Natural Language Processing

An Introduction

Sri Vallabha Deevi, Ph.D.

Director, Data Science, Tiger Analytics.
Adjunct Faculty, Data Science & AI, IIT Madras.

Table of contents

Introduction

Unstructured Interactions

Unstructured Interactions

- Unstructured interactions are widely used today

Unstructured Interactions

- Unstructured interactions are widely used today
- Collecting feedback on webpages and apps

Unstructured Interactions

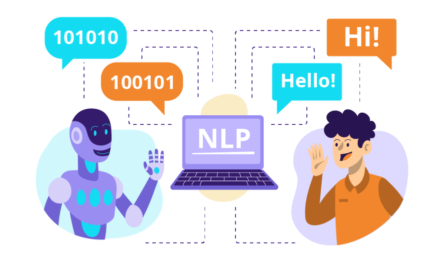
- Unstructured interactions are widely used today
- Collecting feedback on webpages and apps
- Chatting with users and providing answers

Unstructured Interactions

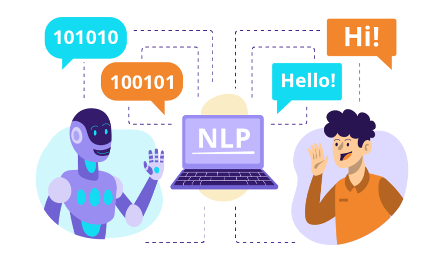
- Unstructured interactions are widely used today
- Collecting feedback on webpages and apps
- Chatting with users and providing answers
- Voice assistants that accomplish a wide variety of tasks

Image: https://techcrunch.com/wp-content/uploads/2018/10/Assistant_1.png

Natural Language Processing

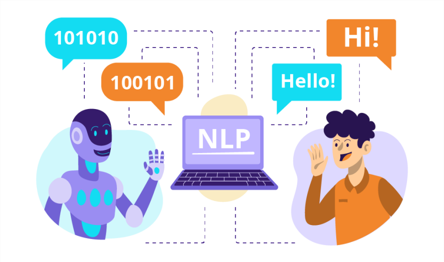


Natural Language Processing



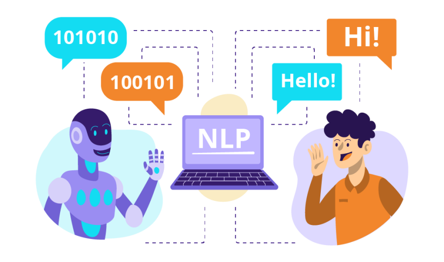
- Natural Language Processing deals with unstructured interactions

Natural Language Processing



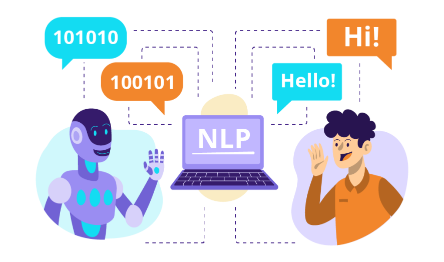
- Natural Language Processing deals with unstructured interactions
- Using human language as input

Natural Language Processing



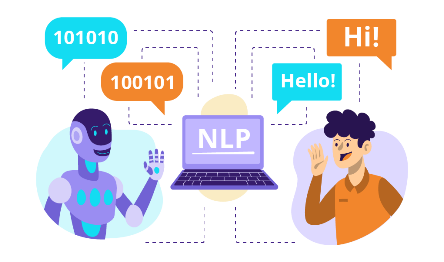
- Natural Language Processing deals with unstructured interactions
- Using human language as input
- Understanding the language - sentiment, intent..

Natural Language Processing



- Natural Language Processing deals with unstructured interactions
- Using human language as input
- Understanding the language - sentiment, intent..
- Taking actions based on the understanding - question answering, summarization, data retrieval..

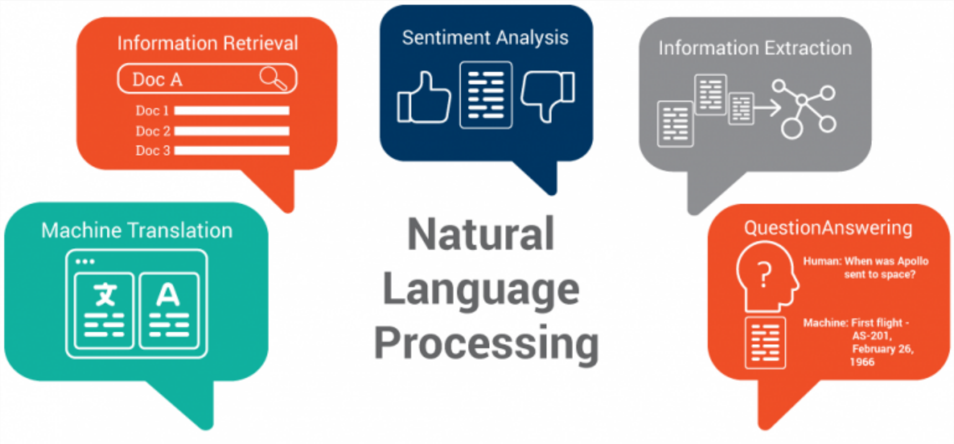
Natural Language Processing



- Natural Language Processing deals with unstructured interactions
- Using human language as input
- Understanding the language - sentiment, intent..
- Taking actions based on the understanding - question answering, summarization, data retrieval..
- Represents text as numbers, that Machine Learning models can understand

Image: `https://images.app.goo.gl/C7otX6X8TdFWfYPY7`

NLP Tasks



Information Retrieval
Doc A
Doc 1
Doc 2
Doc 3
Sentiment Analysis
Information Extraction
Machine Translation
文
A
Natural Language Processing
QuestionAnswering
?
Human: When was Apollo sent to space?
Machine: First flight - AS-201, February 26, 1966

Challenges

- Processing unstructured text different from processing structured data

Challenges

- Processing unstructured text different from processing structured data
- There is no fixed length and order to the input

Challenges

- Processing unstructured text different from processing structured data
- There is no fixed length and order to the input
- Have to know what each word means in a certain position

Challenges

- Processing unstructured text different from processing structured data
- There is no fixed length and order to the input
- Have to know what each word means in a certain position
- Each language has grammar, that has complex rules

Challenges

- Processing unstructured text different from processing structured data
- There is no fixed length and order to the input
- Have to know what each word means in a certain position
- Each language has grammar, that has complex rules
- And a wide variety of expression

Expression in natural language

In languages, especially English, the same meaning can be conveyed in different ways.

