Natural Language Processing(NLP) (IT401)

Prepared by

Mr. Umesh B. Sangule
Assistant Professor

Department of Information Technology

Unit-II

LANGUAGE MODELING AND EMBEDDING

Course Objectives : *To introduce the statistics for NLP and language model.*

Course Outcome(CO2) : *Understand the statistics for NLP and Language modeling.*

Content

- Probabilistic language modeling,
- N-gram Models,
- Word Embedding
- Topic Modeling

PROBABILISTIC LANGUAGE MODELING

- A popular idea in computational linguistics is to create a probabilistic model of language.

- Such a model assigns a probability to every sentence in English in such a way that more likely sentences (in some sense) get higher probability.

- If you are unsure between two possible sentences, pick the higher probability one.

PROBABILISTIC LANGUAGE MODELING

- A "perfect" language model is only attainable with true intelligence.
- However, approximate language models are often easy to create and good enough for many applications.
- Some language model examples are Unigram, Bigram, Tag Bigram, Maximum entropy, Stochastic Context Free,

PROBABILISTIC LANGUAGE MODELING

- Some Models:

Unigram: Words generated one at a time, drawn from a fixed distribution.

Bigram: Probability of word depends on previous word.

Tag Bigram: Probability of part of speech depends on previous part of speech, probability of word depends on part of speech.

Stochastic Context Free: Words generated by a context-free grammar augmented with probabilitistic rewrite rules.

MARKOV MODELS

- A Markov model is a stochastic method for randomly changing systems that possess the Markov property.

- This means that, at any given time, the next state is only dependent on the current state and is independent of anything in the past.

- Two commonly applied types of Markov model are used when the system being represented is autonomous that is, when the system is not influenced by an external agent.

MARKOV MODELS

(i) ***Markov Chains***:

- These are the simplest type of Markov model and are used to represent systems where all states are observable.

- Markov chains show all possible states and between states, they show the transition rate, which is the probability of moving from one state to another per unit of time.

- Applications of this type of model include prediction of market crashes, speech recognition and search engine algorithms.

MARKOV MODELS

*(ii) **Hidden Markov Models**(HMM):*

- These are used to represent systems with some unobservable states.

- In addition to showing states and transition rates, hidden Markov models also represent observations and observation likelihoods for each state.

- Hidden Markov models are used for a range of applications, including thermodynamics, finance and pattern recognition

MARKOV MODELS

Markov analysis is used in many domains, including the following:

- Markov chains are used for several business applications, including predicting customer brand switching for marketing,

- Predicting how long people will remain in their jobs for human resources,

- Predicting time to failure of a machine in manufacturing and forecasting the future price of a stock in finance.

MARKOV MODELS

- Markov analysis is also used in natural language processing (NLP) and in machine learning.
- For NLP, a Markov chain can be used to generate a sequence of words that form a complete sentence or a hidden Markov model can be used for named entity recognition and tagging parts of speech.
- For machine learning, Markov decision processes are used to represent reward in reinforcement learning.

MARKOV MODELS

- A recent example of the use of Markov analysis in healthcare was in Kuwait.
- A continuous-time Markov chain model was used to determine the optimal timing and duration of a full COVID-19 lockdown in the country, minimizing both new infections and hospitalizations.
- The model suggested that a 90-day lockdown beginning 10 days before the epidemic peak was optimal

GRAPH-BASED MODELS N-GRAM MODELS

Simple N-gram Models:

- Language modeling is the way of determining the probability of any sequence of words.
- Language modeling is used in a wide variety of applications such as Speech Recognition, Spam filtering, etc.
- In fact, language modeling is the key aim behind the implementation of many state-of- the-art Natural Language Processing models..

GRAPH-BASED MODELS N-GRAM MODELS

Methods of Language Modelings:

Two types of Language Modelings-

(i) Statistical Language Modelings:

- Statistical Language Modeling or Language Modeling, is the development of probabilistic models that are able to predict the next word in the sequence given the words that precede.

- Examples such as N-gram language modeling.

GRAPH-BASED MODELS N-GRAM MODELS

(ii) Neural Language Modelings:

- Neural network methods are achieving better results than classical methods both on standalone language models.

- When models are incorporated into larger models on challenging tasks like speech recognition and machine translation.

- A way of performing a neural language model is through word embeddings.

