Natural Language Processing

By Kaveri Kale(PhD)
IIT Bombay

Introduction

- Natural Language Processing, usually shortened as NLP, is a branch of artificial intelligence that deals with the interaction between computers and humans using the natural language.

- The ultimate objective of NLP is to read, decipher, understand, and make sense of the human languages in a manner that is valuable.

- Most NLP techniques rely on machine learning to derive meaning from human languages.

What is NLP used for?

- Natural Language Processing is the driving force behind the following common applications:
  1. Language translation applications such as Google Translate
  2. Word Processors such as Microsoft Word and Grammarly that employ NLP to check grammatical accuracy of texts.
  3. Interactive Voice Response (IVR) applications used in call centers to respond to certain users' requests.
  4. Personal assistant applications such as OK Google, Siri, Cortana, and Alexa.

What are the techniques used in NLP?

- Syntactic analysis and semantic analysis are the main techniques used to complete Natural Language Processing tasks.

1. Syntax

- Syntax refers to the arrangement of words in a sentence such that they make grammatical sense.

- In NLP, syntactic analysis is used to assess how the natural language aligns with the grammatical rules.

- Computer algorithms are used to apply grammatical rules to a group of words and derive meaning from them.

- Here are some syntax techniques that can be used:
- **Lemmatization**: It entails reducing the various inflected forms of a word into a single form for easy analysis.
- **Morphological segmentation:** It involves dividing words into individual units called morphemes.
- **Word segmentation:** It involves dividing a large piece of continuous text into distinct units.
- **Part-of-speech tagging:** It involves identifying the part of speech for every word.
- **Parsing:** It involves undertaking grammatical analysis for the provided sentence.
- **Sentence breaking:** It involves placing sentence boundaries on a large piece of text.
- **Stemming:** It involves cutting the inflected words to their root form.

2. Symantics

- Semantics refers to the meaning that is conveyed by a text. Semantic analysis is one of the difficult aspects of Natural Language Processing that has not been fully resolved yet.

- It involves applying computer algorithms to understand the meaning and interpretation of words and how sentences are structured.

- Here are some techniques in semantic analysis:
  1. Named entity recognition (NER): It involves determining the parts of a text that can be identified and categorized into preset groups. Examples of such groups include names of people and names of places.
  2. Word sense disambiguation: It involves giving meaning to a word based on the context.
  3. Natural language generation: It involves using databases to derive semantic intentions and convert them into human language.

Word Embeddings

Introduction

- How do we make computers of today perform clustering, classification etc on a text data since we know that they are generally inefficient at handling and processing strings or texts for any fruitful outputs?

- Computer can match two strings and tell you whether they are same or not. But how do we make computers tell you about football or Ronaldo when you search for Messi? How do you make a computer understand that “Apple” in “Apple is a tasty fruit” is a fruit that can be eaten and not a company?

- The answer to the above questions lie in creating a representation for words that capture their **meanings, semantic relationships and the different types of contexts** they are used in.

- And all of these are implemented by using Word Embeddings or numerical representations of texts so that computers may handle them.

- In very simplistic terms, Word Embeddings are the texts converted into numbers and there may be different numerical representations of the same text.

- A Word Embedding format generally tries to map a word using a dictionary to a vector.

One hot vector representation

- Take a look at this example – sentence=” Word Embeddings are Word converted into numbers ”
- A word in this sentence may be “Embeddings” or “numbers ” etc.
- A dictionary may be the list of all unique words in the sentence. So, a dictionary may look like – [‘Word’,’Embeddings’,’are’,’Converted’,’into’,’numbers’]
- A vector representation of a word may be a one-hot encoded vector where 1 stands for the position where the word exists and 0 everywhere else. The vector representation of “numbers” in this format according to the above dictionary is [0,0,0,0,0,1] and of converted is[0,0,0,1,0,0].
- Consider the following similar sentences: Have a good day and Have a great day.
  V = {Have, a, good, great, day}.
  Have = [1,0,0,0,0]`; a=[0,1,0,0,0]` ; good=[0,0,1,0,0]` ; great=[0,0,0,1,0]` ; day=[0,0,0,0,1]`
- We represent each word as a completely independent entity. This means ‘good’ and ‘great’ are as different as ‘day’ and ‘have’, which is not true.

