Artificial Intelligence (Machine Learning & Deep Learning) [Course]

Week 9 – Deep Learning – Word Embeddings

[See examples / code in GitHub code repository]

It is not about Theory, it is 20% Theory and 80% Practical – Technical/Development/Programming [Mostly Python based]

DL | What is - Word Embedding?

Word Embedding is a language modeling technique for mapping words to vectors of real numbers. It represents words or phrases in vector space with several dimensions. Word embeddings can be generated using various methods like neural networks, co-occurrence matrices, probabilistic models, etc.

Word Embeddings are numeric representations of words in a lower-dimensional space, capturing semantic and syntactic information.

Need for Word Embedding?

- To reduce dimensionality
- To use a word to predict the words around it.
- Inter-word semantics must be captured.

The methods such as Bag of Words (BOW), CountVectorizer and TFIDF rely on the word count in a sentence but do not save any syntactical or semantic information.

Reference:
https://www.geeksforgeeks.org/word-embeddings-in-nlp/
https://d2l.ai/chapter_natural-language-processing-pretraining/word2vec.html
https://www.tensorflow.org/text/tutorials/word2vec
https://wiki.pathmind.com/word2vec
https://www.analyticsvidhya.com/blog/2021/07/word2vec-for-word-embeddings-a-beginners-guide/

DL | What - Approaches for Text Representation?

1.1. One-Hot Encoding [Traditional Approach]

One-hot encoding is a simple method for representing words in natural language processing (NLP). In this encoding scheme, each word in the vocabulary is represented as a unique vector, where the dimensionality of the vector is equal to the size of the vocabulary. The vector has all elements set to 0, except for the element corresponding to the index of the word in the vocabulary, which is set to 1.

While one-hot encoding is a simple and intuitive method for representing words in NLP, it has several disadvantages, which may limit its effectiveness in certain applications.

- ❑ One-hot encoding results in high-dimensional vectors, making it computationally expensive and memory-intensive, especially with large vocabularies.
- ❑ It does not capture semantic relationships between words; each word is treated as an isolated entity without considering its meaning or context.
- ❑ It is restricted to the vocabulary seen during training, making it unsuitable for handling out-of-vocabulary words.

Exercise/Code Example:

Reference:
https://www.geeksforgeeks.org/word-embeddings-in-nlp/
https://d2l.ai/chapter_natural-language-processing-pretraining/word2vec.html
https://www.tensorflow.org/text/tutorials/word2vec
https://wiki.pathmind.com/word2vec
https://www.analyticsvidhya.com/blog/2021/07/word2vec-for-word-embeddings-a-beginners-guide/

DL | What - Approaches for Text Representation?

1.2 Bag of Word (Bow) [Traditional Approach]

Bag-of-Words (BoW) is a text representation technique that represents a document as an unordered set of words and their respective frequencies. It discards the word order and captures the frequency of each word in the document, creating a vector representation.

While BoW is a simple and interpretable representation, below disadvantages highlight its limitations in capturing certain aspects of language structure and semantics:

- ❑ BoW ignores the order of words in the document, leading to a loss of sequential information and context making it less effective for tasks where word order is crucial, such as in natural language understanding.
- ❑ BoW representations are often sparse, with many elements being zero resulting in increased memory requirements and computational inefficiency, especially when dealing with large datasets.

Exercise/Code Example:

Reference:
https://www.geeksforgeeks.org/word-embeddings-in-nlp/
https://d2l.ai/chapter_natural-language-processing-pretraining/word2vec.html
https://www.tensorflow.org/text/tutorials/word2vec
https://wiki.pathmind.com/word2vec
https://www.analyticsvidhya.com/blog/2021/07/word2vec-for-word-embeddings-a-beginners-guide/

DL | What - Approaches for Text Representation?