GRAPH-BASED MODELS N-GRAM MODELS

***N-gram*:**

- N-gram can be defined as the contiguous sequence of n items from a given sample of text or speech.
- The items can be letters, words or base pairs according to the application.
- The N-grams typically are collected from a text or speech corpus (A long text dataset).

GRAPH-BASED MODELS N-GRAM MODELS

***N-gram Language Model*:**

- An N-gram language model predicts the probability of a given N-gram within any sequence of words in the language.

- A good N-gram model can predict the next word in the sentence

- N-gram is a sequence of the N-words in the modeling of NLP.

GRAPH-BASED MODELS N-GRAM MODELS

- Consider an example of the statement for modeling. "I love reading history books and watching documentaries".
- In one-gram or unigram, there is a one-word sequence.
- As for the above statement, in one gram it can be "I", "love", "history", "books", "and", "watching", "documentaries".
- In two-gram or the bi-gram, there is the two-word sequence i.e. "I love", "love reading"

GRAPH-BASED MODELS N-GRAM MODELS

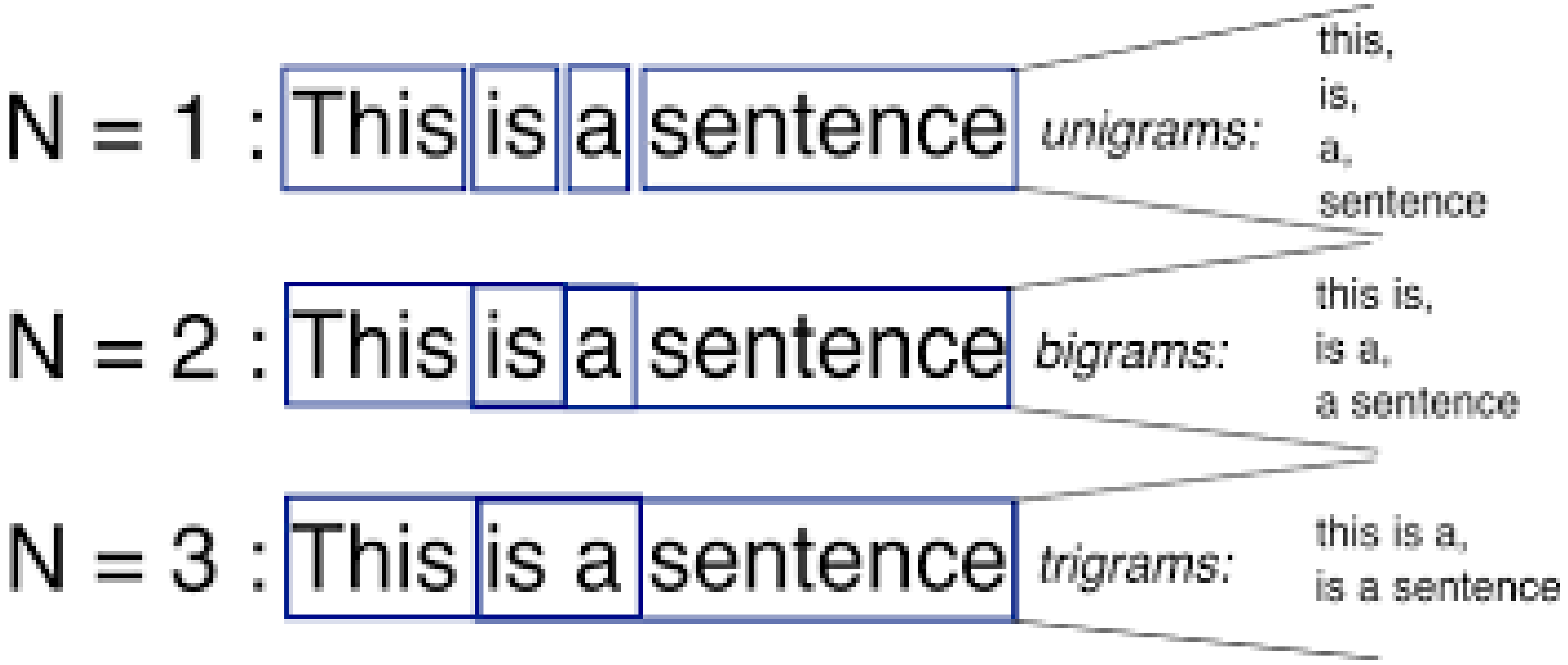
- In the three-gram or the tri-gram, there are the three words sequences i.e. "I love reading", "history books," or "and watching documentaries.