Expression in natural language

In languages, especially English, the same meaning can be conveyed in different ways.

- As soon as it saw the lightning, the dog ran to its owner

Expression in natural language

In languages, especially English, the same meaning can be conveyed in different ways.

- As soon as it saw the lightning, the dog ran to its owner
- The dog had run to its owner on seeing the lightning

Expression in natural language

In languages, especially English, the same meaning can be conveyed in different ways.

- As soon as it saw the lightning, the dog ran to its owner
- The dog had run to its owner on seeing the lightning
- The dog, on seeing the lightning, ran to its owner

Expression in natural language

In languages, especially English, the same meaning can be conveyed in different ways.

- As soon as it saw the lightning, the dog ran to its owner
- The dog had run to its owner on seeing the lightning
- The dog, on seeing the lightning, ran to its owner
- Lightning made the dog run to its owner

Punctuation

Punctuation can lead to different meanings for the same words.

Punctuation

Punctuation can lead to different meanings for the same words.

- Hang him not, save him!

Punctuation

Punctuation can lead to different meanings for the same words.

- Hang him not, save him!
- Hang him, not save him!

Usage of stop words

Many words are used to help sentence formation, but do not add much to the meaning.

Usage of stop words

Many words are used to help sentence formation, but do not add much to the meaning.

Verbose statements

Usage of stop words

Many words are used to help sentence formation, but do not add much to the meaning.

Verbose statements

- this is the most dumb movie that i have ever seen

Usage of stop words

Many words are used to help sentence formation, but do not add much to the meaning.

Verbose statements

- this is the most dumb movie that i have ever seen
- this place is as scenic as some of the most popular places in the world

Usage of stop words

Many words are used to help sentence formation, but do not add much to the meaning.

Verbose statements

- this is the most dumb movie that i have ever seen
- this place is as scenic as some of the most popular places in the world

Concise summarization

Usage of stop words

Many words are used to help sentence formation, but do not add much to the meaning.

Verbose statements

- this is the most dumb movie that i have ever seen
- this place is as scenic as some of the most popular places in the world

Concise summarization

- dumb movie

Usage of stop words

Many words are used to help sentence formation, but do not add much to the meaning.

Verbose statements

- this is the most dumb movie that i have ever seen
- this place is as scenic as some of the most popular places in the world

Concise summarization

- dumb movie
- very scenic place

Preprocessing Methods

Preprocessing Language

- Preprocessing language is essential for standardization

Preprocessing Language

- Preprocessing language is essential for standardization
- Make it consistent for the machine to understand

Preprocessing Language

- Preprocessing language is essential for standardization
- Make it consistent for the machine to understand
- While reducing the variety of ways in which a fact is expressed

Preprocessing Language

- Preprocessing language is essential for standardization
- Make it consistent for the machine to understand
- While reducing the variety of ways in which a fact is expressed
- But preserving the richness of expression

Preprocessing Language

- Preprocessing language is essential for standardization
- Make it consistent for the machine to understand
- While reducing the variety of ways in which a fact is expressed
- But preserving the richness of expression
- Pre-processing methods evolved along with NLP techniques

Preprocessing Language

- Preprocessing language is essential for standardization
- Make it consistent for the machine to understand
- While reducing the variety of ways in which a fact is expressed
- But preserving the richness of expression
- Pre-processing methods evolved along with NLP techniques
- Earliest NLP techniques had standard pre-processing methods

Preprocessing Language

- Preprocessing language is essential for standardization
- Make it consistent for the machine to understand
- While reducing the variety of ways in which a fact is expressed
- But preserving the richness of expression
- Pre-processing methods evolved along with NLP techniques
- Earliest NLP techniques had standard pre-processing methods
- Latter models incorporated pre-processing into process flow, doing away the need for separate pre-processing