Different types of Word Embeddings

- Frequency based Embedding
  1. Count Vector
  2. TF-IDF Vector
  3. Co-Occurrence Vector

- Prediction based Embedding
  1. CBOW (Continuous Bag of words)
  2. Skip – Gram model

- **Frequency based Embedding**

1. Count Vector

Consider a Corpus C of D documents {d_1,d_2…..d_D} and N unique tokens extracted out of the corpus C. The N tokens will form our dictionary and the size of the Count Vector matrix M will be given by D X N. Each row in the matrix M contains the frequency of tokens in document D_i.

Let us understand this using a simple example.

D_1: He is a lazy boy. She is also lazy.

D_2: Neeraj is a lazy person.

The dictionary created may be a list of unique tokens(words) in the corpus =[‘He’,’She’,’lazy’,’boy’,’Neeraj’,’person’]
Here, D=2, N=6
The count matrix M of size 2 X 6 will be represented as –

	He	She	lazy	boy	Neeraj	person
D1	1	1	2	1	0	0
D2	0	0	1	0	1	1

2. TF-IDF vectorization

- Common words like ‘is’, ‘the’, ‘a’ etc. tend to appear quite frequently in comparison to the words which are important to a document. For example, a document A on Lionel Messi is going to contain more occurences of the word “Messi” in comparison to other documents. But common words like “the” etc. are also going to be present in higher frequency in almost every document.

- Ideally, what we would want is to down weight the common words occurring in almost all documents and give more importance to words that appear in a subset of documents.

- TF-IDF works by penalising these common words by assigning them lower weights while giving importance to words like Messi in a particular document.

- Consider the below sample table which gives the count of terms(tokens/words) in two documents.

Document 1

Term	Count
This	1
is	1
about	2
Messi	4

Document 2

Term	Count
This	1
is	2
about	1
Tf-idf	1

- Now, let us define a few terms related to TF-IDF.

  TF = (Number of times term t appears in a document)/(Number of terms in the document)
  So, TF(This,Document1) = 1/8
  TF(This, Document2)=1/5
  It denotes the contribution of the word to the document i.e words relevant to the document should be frequent. eg: A document about Messi should contain the word ‘Messi’ in large number

- IDF = log(N/n), where, N is the number of documents and n is the number of documents a term t has appeared in.
  So, IDF(This) = log(2/2) = 0
  IDF(Messi) = log(2/1) = 0.301.

- Now, let us compare the TF-IDF for a common word ‘This’ and a word ‘Messi’ which seems to be of relevance to Document 1.

  TF-IDF(This,Document1) = (1/8) * (0) = 0
  TF-IDF(This, Document2) = (1/5) * (0) = 0
  TF-IDF(Messi, Document1) = (4/8)*0.301 = 0.15

- As, you can see for Document1 , TF-IDF method heavily penalises the word ‘This’ but assigns greater weight to ‘Messi’. So, this may be understood as ‘Messi’ is an important word for Document1 from the context of the entire corpus.

3. Co-Occurrence Matrix with a fixed context window

- Similar words tend to occur together and will have similar context for example – Apple is a fruit. Mango is a fruit. Apple and mango tend to have a similar context i.e fruit.

- Co-occurrence – For a given corpus, the co-occurrence of a pair of words say w_1 and w_2 is the number of times they have appeared together in a Context Window.

- Context Window – Context window is specified by a number and the direction. So what does a context window of 2 (around).

- Quick Brown Fox Jump Over The Lazy Dog

  The words are a 2 (around) context window for the word ‘Fox’ and for calculating the co-occurrence only these words will be counted.

- Let us see context window for the word ‘Over’.

  Quick Brown Fox Jump Over The Lazy Dog

- Now, let us take an example corpus to calculate a co-occurrence matrix.
- Corpus = He is not lazy. He is intelligent. He is smart.
-

	He	is	not	lazy	intelligent	smart
He	0	4	2	1	2	1
is	4	0	1	2	2	1
not	2	1	0	1	0	0
lazy	1	2	1	0	0	0
intelligent	2	2	0	0	0	0
smart	1	1	0	0	0	0

Let us understand this co-occurrence matrix by seeing two examples in the table above. Red and the blue box.