- The illustration of the N-gram modeling i.e. for N=1,2,3 is given below

This is Big Data AI Book

Uni-Gram	This	Is	Big	Data	AI	Book
Bi-Gram	This is	Is Big	Big Data	Data AI	AI Book	
Tri-Gram	This is Big	Is Big Data	Big Data AI	Data AI Book		

GRAPH-BASED MODELS N-GRAM MODELS



GRAPH-BASED MODELS N-GRAM MODELS

- For N-1 words, the N-gram modeling predicts most occurred words that can follow the sequences.
- The model is the probabilistic language model which is trained on the collection of the text.
- This model is useful in applications i.e. speech recognition and machine translations.
- A simple model has some limitations that can be improved by smoothing, interpolations and back off. So, the N-gram about finding probability language model is about finding distributions over the sequences of the word.

GRAPH-BASED MODELS N-GRAM MODELS

- Consider the sentences i.e. "There was heavy rain" and "There was heavy flood". By using experience, it can be said that the first statement is good.

- The N-gram language model tells that the "heavy rain" occurs more frequently than the "heavy flood".

- So, the first statement is more likely to occur and it will be then selected by this model.

- ***"In the one-gram model, the model usually relies on that which word occurs often without pondering the previous words. In 2-gram, only the previous word is considered for predicting the current word. In 3-gram, two previous words are considered."***

WORD EMBEDDINGS/ VECTOR SEMANTICS

Bag of Word (BoW):

- In NLP the input to various algorithms for different applications is in text form.
- But these machine learning algorithms cannot work on the raw text.
- The text must be converted to numbers for further processing.
- More specifically the vectors of these numbers representing text are generated.

WORD EMBEDDINGS/ VECTOR SEMANTICS

***Bag of Word (BoW)*:**

- Bag Of Words (BOW) representation is used to convert text into fixed length vectors.
- BOW model takes into account the frequency of occurrence of words in input text document.
- It focuses on two things:
  1. Vocabulary of known words.
  2. Measure of presence of these known words.

WORD EMBEDDINGS/ VECTOR SEMANTICS

Bag of Word (BoW):

- The word 'Bag' in bag of words is analogous to the bag of items.
- When we fill the bag the order of the items is discarded, similarly in bag of words model represents unordered set of words irrespective of their position in the document.
- It only considers the frequency of words in the text.

WORD EMBEDDINGS/ VECTOR SEMANTICS

Bag of Word (BoW):

- Let's consider the following example to understand how bag of words model works stepwise.
- Step 1: Step 1 includes collection of data. Consider each of the following sentences as text documents.

The dog sat
The dog sat in the hat
The dog with the hat

WORD EMBEDDINGS/ VECTOR SEMANTICS

***Bag of Word (BoW)*:**

- Step 2 includes designing the vocabulary. All the words present in document set are listed.
- In out example the words formed are: the, dog, sat, in, the, hat, with.
- Step 3 includes computation of the frequency of the occurrence of words in the document.

WORD EMBEDDINGS/ VECTOR SEMANTICS

Bag of Word (BoW):

- With the 6 distinct words and their frequency of occurrence in the document, now we can write fixed length vector for each document as follows:

The dog sat- [1, 1, 1, 0, 0, 0]

The dog sat in the hat → [2, 1, 1, 1, 1, 0]

The dog with the hat → [2, 1, 0, 0, 1, 1]

WORD EMBEDDINGS/ VECTOR SEMANTICS

Bag of Word (BoW):

- The drawback of BOW model is, we loose the context of the document.

- BOW model only tells what words will appear in the document with them Frequency but it doesn't tell where they occurred.

- In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity.

WORD EMBEDDINGS/ VECTOR SEMANTICS

Bag of Word (BoW):

- ***Disadvantages of using bag of words model*** :

- The model ignores the location information of the word. The location information is a piece of very important information in the text

- Bag of word models doesn't respect the semantics of the word.
  For example, words 'soccer' and 'football' are often used in the same context.

- The range of vocabulary is a big issue faced by the bag-of-words model.