Natural Language Processing

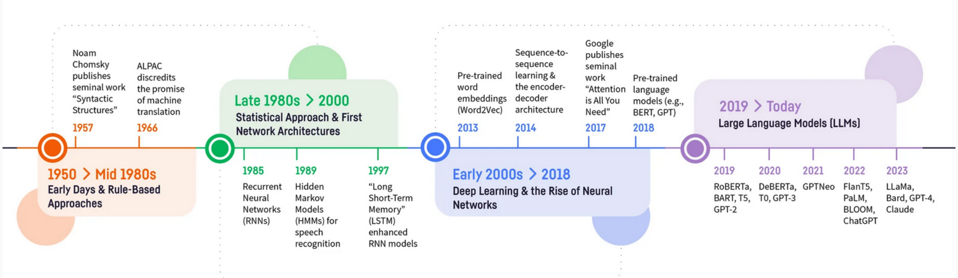


Image: `https://blog.dataiku.com/nlp-metamorphosis`

Some Milestones

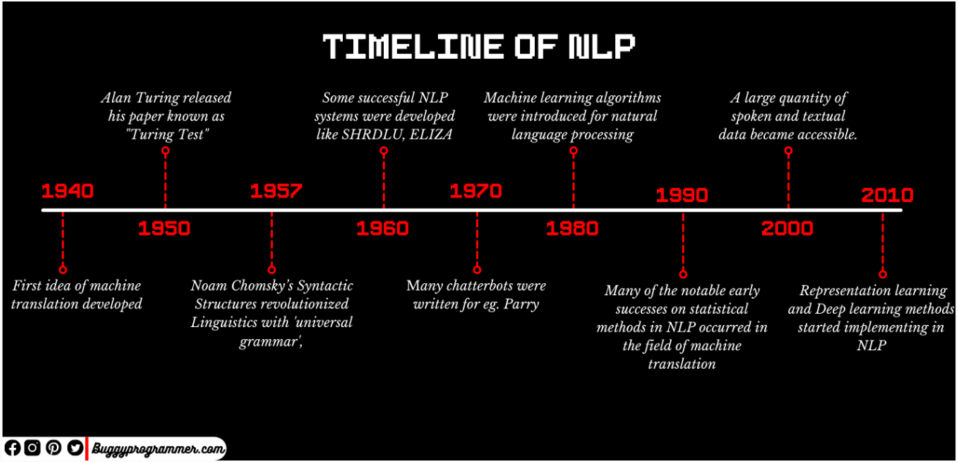


Image: `Buggyprogrammer.com`

Pre-processing Techniques

Earliest NLP techniques relied on reducing the number of unique words needed for the model

- Tokenization
- Stop Word Removal
- Stemming
- Lemmatization
- n-grams

Text pre-processing

Text needs to be pre-processed before analysis.

Text pre-processing

Text needs to be pre-processed before analysis.

- Noise removal: Removing headers/HTML tags etc from text file

Text pre-processing

Text needs to be pre-processed before analysis.

- Noise removal: Removing headers/HTML tags etc from text file
- Normalization: Making the text uniform for processing (removing punctuation, same case)

Text pre-processing

Text needs to be pre-processed before analysis.

- Noise removal: Removing headers/HTML tags etc from text file
- Normalization: Making the text uniform for processing (removing punctuation, same case)
- Stop word removal: Words that are very common and do not have an impact on the meaning

Text pre-processing

Text needs to be pre-processed before analysis.

- Noise removal: Removing headers/HTML tags etc from text file
- Normalization: Making the text uniform for processing (removing punctuation, same case)
- Stop word removal: Words that are very common and do not have an impact on the meaning
- Stemming: Reducing verbs to simple tense for analysis

Text pre-processing

Text needs to be pre-processed before analysis.

- Noise removal: Removing headers/HTML tags etc from text file
- Normalization: Making the text uniform for processing (removing punctuation, same case)
- Stop word removal: Words that are very common and do not have an impact on the meaning
- Stemming: Reducing verbs to simple tense for analysis
- Lemmatization: Reduces words to their base form

Text pre-processing

Text needs to be pre-processed before analysis.

- Noise removal: Removing headers/HTML tags etc from text file
- Normalization: Making the text uniform for processing (removing punctuation, same case)
- Stop word removal: Words that are very common and do not have an impact on the meaning
- Stemming: Reducing verbs to simple tense for analysis
- Lemmatization: Reduces words to their base form
- Tokenization: Splitting longer strings of text into smaller pieces, usually words

Noise removal

Text Strings with html tags	Cleaned Results
Excel 365	Excel 365
Insert multiple blank rows	Insert multiple blank rows
<i>Merge all sheets into one</i>	Merge all sheets into one
Office Tab	Office Tab
Kutools for Excel	Kutools for Excel
www.extendoffice.com	www.extendoffice.com

Image: https://images.app.goo.gl/vmivtWRWVkSGGLJW6

Normalization

- Text is converted to lower case, to make it uniform

Normalization

- Text is converted to lower case, to make it uniform
- Reduces dimensionality of data. Ex. Cat and cat are the same

Normalization

- Text is converted to lower case, to make it uniform
- Reduces dimensionality of data. Ex. Cat and cat are the same
- Improves performance of NLP algorithms

Normalization

- Text is converted to lower case, to make it uniform
- Reduces dimensionality of data. Ex. Cat and cat are the same
- Improves performance of NLP algorithms
- Leads to loss of information, about proper nouns and emphasis

Normalization

- Text is converted to lower case, to make it uniform
- Reduces dimensionality of data. Ex. Cat and cat are the same
- Improves performance of NLP algorithms
- Leads to loss of information, about proper nouns and emphasis
- “This is SPARTA” $\rightarrow$ “this is sparta”

Stop word removal

- Some words are very commonly used - “the”, “a”, “an”, “so”, “what”

Stop word removal

- Some words are very commonly used - “the”, “a”, “an”, “so”, “what”
- May not add much meaning to the sentence

Stop word removal

- Some words are very commonly used - “the”, “a”, “an”, “so”, “what”
- May not add much meaning to the sentence
- Can be removed without loss of generality

Stop word removal

- Some words are very commonly used - “the”, “a”, “an”, “so”, “what”
- May not add much meaning to the sentence
- Can be removed without loss of generality
- Some times may lead to loss of meaning

Stop word removal

- Some words are very commonly used - "the", "a", "an", "so", "what"
- May not add much meaning to the sentence
- Can be removed without loss of generality
- Some times may lead to loss of meaning
- "The movie was not good at all." $\rightarrow$ "movie good" or "movie not good"

Stop word removal

- Some words are very commonly used - "the", "a", "an", "so", "what"
- May not add much meaning to the sentence
- Can be removed without loss of generality
- Some times may lead to loss of meaning
- "The movie was not good at all." $\rightarrow$ "movie good" or "movie not good"
- "The movie was not ok, its a bad one." $\rightarrow$ "movie bad" or "movie not bad"

Stop word removal

- Some words are very commonly used - “the”, “a”, “an”, “so”, “what”
- May not add much meaning to the sentence
- Can be removed without loss of generality
- Some times may lead to loss of meaning
- “The movie was not good at all.” $\rightarrow$ “movie good” or “movie not good”
- “The movie was not ok, its a bad one.” $\rightarrow$ “movie bad” or “movie not bad”
- Have to choose appropriate stop words to remove

Stemming

- Stemming reduces words to their root form

Stemming

- Stemming reduces words to their root form
- Removes or replaces word suffixes & prefixes based on set of rules/heuristics

Stemming

- Stemming reduces words to their root form
- Removes or replaces word suffixes & prefixes based on set of rules/heuristics
- Variants - Porter Stemmer, Lancaster Stemmer, Snowball Stemmer