1.3 TF-IDF (Term Frequency-Inverse Document Frequency) [Traditional Approach]

TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure used in natural language processing and information retrieval **to evaluate the importance of a word in a document relative to a collection of documents (corpus).**

$$\text{TF}(t, d) = \frac{\text{Number of times term t appears in document d}}{\text{Total number of terms in document d}}$$

Term Frequency (TF)

Limitations of TF Alone:

- TF does not account for the global importance of a term across the entire corpus.
- Common words like **"the"** or **"and"** may have high TF scores but are not meaningful in distinguishing documents.

$$\text{IDF}(t, D) = \log \frac{\text{Total number of documents in corpus D}}{\text{Number of documents containing term t}}$$

Inverse Document Frequency (IDF)

Limitations of IDF Alone:

- IDF does not consider how often a term appears within a specific document.
- A term might be rare across the corpus (high IDF) but irrelevant in a specific document (low TF).

sklearn's online documentation, it uses the below method to calculate idf of a term in a document.
$\text{idf}(t) = \log_e [(1+n) / (1 + \text{df}(t))] + 1$ (default i:e smooth_idf = True)
where
n = Total number of documents available
t = term for which idf value has to be calculated
df(t) = Number of documents in which the term t appears

Exercise/Code Example:

https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/
https://www.analyticsvidhya.com/blog/2021/11/how-sklearns-tfidfvectorizer-calculates-tf-idf-values/

DL | What - Approaches for Text Representation?

2.1 Word2Vec - Neural Approach

Word2Vec is a neural approach for generating word embeddings. It belongs to the family of neural word embedding techniques and specifically falls under the category of distributed representation models. It is a popular technique in natural language processing (NLP) that is used to represent words as continuous vector spaces. Developed by a team at Google, Word2Vec aims to capture the semantic relationships between words by mapping them to high-dimensional vectors. The underlying idea is that words with similar meanings should have similar vector representations.

Why we need Word2Vec?

•**Semantic Representations:** Word2Vec records the connections between words semantically. Words are represented in the vector space so that similar words are near to one another. This enables the model to interpret words according to their context within a particular corpus.

•**Distributional Semantics:** The foundation of Word2Vec is the distributional hypothesis, which holds that words with similar meanings are more likely to occur in similar contexts. Word2Vec generates vector representations that reflect semantic similarities by learning from the distributional patterns of words in a large corpus.

•**Vector Arithmetic:** Word2Vec generates vector representations that have intriguing algebraic characteristics. Vector arithmetic, for instance, can be used to record word relationships. One well-known example is that the vector representation of "queen" could resemble the vector representation of "king" less "man" plus "woman."

•**Efficiency:** Word2Vec's high computational efficiency makes training on big datasets possible. Learning high-dimensional vector representations for a large vocabulary requires this efficiency.

•**Transfer Learning:** A variety of natural language processing tasks can be initiated with pre-trained Word2Vec models. Time and resources can be saved by fine-tuning the embeddings discovered on a sizable dataset for particular uses.

•**Applications:** Word2Vec embeddings have shown promise in a number of natural language processing (NLP) applications, such as machine translation, text classification, sentiment analysis, and information retrieval. These applications are successful in part because of their capacity to capture semantic relationships.