WORD EMBEDDINGS/ VECTOR SEMANTICS

***Term Frequency Inverse Document Frequency (TF-IDF)*:**

- It can be defined as the calculation of how relevant a word in a series or corpus to text,

- ***Key concepts in TF / IDF*** :
- ***Term Frequency(TF):***

- In document d, the frequency represents the number of instances of a given word t.
- Therefore, we can see that it becomes more relevant when a word appears in the text, which is rational.

WORD EMBEDDINGS/ VECTOR SEMANTICS

***(TF-IDF)*:**

- ***Term Frequency*(TF):**

  tf(t,d)= count of t in d/number of words in d

- ***Document Frequency*(DF):):**

- This tests the meaning of the text, which is very similar to TF, in the whole corpus collection.

- The only difference is that in document d, TF is the frequency counter for a term t, while df is the number of occurrences in the document set N of the term t

WORD EMBEDDINGS/ VECTOR SEMANTICS

(TF-IDF):

- ***Inverse Document Frequency(IDF):):***
- Mainly, it tests how relevant the word is,
- Since tf considers all terms equally significant, it is therefore not only possible to use the term frequencies to measure the weight of the term in the paper.
- First, find the document frequency of a term t by counting the number of documents containing the term:

<u>WORD EMBEDDINGS/ VECTOR SEMANTICS</u>

<u>(TF-IDF)</u>:

- ***<u>Inverse Document Frequency</u>(IDF):):***
- df(t) = N(t)

  where,

  df(t) = Document frequency of a term t

  N(t) = Number of documents containing the term t

- Term frequency is the number of instances of a term in a single document only, although the frequency of the document is the number of separate documents in which the term appears, it depends on the entire corpus.

WORD EMBEDDINGS/ VECTOR SEMANTICS

(TF-IDF):

- ***Inverse Document Frequency(IDF):):***
- Now let's look at the definition of the frequency of the inverse paper. The IDF of the word is the number of documents in the corpus separated by the frequency of the text.

$$idf(t) = N/df(t) = N/N(t)$$

- The more common word is supposed to be considered less significant, but the element (most definite integers) seems too harsh. We then take the logarithm (with base 2) of the inverse frequency of the paper. So the if of the term t becomes: $idf(t)= \log(N/ df(t))$

WORD EMBEDDINGS/ VECTOR SEMANTICS

***(TF-IDF)*:**

- Tf-idf is one of the best metrics to determine how significant a term is to a text in a series or a corpus.

- tf-idf is a weighting system that assigns a weight to each word in a document based on its term frequency (tf) and the reciprocal document frequency (tf) (idf).

- The words with higher scores of weight are deemed to be more significant. Hence we get

$$\text{tf-idf}(t, d) = \text{tf}(t, d) * \text{idf}(t)$$

WORD EMBEDDINGS/ VECTOR SEMANTICS

- ***Applications of TFIDF*:**
- Document classification - Helps in classifying the type and genre of a document.
- Topic modelling - It helps in predicting the topic for a corpus.
- Information retrieval system - To extract the important information out of a corpus.
- Stop word filtering - Helps in removing the unnecessary words out of a text body.

WORD EMBEDDINGS/ VECTOR SEMANTICS

***Word Embedding/Vector Semantics*:**

- As described in the BOW model we have seen how a word in a document can be represented in numerical form as a vector representation.

- In other words this numerical representation of words in called as word embedding.

- Word embedding are classified in two types:

  1. Frequency based embedding
  2. Prediction based embedding

WORD EMBEDDINGS/ VECTOR SEMANTICS

1. Simple frequency based embeddings:

One-hot encoding:

- Let's start by looking at the simplest way of converting text into a vector.
- We could have a vector of the size of our vocabulary where each element in the vector corresponds to a word from the vocabulary.
- The encoding of a given word would then simply be the vector in which the corresponding element is set to 1 and all other elements are 0.
- Simple technique of creating numerical representations is called One-Hot Encoding.