Stemming

- Stemming reduces words to their root form
- Removes or replaces word suffixes & prefixes based on set of rules/heuristics
- Variants - Porter Stemmer, Lancaster Stemmer, Snowball Stemmer
- Text simplification, improved model performance and standardization

Stemming example

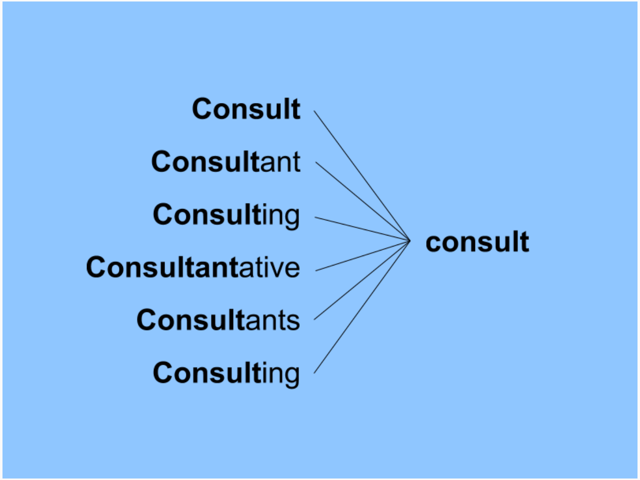


Image: `https://images.app.goo.gl/gRobrUoMPV3xNKA16`

Lemmatization

- Uses vocabulary and morphological analysis of words

https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

Lemmatization

- Uses vocabulary and morphological analysis of words
- To remove inflectional endings and return the base or dictionary form of the word

https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

Lemmatization

- Uses vocabulary and morphological analysis of words
- To remove inflectional endings and return the base or dictionary form of the word
- Example: Stemming 'saw' might return 's', while Lemmatization will return 'see' or 'saw' based on whether the token was used as verb or noun

https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

Lemmatization example

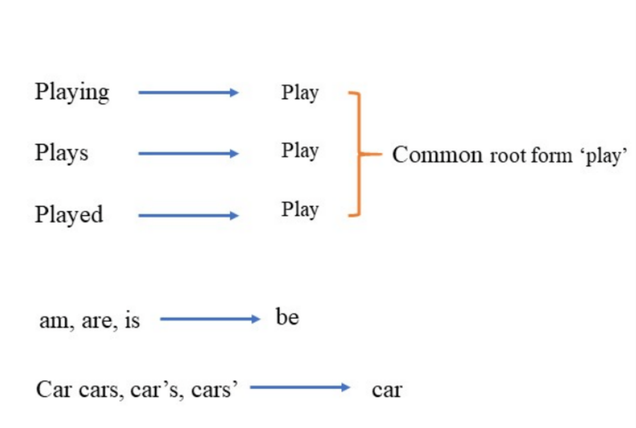


Image: https://github.com/Learn-Write-Repeat/Open-contributions/blob/master/B2-NLP/Amey_Nlp_Lemmatization_stemming.md

Stemming vs. Lemmatization

Stemming commonly collapses derivationally similar words, while Lemmatization only collapses different inflectional form of a lemma.

Image: https://www.johnsnowlabs.com/boost-your-nlp-results-with-spark-nlp-stemming-and-lemmatizing-techniques/

Stemming vs. Lemmatization

Stemming commonly collapses derivationally similar words, while Lemmatization only collapses different inflectional form of a lemma.

Image: https://www.johnsnowlabs.com/boost-your-nlp-results-with-spark-nlp-stemming-and-lemmatizing-techniques/

Tokenization

- Tokenization is breaking the long string into smaller parts (discrete elements)

Tokenization

- Tokenization is breaking the long string into smaller parts (discrete elements)
- Token occurrences in a document can be treated as a vector representation of the document

Tokenization

- Tokenization is breaking the long string into smaller parts (discrete elements)
- Token occurrences in a document can be treated as a vector representation of the document
- Two basic types - sentence tokenization and word tokenization

Tokenization

- Tokenization is breaking the long string into smaller parts (discrete elements)
- Token occurrences in a document can be treated as a vector representation of the document
- Two basic types - sentence tokenization and word tokenization
- Simple tokenization can break the strings using punctuation marks, like . and ?

Tokenization

- Tokenization is breaking the long string into smaller parts (discrete elements)
- Token occurrences in a document can be treated as a vector representation of the document
- Two basic types - sentence tokenization and word tokenization
- Simple tokenization can break the strings using punctuation marks, like . and ?
- Needs care to ensure that meaning is not lost

Tokenization methods - NLTK

Natural Language Toolkit is an open source Python library for NLP. Has interfaces for different corpora like WordNet

Image: https://neptune.ai/blog/tokenization-in-nlp

Tokenization methods - NLTK

Natural Language Toolkit is an open source Python library for NLP. Has interfaces for different corpora like WordNet

- Whitespace tokenization

Image: https://neptune.ai/blog/tokenization-in-nlp

Tokenization methods - NLTK

Natural Language Toolkit is an open source Python library for NLP. Has interfaces for different corpora like WordNet

- Whitespace tokenization
  - Break text using white space as delimiter

Image: `https://neptune.ai/blog/tokenization-in-nlp`

Tokenization methods - NLTK

Natural Language Toolkit is an open source Python library for NLP. Has interfaces for different corpora like WordNet

- Whitespace tokenization
  - Break text using white space as delimiter
  - Can preserve hyphenated words like 'pre-processing'

Image: https://neptune.ai/blog/tokenization-in-nlp

Tokenization methods - NLTK

Natural Language Toolkit is an open source Python library for NLP. Has interfaces for different corpora like WordNet

- Whitespace tokenization
  - Break text using white space as delimiter
  - Can preserve hyphenated words like 'pre-processing'
- Punctuation based tokenizer

Image: https://neptune.ai/blog/tokenization-in-nlp

Tokenization methods - NLTK

Natural Language Toolkit is an open source Python library for NLP. Has interfaces for different corpora like WordNet

- Whitespace tokenization
  - Break text using white space as delimiter
  - Can preserve hyphenated words like 'pre-processing'
- Punctuation based tokenizer
  - Split based on punctuation and white spaces

Image: `https://neptune.ai/blog/tokenization-in-nlp`

Tokenization methods - NLTK

Natural Language Toolkit is an open source Python library for NLP. Has interfaces for different corpora like WordNet

- Whitespace tokenization
  - Break text using white space as delimiter
  - Can preserve hyphenated words like 'pre-processing'
- Punctuation based tokenizer
  - Split based on punctuation and white spaces
  - Splits on different punctuation marks like '! . , ' etc.