- Red box- It is the number of times ‘He’ and ‘is’ have appeared in the context window 2 and it can be seen that the count turns out to be 4. The below table will help you visualise the count.
- while the word ‘lazy’ has never appeared with ‘intelligent’ in the context window and therefore has been assigned 0 in the blue box.

He	is	not	lazy	He	is	intelligent	He	is	smart
He	is	not	lazy	He	is	intelligent	He	is	smart
He	is	not	lazy	He	is	intelligent	He	is	smart
He	is	not	lazy	He	is	intelligent	He	is	smart

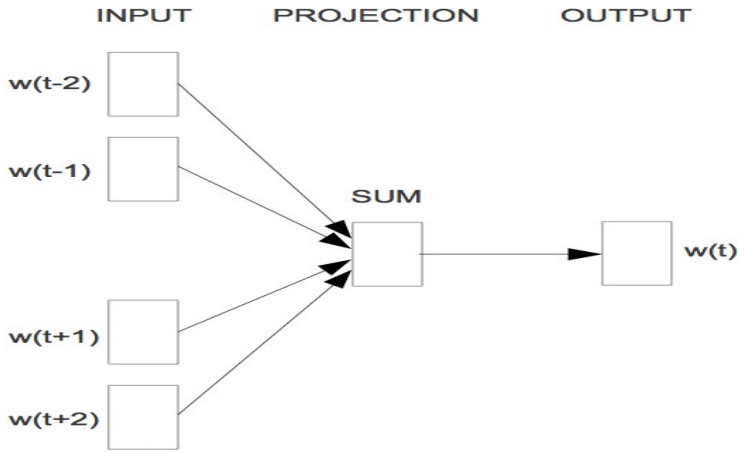
- **Prediction based Vector**

Word2vec

- Instead of computing and storing global information about some huge dataset (which might be billions of sentences), we can try to create a model that will be able to learn one iteration at a time and eventually be able to encode the probability of a word given its context.
- word2vec models are shallow neural network with an input layer, a projection layer and an output layer. It is trained to reconstruct linguistic contexts of words.
-
- Word2vec is a software package that actually includes :
- - 2 algorithms: continuous bag-of-words (CBOW) and skip-gram.
- CBOW aims to predict a center word from the surrounding context in terms of word vectors. Skip-gram does the opposite, and predicts the distribution (probability) of context words from a center word.

1. Continuous bag-of-words (CBOW) Model

- This algorithm use both the n words before and after the target word w_t to predict it as depicted in Figure.
- They call this continuous bag-of-words (CBOW), as it uses continuous representations whose order is of no importance.



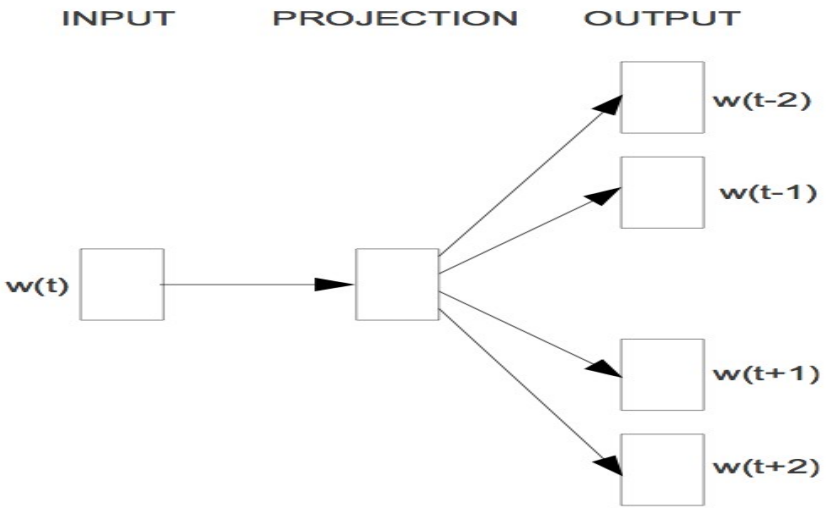
- The objective function of CBOW in turn is only slightly different than the language model one:

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log p(w_t | w_{t-1}, \ldots, w_{t+n})$$

- Instead of feeding n previous words into the model, the model receives a window of n words around the target word wt at each time step t.