WORD EMBEDDINGS/ VECTOR SEMANTICS

1. Simple frequency based embeddings:

One-hot encoding:

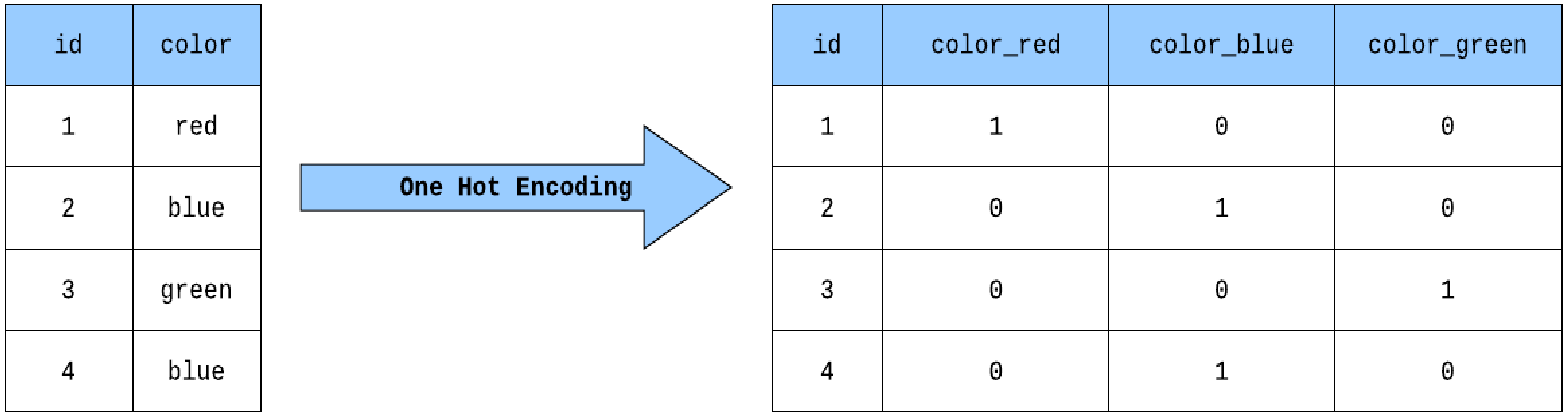
- This technique is also not able to capture any semantic relationships or context information.

- If we have a vocabulary of ten million words, it would mean that we would need vectors of size 10M, where each word would be represented by a single 1 and all other 9,999,999 elements would be set to 0

WORD EMBEDDINGS/ VECTOR SEMANTICS

1. Simple frequency based embeddings:

One-hot encoding:



id	color
1	red
2	blue
3	green
4	blue

id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

WORD EMBEDDINGS/ VECTOR SEMANTICS

2. Prediction based embeddings:

- Prediction based embedding is one of the most sought word embedding technique in NLP.

- These methods were prediction-based as they give the probabilities to the words.

- They proved to be state of the art for tasks like word analogies and word similarities.

WORD EMBEDDINGS/ VECTOR SEMANTICS

2. Prediction based embeddings:

- Prediction based embedding is one of the most sought word embedding technique in NLP.

- Following are the two algorithms in prediction based embedding,.

  1. Word2Vec
  2. Doc2Vec

WORD EMBEDDINGS/ VECTOR SEMANTICS

1. Word2Vec:

- The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text.

- Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence.

- As the name implies, word2vec represents each distinct word with a particular list of numbers called a vector.

WORD EMBEDDINGS/ VECTOR SEMANTICS

1. Word2Vec:

- The vectors are chosen carefully such that they capture the semantic and syntactic qualities of words; as such, a simple mathematical function (cosine similarity) can indicate the level of semantic similarity between the words represented by those vectors.

- Word2vec can utilize either of two model architectures to produce these distributed representations of words: Continuous bag-of-words (CBOW) or continuous skip- gram.

WORD EMBEDDINGS/ VECTOR SEMANTICS

1. Word2Vec:

- In both architectures, word2vec considers both individual words and a sliding window of context words surrounding individual words as it iterates over the entire corpus.

- In the continuous bag-of-words(CBOW) architecture, the model predicts the current word from the window of surrounding context words.

- The order of context words does not influence prediction (bag-of-words assumption).