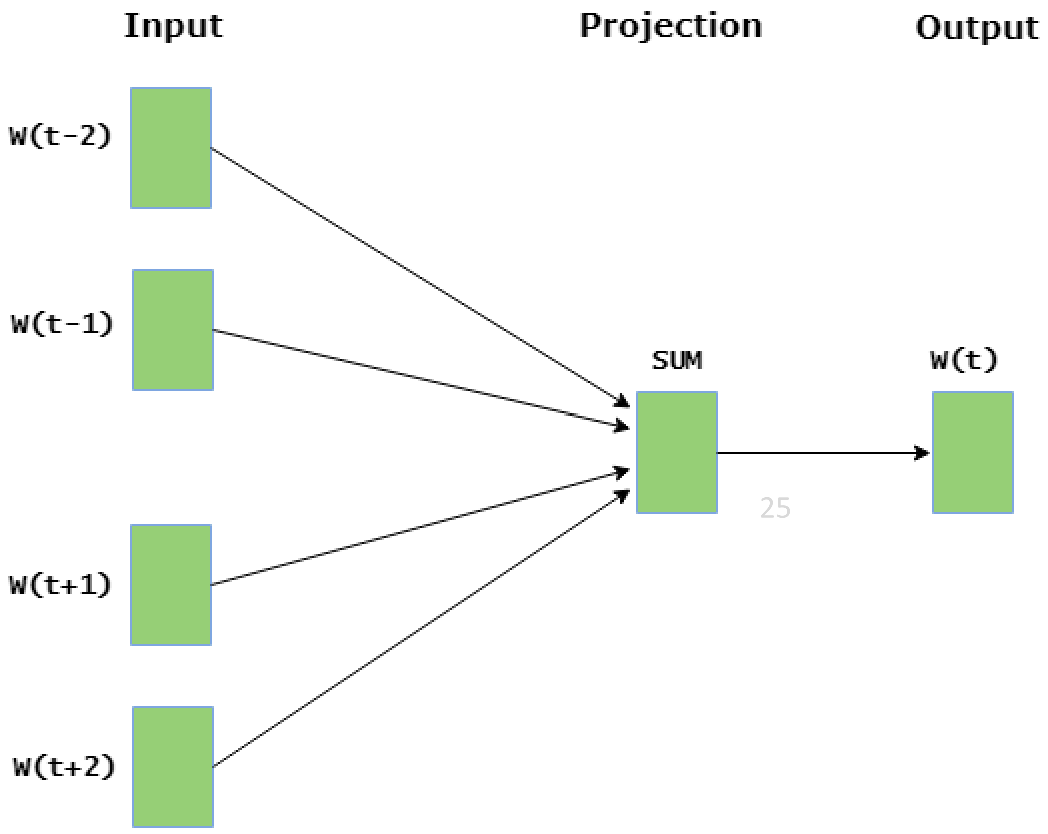
•**Scalability:** Word2Vec can handle big corpora with ease and is scalable. Scalability like this is essential for training on large text datasets.

•**Open Source Implementations:** Word2Vec has open-source versions, including one that is included in the Gensim library. Its widespread adoption and use in both research and industry can be attributed in part to its accessibility.