2. Skip-gram Model

- While CBOW can be seen as a precognitive language model, skip-gram turns the language model objective on its head: Instead of using the surrounding words to predict the centre word as with CBOW, skip-gram uses the centre word to predict the surrounding words as can be seen in Figure.



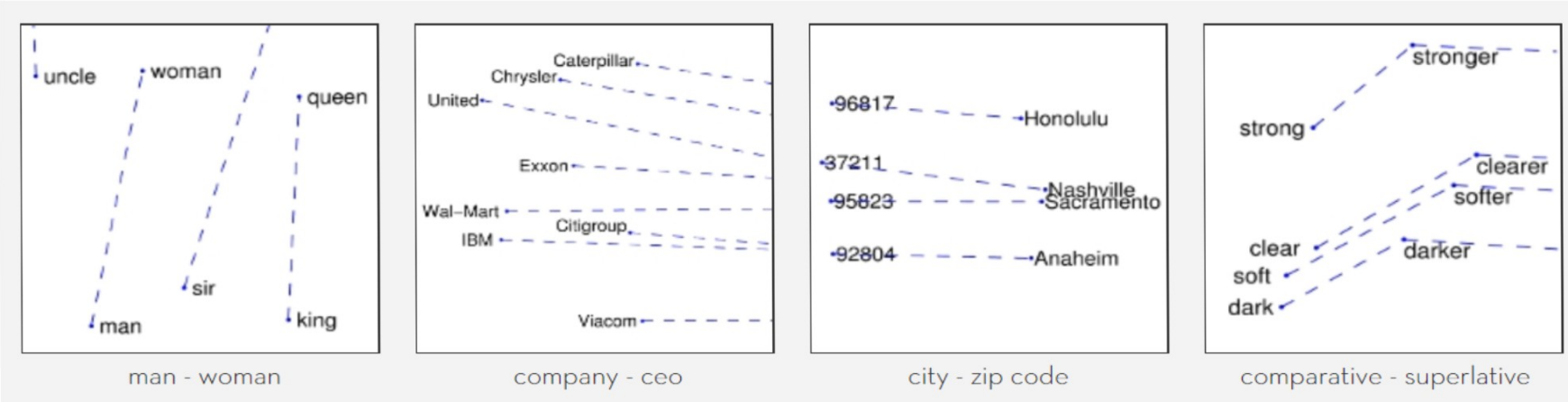
- The skip-gram objective thus sums the log probabilities of the surrounding n words to the left and to the right of the target word wt to produce the following objective:

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T}\sum_{-n\le j\le n,\neq 0}\log p(w_{t+j}|w_t)$$

GloVe: Global Vector for word representation

- It is called Global Vectors as the global corpus statistics are captured directly by the model.
- It leverages both
  1. Global matrix factorization methods like latent semantic analysis (LSA) for generating low-dimensional word representations
  2. Local context window methods such as the skip-gram model of Mikolov et al
- Methods like skip-gram perform better on the analogy task, but poorly utilize the statistics of the corpus as they are not trained on global co-occurrence counts. GloVe uses a specific weighted least squares model to train on global word co-occurrence counts to make efficient use of statistics.
- Consider two words i=ice and j=steam in the context of Thermodynamics domain. The relationship of these words can be examined by studying the ratio of their co-occurrence probabilities with various probe words k.
- Ratio of co-occurrence probabilities is: $\frac{P_{ik}}{P_{jk}}$
-
-
- Probe words like solid related to ice but not to steam will have large value for the ratio

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96



- It is the gender that distinguishes man from woman, similar to word pairs, such as king and queen or brother and sister. To state this observation mathematically, we might expect that the vector differences man: woman, king: queen, and brother: sister might all be roughly equal. This property and other interesting patterns can be observed in the above set of visualizations using GloVe.

References

- https://ruder.io/word-embeddings-softmax/index.html
- https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/
- https://towardsdatascience.com/word-embeddings-for-nlp-5b72991e01d4