WORD EMBEDDINGS/ VECTOR SEMANTICS

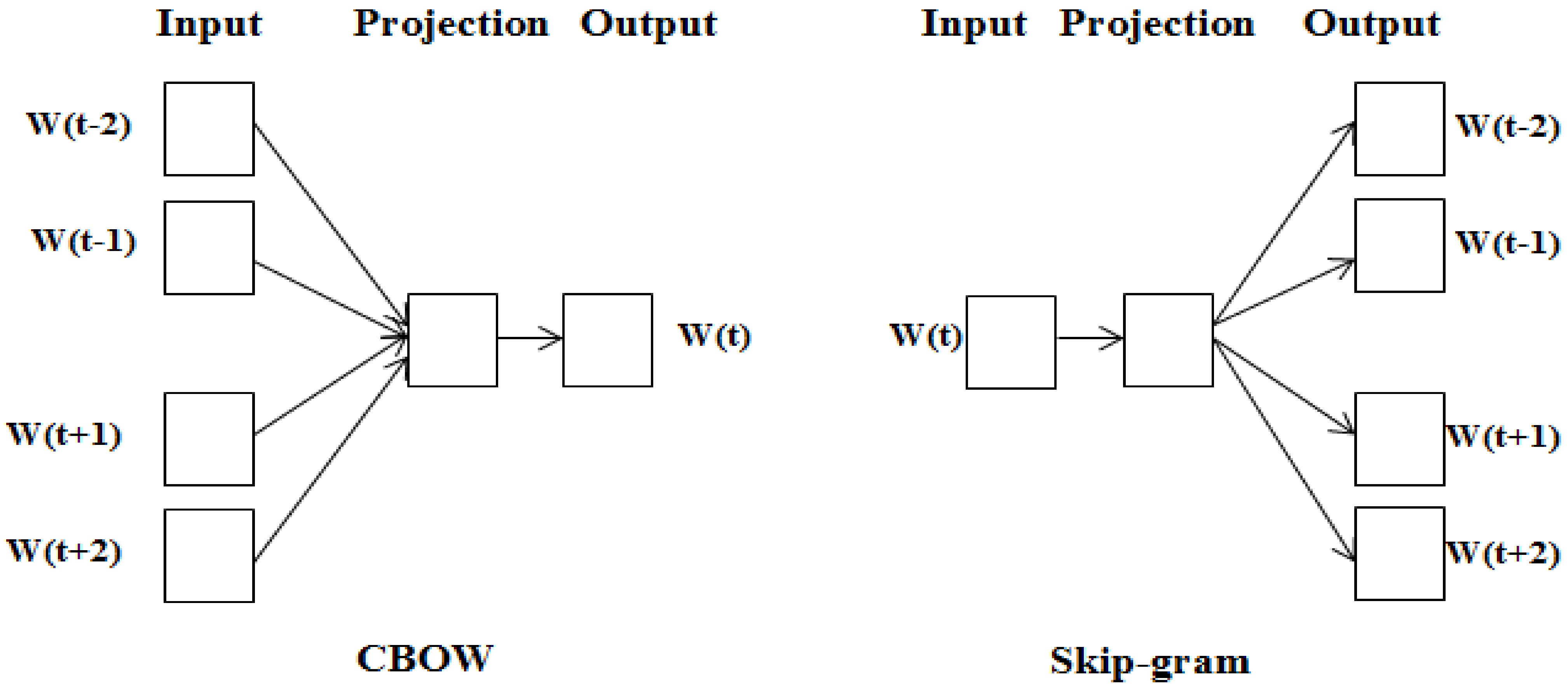
1. Word2Vec:

- The skip-gram architecture weighs nearby context words more heavily than distant context words. According to the authors' note, CBOW is faster while skip-gram does a better job for infrequent words.

- After the model has trained, the learned word embeddings are positioned in the vector space such that words that share common contexts in the corpus - that is words that are semantically and syntactically similar are located close to one another in the space. More dissimilar words are located farther from one another in the space.

WORD EMBEDDINGS/ VECTOR SEMANTICS

1. Word2Vec:



WORD EMBEDDINGS/ VECTOR SEMANTICS

2. Doc2Vec:

- Doc2Vec is an extension to word2vec. doc2vec, generates distributed representations of variable-length pieces of texts, such as sentences, paragraphs, or entire documents.

- Doc2Vec estimates the distributed representations of documents much like how word2vec estimates representations of words.

- Doc2Vec can estimate the semantic embeddings for speakers or speaker attributes, groups and periods of time.

WORD EMBEDDINGS/ VECTOR SEMANTICS

2. Doc2Vec:

- For example, doc2vec has been used to estimate the political positions of political parties in various Congresses and Parliaments in the US. and U.K., respectively and various governmental institutions,

- Doc2Vec utilizes either of two model architectures, both of which are allegories to the architectures used in word2vec.

- The first, Distributed Memory Model of Paragraph Vectors (PV-DM) identical to CBOW other than it also provides a unique document identifier as a piece of additional context.

WORD EMBEDDINGS/ VECTOR SEMANTICS

Bidirectional Encoder Representations from Transformers (BERT):

- BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context.