Image: `https://neptune.ai/blog/tokenization-in-nlp`

Tokenization methods - NLTK

Natural Language Toolkit is an open source Python library for NLP. Has interfaces for different corpora like WordNet

- Whitespace tokenization
  - Break text using white space as delimiter
  - Can preserve hyphenated words like 'pre-processing'
- Punctuation based tokenizer
  - Split based on punctuation and white spaces
  - Splits on different punctuation marks like '! . , ' etc.
- Treebank tokenizer

Image: `https://neptune.ai/blog/tokenization-in-nlp`

Tokenization methods - NLTK

Natural Language Toolkit is an open source Python library for NLP. Has interfaces for different corpora like WordNet

- Whitespace tokenization
  - Break text using white space as delimiter
  - Can preserve hyphenated words like 'pre-processing'
- Punctuation based tokenizer
  - Split based on punctuation and white spaces
  - Splits on different punctuation marks like '! . , ' etc.
- Treebank tokenizer
  - Incorporates common rules for English words

Image: `https://neptune.ai/blog/tokenization-in-nlp`

Tokenization methods - NLTK

Natural Language Toolkit is an open source Python library for NLP. Has interfaces for different corpora like WordNet

- Whitespace tokenization
  - Break text using white space as delimiter
  - Can preserve hyphenated words like 'pre-processing'
- Punctuation based tokenizer
  - Split based on punctuation and white spaces
  - Splits on different punctuation marks like '! . , ' etc.
- Treebank tokenizer
  - Incorporates common rules for English words
  - Ex. “don't ”is broken as “do ”and “n't ”

Image: `https://neptune.ai/blog/tokenization-in-nlp`

Tokenization methods - NLTK

Natural Language Toolkit is an open source Python library for NLP. Has interfaces for different corpora like WordNet

- Whitespace tokenization
  - Break text using white space as delimiter
  - Can preserve hyphenated words like 'pre-processing'
- Punctuation based tokenizer
  - Split based on punctuation and white spaces
  - Splits on different punctuation marks like '! . , ' etc.
- Treebank tokenizer
  - Incorporates common rules for English words
  - Ex. “don't ”is broken as “do ”and “n't ”
  - Retains decimal numbers as single token

Image: `https://neptune.ai/blog/tokenization-in-nlp`

Tokenization methods - NLTK

Natural Language Toolkit is an open source Python library for NLP. Has interfaces for different corpora like WordNet

- Whitespace tokenization
  - Break text using white space as delimiter
  - Can preserve hyphenated words like 'pre-processing'
- Punctuation based tokenizer
  - Split based on punctuation and white spaces
  - Splits on different punctuation marks like '! . , ' etc.
- Treebank tokenizer
  - Incorporates common rules for English words
  - Ex. "don't "is broken as "do "and "n't "
  - Retains decimal numbers as single token
  - Separates based on phrase splitting punctuations like ?!

Image: https://neptune.ai/blog/tokenization-in-nlp

Tokenization methods - NLTK

- Tweet tokenizer

Image: https://neptune.ai/blog/tokenization-in-nlp

Tokenization methods - NLTK

- Tweet tokenizer
  - Tokenizer for tweets

Image: `https://neptune.ai/blog/tokenization-in-nlp`

Tokenization methods - NLTK

- Tweet tokenizer
  - Tokenizer for tweets
  - Separates emojis from text, for analysis

Image: https://neptune.ai/blog/tokenization-in-nlp

Tokenization methods - NLTK

- Tweet tokenizer
  - Tokenizer for tweets
  - Separates emojis from text, for analysis
- MWET tokenizer

Image: https://neptune.ai/blog/tokenization-in-nlp

Tokenization methods - NLTK

- Tweet tokenizer
  - Tokenizer for tweets
  - Separates emojis from text, for analysis
- MWET tokenizer
  - Multi word expression tokenizer

Image: `https://neptune.ai/blog/tokenization-in-nlp`

Tokenization methods - NLTK

- Tweet tokenizer
  - Tokenizer for tweets
  - Separates emojis from text, for analysis
- MWET tokenizer
  - Multi word expression tokenizer
  - Can merge required words into single tokens for analysis

Image: https://neptune.ai/blog/tokenization-in-nlp

Other Tokenizers

- TextBlob tokenizer - has rules for word contractions

Other Tokenizers

- TextBlob tokenizer - has rules for word contractions
- spaCy tokenizer - flexibility for special tokens that need not be segmented

Other Tokenizers

- TextBlob tokenizer - has rules for word contractions
- spaCy tokenizer - flexibility for special tokens that need not be segmented
- Gensim tokenizer - package for topic modeling, document indexing and similarity retrieval

Other Tokenizers

- TextBlob tokenizer - has rules for word contractions
- spaCy tokenizer - flexibility for special tokens that need not be segmented
- Gensim tokenizer - package for topic modeling, document indexing and similarity retrieval
- Keras tokenizer - convert alphabet to lower case before tokenizing

Other Tokenizers

- TextBlob tokenizer - has rules for word contractions
- spaCy tokenizer - flexibility for special tokens that need not be segmented
- Gensim tokenizer - package for topic modeling, document indexing and similarity retrieval
- Keras tokenizer - convert alphabet to lower case before tokenizing
- Choice of tokenization governed by applications

