- Bag of words assigns equal weightage to all kind of words, thus, we cannot say which word is more semantically significant
 - ***TF-IDF** is the solution to this*
- Bag of word does not work well for huge amount of data
 - ***Word2vec** is the solution to this*

TF-IDF

- **TF-IDF** (Term frequency-Inverse document frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents
- **TF** counts the number of words in a document/ sentence.
- **IDF** suppresses the effect of common words in a document.
- **TF-IDF** is a vector representation of Text
- Python Library: NLTK

TF-IDF

*TFIDF score for term i in document $j = TF(i,j) * IDF(i)$*

where

IDF = Inverse Document Frequency

TF = Term Frequency

$$TF(i,j) = \frac{\text{Term } i \text{ frequency in document } j}{\text{Total words in document } j}$$

$$IDF(i) = \log_2 \left(\frac{\text{Total documents}}{\text{documents with term } i} \right)$$

TF-IDF

Lowering words

Stemming

Lemmatization

Stop words removal etc.

Bag of Words

Sentence 1: He is a good boy

Sentence 2: She is an intelligent girl

Sentence 3: He is a good boy and smart boy

Sentence 1: good boy

Sentence 2: intelligent girl

Sentence 3: good boy smart boy

TF →

	Sentence 1	Sentence 2	Sentence 3
good	1/2	0	1/4
Boy	1/2	0	2/4
intelligent	0	1/2	0
girl	0	1/2	0
smart	0	0	1/4

IDF →

Words	IDF
good	$\log(3/2)$
Boy	$\log(3/2)$
intelligent	$\log(3/1)$
girl	$\log(3/1)$
smart	$\log(3/1)$

TF-IDF

TF →

	Sentence 1	Sentence 2	Sentence 3
good	1/2	0	1/4
Boy	1/2	0	2/4
intelligent	0	1/2	0
girl	0	1/2	0
smart	0	0	1/4

IDF →

Words	IDF
good	$\log(3/2)$
Boy	$\log(3/2)$
intelligent	$\log(3/1)$
girl	$\log(3/1)$
smart	$\log(3/1)$

TF-IDF →

	good	boy	intelligent	girl	Smart	Output
Sentence 1	$1/2 * \log(3/2)$	$1/2 * \log(3/2)$	$0 * \log(3/1)$	$0 * \log(3/1)$	$0 * \log(3/1)$?
Sentence 2	$0 * \log(3/2)$	$0 * \log(3/2)$	$1/2 * \log(3/1)$	$1/2 * \log(3/1)$	$0 * \log(3/1)$?
Sentence 3	$1/4 * \log(3/2)$	$2/4 * \log(3/2)$	$0 * \log(3/1)$	$0 * \log(3/1)$	$1/4 * \log(3/1)$?

One hot Encoding

- In digital circuits and machine learning, a one-hot is a group of bits with a single high bit and all the others low. **OR** A group of bits with a single '1' and all the others are '0'.
- Machine algorithm works on numerical values
- **One hot encoding** is a process by which categorical variables are converted into numerical vectors (Male = 0 1 and Female=1 0)

Limitations of One hot vectors:

- Resulting very sparse matrix, most of the values are zeros
- Thus having high space and computational complexity
- Having no relationship with the similar words, so it is difficult to generalize such attributes for prediction

Word Embedding

- A learned representation for text where words that have the same meaning have a similar representation.
- A feature learning techniques in a NLP where words or phrases from the vocabulary are mapped to meaningful vectors of real numbers
- An improvement over frequency based approaches
- Useful for NLP as well as deep learning
- Python Libraries: NLTK Library, Keras Library (LSTM)

Word Embedding

- Apple is good for health (Here Apple refer to Fruit)
- Apple is good place to work (Here Apple refer to Place)
- Thus Apple in above sentences has semantically different meaning with the context

Word Embedding

- **The working principal:**

- Meaning of the word is learned by analyzing its neighbors

- **How to use it**

- User pre trained model (Word2vec, Glove)
 - Train your own Embedding using Word2vec
 - Train your own Embedding using Keras Library

Word Embedding Types

- **Word Embedding are of two types:**

1. Frequency Based:

- Count Vectorization (aka One-Hot Encoding)
- TF-IDF
- Co-occurrence Matrix i.e. Glove

2. Prediction Based:

- Continuous Bag of Words Model (CBOW) i.e. Word2vec
- Skip-gram

Word Embedding: Glove

- Co-occurrence matrix based word embedding Implementation

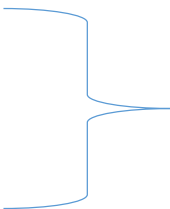
Text: He is a good boy, he is intelligent.

	He	is	a	good	boy	intelligent
He	0	2	0	0	0	0
is	2	0	1	0	0	0
a	0	1	0	1	0	0
good	0	0	1	0	1	0
boy	0	0	0	1	0	0
intelligent	0	1	0	0	0	0

Word Embedding: Word2vec

- CBOW based word embedding implementation by Google
- Vectors are created based on feature set.

	Fruit	Eatable	Sphere	Animal
Apple	0.96	0.95	0.8	0.1
Ball	0.1	0.1	0.94	0.15
Horse	0.13	0.53	0.2	0.95



Vectors with 4 dimensions

Word Embedding: Prediction based

- In case of CBOW, it predicts next word given context word
- In case of Skip-gram, it predicts the context word given the next word
- Let Consider a sentence; “He is working fine”
- For sake of simplicity, consider one hot values

	F1	F2	F3	F3
He	1	0	0	0
Is	0	1	0	0
Working	0	0	1	0
Fine	0	0	0	1

Considering Window size = 1

→ Input Vector to NN

→ Output Vector to NN