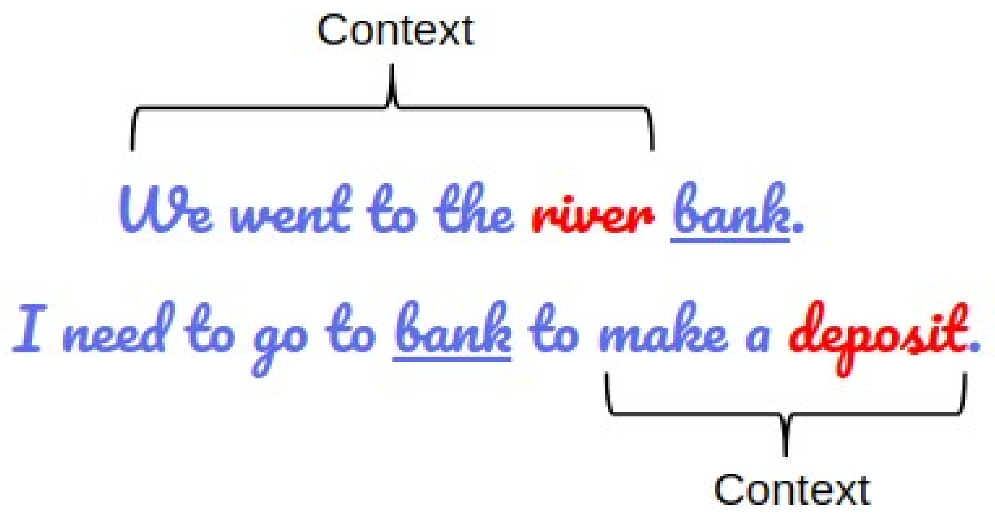
- As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks.

- BERT is pre-trained on a large corpus of unlabelled text including the entire Wikipedia(that’s 2,500 million words!) and Book Corpus (800 million words).

WORD EMBEDDINGS/ VECTOR SEMANTICS

Bidirectional Encoder Representations from Transformers (BERT):

- BERT is a “deeply bidirectional” model. Bidirectional means that BERT learns information from both the left and the right side of a token’s context during the training phase.


Context
We went to the river bank.
I need to go to bank to make a deposit.
Context

Bidirectional Encoder Representations from Transformers (BERT):

How does it work-

- BERT relies on a Transformer (the attention mechanism that learns contextual relationships between words in a text).

- A basic Transformer consists of an encoder to read the text input and a decoder to produce a prediction for the task.

- Since BERT's goal is to generate a language representation model, it only needs the encoder part.
- The input to the encoder for BERT is a sequence of tokens, which are first converted into vectors and then processed in the neural network.

Bidirectional Encoder Representations from Transformers (BERT):