Other Tokenizers

- TextBlob tokenizer - has rules for word contractions
- spaCy tokenizer - flexibility for special tokens that need not be segmented
- Gensim tokenizer - package for topic modeling, document indexing and similarity retrieval
- Keras tokenizer - convert alphabet to lower case before tokenizing
- Choice of tokenization governed by applications
- Each tokenizer might have slight differences in the way words are broken

An example

Original text : Alice in wonderland

An example

Original text : Alice in wonderland

```
Alice was beginning to get very tired of sitting by her sister on the bank, and of
     having nothing to do. Once or twice she had peeped into the book her sister
    was reading, but it had no pictures or conversations in it, "and what is the
    use of a book," thought Alice, "without pictures or conversations?"
```

An example

Word tokenize: Alice in wonderland

An example

Word tokenize: Alice in wonderland

```
['Alice', 'was', 'beginning', 'to', 'get', 'very', 'tired', 'of', 'sitting', 'by',
    'her', 'sister', 'on', 'the', 'bank', ',', 'and', 'of', 'having', 'nothing', '
    to', 'do', '.', 'Once', 'or', 'twice', 'she', 'had', 'peeped', 'into', 'the', '
    book', 'her', 'sister', 'was', 'reading', ',', 'but', 'it', 'had', 'no', '
    pictures', 'or', 'conversations', 'in', 'it', ',', '``', 'and', 'what', 'is', '
    the', 'use', 'of', 'a', 'book', ',', "''", 'thought', 'Alice', ',', '``', '
    without', 'pictures', 'or', 'conversations', '?', "''"]
```

An example

Stemming: Alice in wonderland

An example

Stemming: Alice in wonderland

```
beginning : begin

very : veri

tired : tire

sitting : sit

having : have
```

An example

Word stemming: Alice in wonderland

An example

Word stemming: Alice in wonderland

```
['alic', 'wa', 'begin', 'get', 'veri', 'tire', 'sit', 'sister', 'bank', ',', 'noth
    ', '.', 'onc', 'twice', 'peep', 'book', 'sister', 'wa', 'read', ',', 'pictur',
    'convers', ',', '``', 'use', 'book', ',', "''", 'thought', 'alic', ',', '``', '
    without', 'pictur', 'convers', '?', "''"]
```

Word Representation

Word representation

- Pre-processing reduces the wide variety of words that need to be handled

Word representation

- Pre-processing reduces the wide variety of words that need to be handled
- After this, words are converted to numerical form for training

Word representation

- Pre-processing reduces the wide variety of words that need to be handled
- After this, words are converted to numerical form for training
- Representing these words in numerical form can be done in a variety of ways

Word representation

- Pre-processing reduces the wide variety of words that need to be handled
- After this, words are converted to numerical form for training
- Representing these words in numerical form can be done in a variety of ways
  - Bag of Words, One hot encoding

Word representation

- Pre-processing reduces the wide variety of words that need to be handled
- After this, words are converted to numerical form for training
- Representing these words in numerical form can be done in a variety of ways
  - Bag of Words, One hot encoding
  - TF-IDF

Word representation

- Pre-processing reduces the wide variety of words that need to be handled
- After this, words are converted to numerical form for training
- Representing these words in numerical form can be done in a variety of ways
    - Bag of Words, One hot encoding
    - TF-IDF
    - Word2Vec, GloVe

Word representation

- Pre-processing reduces the wide variety of words that need to be handled
- After this, words are converted to numerical form for training
- Representing these words in numerical form can be done in a variety of ways
  - Bag of Words, One hot encoding
  - TF-IDF
  - Word2Vec, GloVe
  - SentBERT

Bag of Words

- Build list of unique words, called vocabulary

Bag of Words

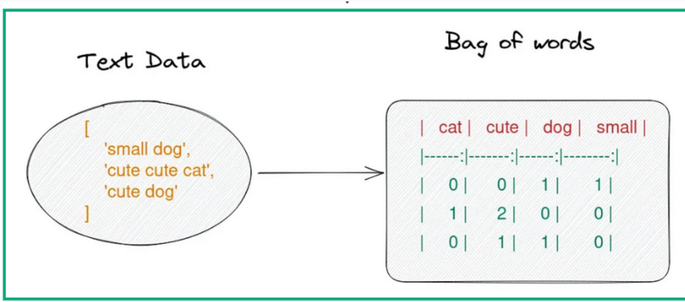
- Build list of unique words, called vocabulary
- Can consider bi-grams, tri-grams to improve performance

Bag of Words

- Build list of unique words, called vocabulary
- Can consider bi-grams, tri-grams to improve performance
- Each document: vector with each word in vocabulary having a) 1 or 0 for present/absent b) count of occurrence

Bag of Words

- Build list of unique words, called vocabulary
- Can consider bi-grams, tri-grams to improve performance
- Each document: vector with each word in vocabulary having a) 1 or 0 for present/absent b) count of occurrence



Term Frequency - Inverse Document frequency

- Term frequency: no of times a term appears in document / number of terms in document

Term Frequency - Inverse Document frequency

- Term frequency: no of times a term appears in document / number of terms in document
- Inverse document frequency: log(no of documents/no of documents a term t appeared in)

Term Frequency - Inverse Document frequency

- Term frequency: no of times a term appears in document / number of terms in document
- Inverse document frequency: log(no of documents/no of documents a term t appeared in)
- TF-IDF $=$ Term frequency $\times$ Inverse document frequency

Term Frequency - Inverse Document frequency

- Term frequency: no of times a term appears in document / number of terms in document
- Inverse document frequency: log(no of documents/no of documents a term t appeared in)
- TF-IDF = Term frequency $\times$ Inverse document frequency

Term Frequency X Inverse Document Frequency

$$w_{x,y} = tf_{x,y} \times log(\frac{N}{df_x})$$

Text1: Basic Linux Commands for Data Science
Text2: Essential DVC Commands for Data Science

	basic	commands	data	dvc	essential	for	linux	science
Text 1	0.5	0.35	0.35	0.0	0.0	0.35	0.5	0.35
Text 2	0.0	0.35	0.35	0.5	0.5	0.35	0.0	0.35