DL | Word2Vec - Types

1. CBOW (Continuous Bag of Words):

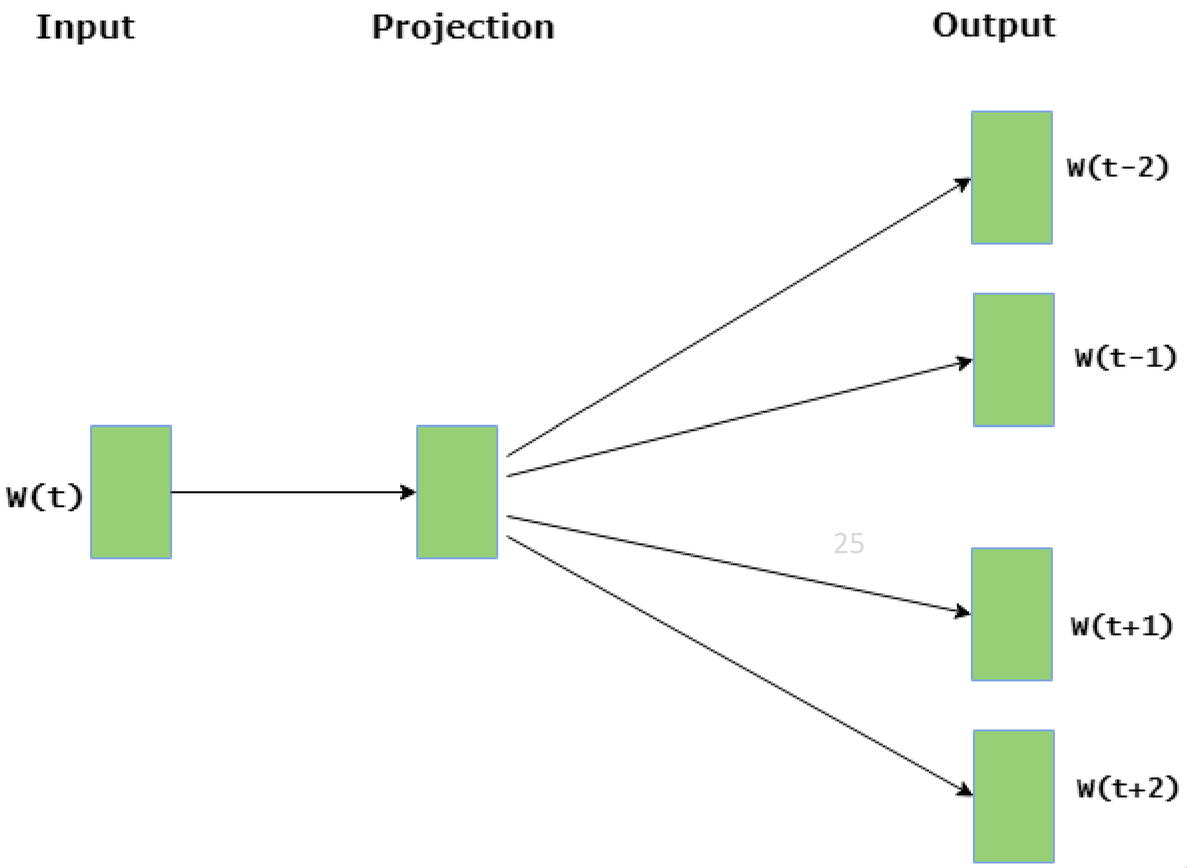
CBOW (Continuous Bag of Words): The CBOW model predicts the current word given context words within a specific window. The input layer contains the context words and the output layer contains the current word. The hidden layer contains the dimensions we want to represent the current word present at the output layer.



DL | Word2Vec - Types

2. Skip Gram :

Skip gram predicts the surrounding context words within specific window given current word. The input layer contains the current word and the output layer contains the context words. The hidden layer contains the number of dimensions in which we want to represent current word present at the input layer.



DL | Applications of Word Embedding

Applications of Word Embedding:

- **Text classification:** Using word embeddings to increase the precision of tasks such as topic categorization and sentiment analysis.
- **Named Entity Recognition (NER):** Using word embeddings semantic context to improve the identification of entities (such as names and locations).
- **Information Retrieval:** To provide more precise search results, embeddings are used to index and retrieve documents based on semantic similarity.
- **Machine Translation:** The process of comprehending and translating the semantic relationships between words in various languages by using word embeddings.
- Question Answering: **Increasing response accuracy and understanding of semantic context in Q&A systems.**

DL| WordEmbedding - Exercise

See code here: https://github.com/ShahzadSarwar10/AI-ML-Explorer/tree/main/Week9

You should be able to analyze – each code statement, you should be able to see trace information – at each step of debugging. “DEBUGGING IS BEST STRATEGY TO LEARN A LANAGUAGE.” So debug code files, line by line, analyze the values of variable – changing at each code statement. BEST STRATEGY TO LEARN DEEP.

Let’s put best efforts.
Thanks.
Shahzad – Your AI – ML Instructor

Thank you - for listening and participating

- Questions / Queries
- Suggestions/Recommendation
- Ideas…..?

Shahzad Sarwar
Cognitive Convergence
https://cognitiveconvergence.com
shahzad@cognitiveconvergence.com
voice: +1 4242530744 (USA) +92-3004762901 (Pak)