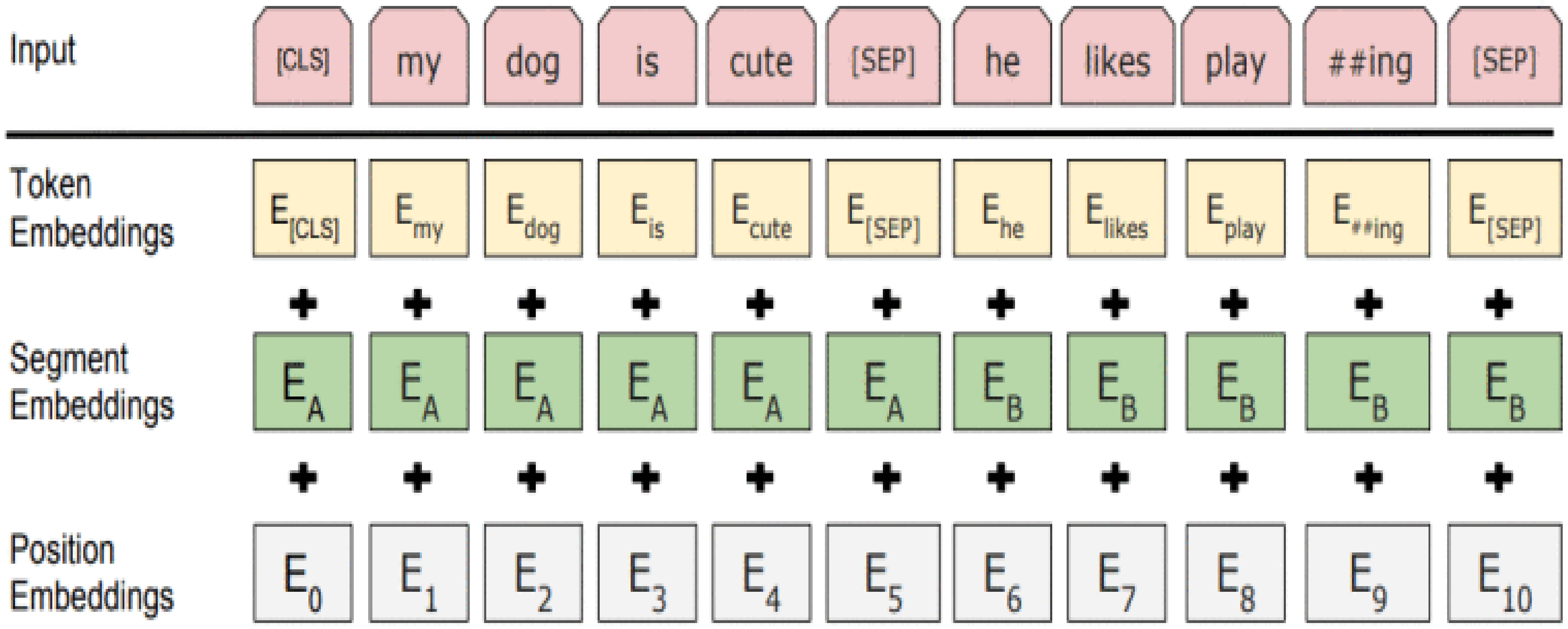
How does it work-

- But before processing can start, BERT needs the input to be massaged and decorated with some extra metadata:

- ***Token embeddings***: A [CLS] token is added to the input word tokens at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
- ***Segment embeddings***: A marker indicating Sentence A or Sentence B is added to each token. This allows the encoder to distinguish between sentences.
- ***Positional embeddings***: A positional embedding is added to each token to indicate its position in the sentence.

Bidirectional Encoder Representations from Transformers (BERT):

How does it work-


Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Bidirectional Encoder Representations from Transformers (BERT):

Applications of BERT:

- Sentiment Analysis.
- Language Translation.
- Question Answering.
- Google Search.
- Text Summarization.
- Matching and Retrieving text.
- Highlighting paragraphs.

GLOVE

GLOVE(Global Vectors for Word Representation):

- Glove is an alternative method to develop word embeddings.
- It is purely based on matrix factorization techniques on the "word-context matrix".
- Normally, we can scan our corpus in the following manner: for every term, we look for context terms within the area defined by a window size before the term and a window size after the term. And hence, we give less amount of weight to more distant words.

GLOVE

GLOVE(Global Vectors for Word Representation):

Applications:

- Analysing survey responses .
- Analysing verbatim comments.
- Music/Video recommendation system.

WORD EMBEDDINGS/ VECTOR SEMANTICS

Latent Semantic Analysis(LSA):

- Latent Semantic Analysis (LSA) is a technique in natural language processing, in particular distributional semantics, of analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms.

- LSA assumes that words that are close in meaning will occur in similar pieces of text (the distributional hypothesis).

WORD EMBEDDINGS/ VECTOR SEMANTICS

Latent Semantic Analysis(LSA):

- ***Application of LSA***
- Given a query of terms, translate it into the low-dimensional space and find matching documents (information retrieval).

- Find the best similarity between small groups of terms, in a semantic way (i.e. in a context of a knowledge corpus), as for example in multiple choice questions MCQ answering model.

WORD EMBEDDINGS/ VECTOR SEMANTICS

Latent Semantic Analysis(LSA):

- ***Application of LSA***
- Compare the documents in the low-dimensional space (data clustering, document classification).
- Find similar documents across languages, after analyzing a base set of translated documents (cross-language information retrieval).
- Find relations between terms (synonymy and polysemy).

WORD EMBEDDINGS/ VECTOR SEMANTICS

Latent Semantic Analysis(LSA):

- ***Application of LSA***
- Given a query of terms, translate it into the low-dimensional space and find matching documents (information retrieval).

- Find the best similarity between small groups of terms, in a semantic way (i.e. in a context of a knowledge corpus), as for example in multiple choice questions MCQ answering model.

THANK YOU