Image: https://www.kdnuggets.com/wp-content/uploads/awan_convert_text_documents_tfidf_matrix_tfidfvectorizer_3.png

Advantages and Disadvantages

- Bag of Words and TF IDF do not consider any word ordering as relevant

Advantages and Disadvantages

- Bag of Words and TF IDF do not consider any word ordering as relevant
- Documents are represented as vectors of these words

Advantages and Disadvantages

- Bag of Words and TF IDF do not consider any word ordering as relevant
- Documents are represented as vectors of these words
- Same word can have different representations based on occurrence every time

Advantages and Disadvantages

- Bag of Words and TF IDF do not consider any word ordering as relevant
- Documents are represented as vectors of these words
- Same word can have different representations based on occurrence every time
- Keeping all words in corpus makes the feature vector sparse for machine learning

Advantages and Disadvantages

- Bag of Words and TF IDF do not consider any word ordering as relevant
- Documents are represented as vectors of these words
- Same word can have different representations based on occurrence every time
- Keeping all words in corpus makes the feature vector sparse for machine learning
- Top n words based on frequency or TF IDF scores

Advantages and Disadvantages

- Bag of Words and TF IDF do not consider any word ordering as relevant
- Documents are represented as vectors of these words
- Same word can have different representations based on occurrence every time
- Keeping all words in corpus makes the feature vector sparse for machine learning
- Top n words based on frequency or TF IDF scores
- These techniques are typically used for Sentiment Analysis

Embeddings

- Standardize word representation

Embeddings

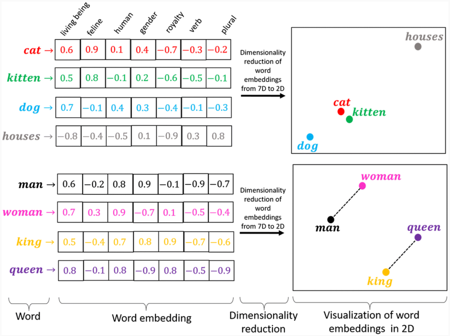
- Standardize word representation
- Word represented as real valued vector in low dimensional space

Embeddings

- Standardize word representation
- Word represented as real valued vector in low dimensional space
- Words with similar meaning close to each other in vector space

Embeddings

- Standardize word representation
- Word represented as real valued vector in low dimensional space
- Words with similar meaning close to each other in vector space



Embedding Methods

- Word2Vec

Embedding Methods

- Word2Vec
    - Vector space from large corpus of text, with each unique word having a vector

Embedding Methods

- Word2Vec
    - Vector space from large corpus of text, with each unique word having a vector
    - Words that share common context are located in close proximity to one another

Embedding Methods

- Word2Vec
  - Vector space from large corpus of text, with each unique word having a vector
  - Words that share common context are located in close proximity to one another
  - Can capture meaning in the document

Embedding Methods

- Word2Vec
  - Vector space from large corpus of text, with each unique word having a vector
  - Words that share common context are located in close proximity to one another
  - Can capture meaning in the document
- GloVe

Embedding Methods

- Word2Vec
  - Vector space from large corpus of text, with each unique word having a vector
  - Words that share common context are located in close proximity to one another
  - Can capture meaning in the document
- GloVe
  - Global vectors for word representation

Embedding Methods

- Word2Vec
  - Vector space from large corpus of text, with each unique word having a vector
  - Words that share common context are located in close proximity to one another
  - Can capture meaning in the document
- GloVe
  - Global vectors for word representation
  - Constructs word context or word co-occurrence matrix

Word Embeddings

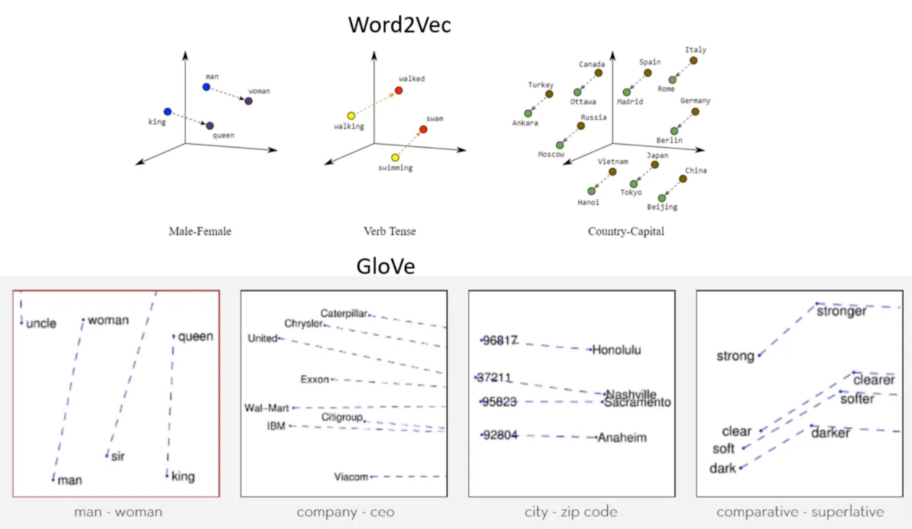


Image: `https://towardsdatascience.com/word-embeddings-for-nlp-5b72991e01d4`

Embedding Methods

- BERT

Embedding Methods

- BERT
  - Bidirectional Encoded Representation from Transformers

Embedding Methods

- BERT
  - Bidirectional Encoded Representation from Transformers
  - A faster way to train language models

Embedding Methods

- BERT
  - Bidirectional Encoded Representation from Transformers
  - A faster way to train language models
  - Embeddings learnt from pre-trained models can be used for other tasks

Embedding Methods

- BERT
  - Bidirectional Encoded Representation from Transformers
  - A faster way to train language models
  - Embeddings learnt from pre-trained models can be used for other tasks
  - Word, as well as Sentence embeddings

Embedding Methods

- BERT
  - Bidirectional Encoded Representation from Transformers
  - A faster way to train language models
  - Embeddings learnt from pre-trained models can be used for other tasks
  - Word, as well as Sentence embeddings

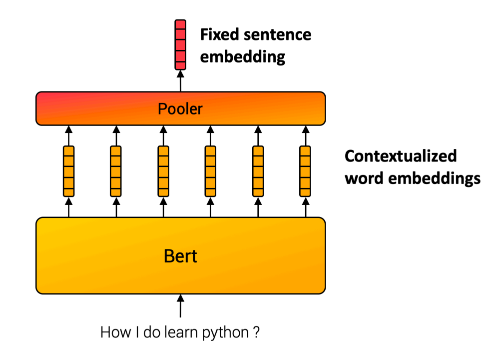


Image: `https://images.app.goo.gl/rVMefeyEKT8jnsBBA`

Long Embeddings

- Large Language Models train their own embeddings

Long Embeddings

- Large Language Models train their own embeddings
- A longer vector representation captures more details

Long Embeddings

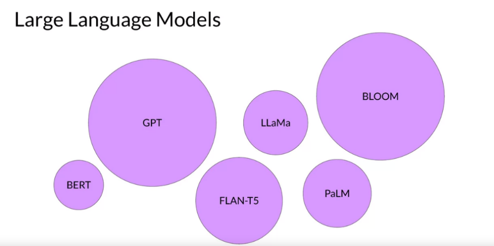
- Large Language Models train their own embeddings
- A longer vector representation captures more details
- Expensive and time taking to use for small tasks

Long Embeddings

- Large Language Models train their own embeddings
- A longer vector representation captures more details
- Expensive and time taking to use for small tasks
- GPT has four types of embedding lengths - Ada (1024), Babbage (2048), Curie (4096), Davinci (12288)

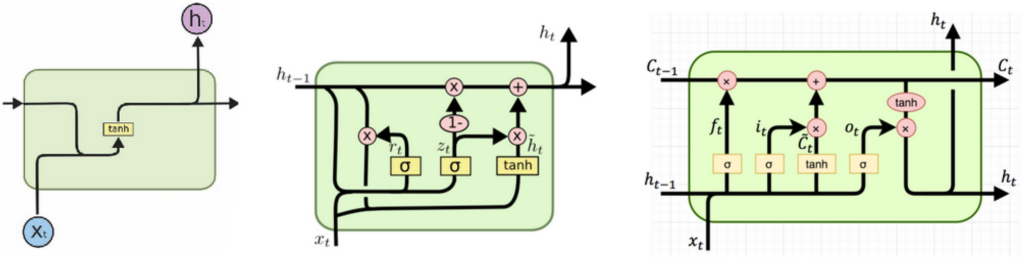
Long Embeddings

- Large Language Models train their own embeddings
- A longer vector representation captures more details
- Expensive and time taking to use for small tasks
- GPT has four types of embedding lengths - Ada (1024), Babbage (2048), Curie (4096), Davinci (12288)



NLP Models

Basic NLP Models



Basic architectures of RNN, GRU and LSTM cells

- RNN: Recurrent Neural Network
- GRU: Gated Recurrent Unit
- LSTM: Long Short-Term Memory

https://medium.com/@saurabh.rathor092/simple-rnn-vs-gru-vs-lstm-difference-lies-in-more-flexible-control-5f33e07b1e57

Transformers

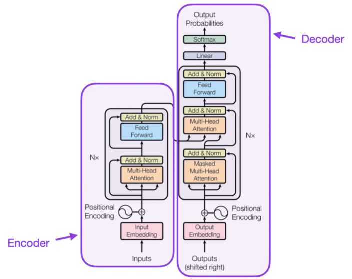


Figure 1: The Transformer - model architecture.

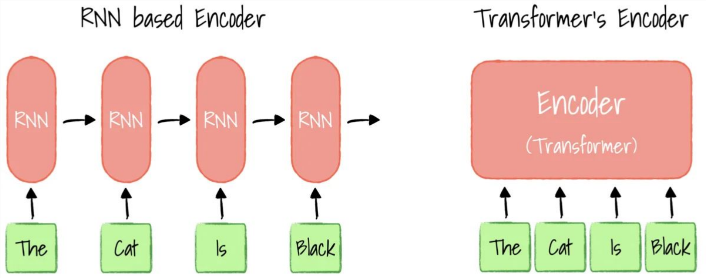


Image: `https://jinglescode.github.io/2020/05/27/illustrated-guide-transformer/`

Large Language Models

Large Language Models

- Large Language Models have billions of parameters

Large Language Models

- Large Language Models have billions of parameters
- Trained on a wide variety of NLP tasks

Large Language Models

- Large Language Models have billions of parameters
- Trained on a wide variety of NLP tasks
- Auto Encoding models for Sentiment Analysis etc.

Large Language Models

- Large Language Models have billions of parameters
- Trained on a wide variety of NLP tasks
- Auto Encoding models for Sentiment Analysis etc.
- Auto Regressive models for Text Generation

Large Language Models

- Large Language Models have billions of parameters
- Trained on a wide variety of NLP tasks
- Auto Encoding models for Sentiment Analysis etc.
- Auto Regressive models for Text Generation
- Sequence to Sequence models for Question Answering

Image: https://medium.com/mlearning-ai/3-llm-architectures-f527ed781ba9

Summary

- Earlier NLP methods required lot of pre-processing for better performance

Summary

- Earlier NLP methods required lot of pre-processing for better performance
- Large Language Models use their own tokenizers and embeddings

Summary

- Earlier NLP methods required lot of pre-processing for better performance
- Large Language Models use their own tokenizers and embeddings
- Minimal pre-processing required to use LLMs for NLP tasks

Summary

- Earlier NLP methods required lot of pre-processing for better performance
- Large Language Models use their own tokenizers and embeddings
- Minimal pre-processing required to use LLMs for NLP tasks
- However, LLMs are computationally heavy - may not be required for simple tasks

Thank you.