iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VI **Month of publication:** June 2026

DOI: https://doi.org/10.22214/ijraset.2026.83505

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

A Review of Word and Sentence Embedding Approaches: From One-Hot Encoding to LLM-based Models

Arwinder Singh, Satpal Singh
[1]*Assistant Professor in Computer Science, University College, Ghanaur, Punjabi University, Patiala*
[2]*Assistant Professor in Computer Science, University College, Chunni Kalan, Punjabi University, Patiala*

Abstract: ***The embedding techniques are the great advancement in Natural Language Processing that converts plain text into numerical representation. This paper presents a review of word as well as sentence embedding techniques. The paper explains traditional embeddings approaches such as one-hot encoding, count-based models, and TF-IDF also current state-of-the-art embedding models such as Word2Vec, GloVe, BERT and transformer based. The paper categorizes word embeddings as frequency & prediction based whereas sentence embeddings are shown in separate section. The paper also presents a comparative analysis of architecture, performance, type and semantic capabilities of the models.***

The embeddings alone are nothing so that semantic similarity measure techniques are applied for detecting similarity between embeddings. This paper also explains various semantic similarity measure techniques such as cosine similarity, Jaccard similarity and distance-based metrics. Intrinsic and extrinsic are two main evaluation methods are available for embeddings and those are also discussed in this paper. The evolution of static, contextual and also large language model-based embeddings is presented in the paper. The paper explains a case study using Punjabi sentences that how embeddings (numeric representation of sentences) capture the semantic similarity between text. This representation is presented using table and images for better understanding. The paper further discusses the recent embedding techniques especially multilingual and cross-lingual models such as mBERT, LaBSE and MiniLM. The type, architecture, key features, strength and limitations of the models are presented in the table. The semantic similarity between sentences of various models is also shown in the paper. The model's name is presented with other important information such as methods/model name, dataset used in the model, evaluation metric, similarity score. One more analysis is presented in the paper about the sentence level embedding approaches available for low resource language especially for Punjabi. The challenges of limited dataset, the effectiveness of pretrained models is also shown in the analysis. Next important thing in this analysis is that it shows the fine-tuning strategies of such models. The paper concludes by presenting the important findings related to word as well sentence embeddings.

Keywords: Word Embeddings, Sentence Embeddings, Semantic Similarity, Deep Learning, Natural Language Processing, Transformer Models, Multilingual Embeddings, Punjabi Language.

I. INTRODUCTION

Artificial Intelligence (AI) is a technology that enables the computer systems to perform the tasks that require human intelligence. In this type of technology, models learn from large data, also consider past experiences to generate outputs. The advancement of AI has enabled it to be used in various fields such as healthcare, industry, transport, finance, agriculture, education. As per the involvement of AI in every field, such models can process text, images and videos. This paper introduces Artificial Intelligence (AI), Machine Learning (ML), Deep Learning (DL) at the beginning of the papers. As shown in the Figure 1, AI is a broad technology whereas Machine Learning is a part of AI and Deep Learning is a sub part of Machine Learning.

AI models have great success in the development of text-based applications which covers paraphrase detection & generation, question-answering, sentiment analysis, summarization. This paper focuses on how text data is processed using Deep Learning because the deep learning algorithms can only understand numbers. So, we need to process text data to convert from text to numbers called vectors or embeddings. There are traditional as well as current approaches to convert text into embeddings. One hot encoding, frequency-based approaches come under traditional approaches whereas prediction-based approaches such Word2Vec and GloVe are also explored in this paper. The text can be converted into word, sentence or even document embeddings. This paper aims to introduce the various traditional as well as current approaches to convert words and sentences into embeddings.

The conversion of text into embeddings is so important because embeddings are used to detect semantic similarity between text. The analysis of word as well as sentence embeddings is also presented in the paper.

The paper explains a case study using Punjabi sentences as examples that how embeddings (numeric representation of sentences) capture the semantic similarity between text. This representation is presented using table and images in this paper for better understanding. The paper further discusses the recent embedding techniques especially multilingual and cross-lingual models such as mBERT, LaBSE and MiniLM. The different type of information such as type, architecture, key features, strength and limitations of multilingual and cross-lingual models is presented in the table.

The performance of various embedding models is also shown in the paper. The model's name is presented with other important information such as method/model name, dataset used in the model, evaluation metric, similarity score. One more important information is provided in the paper that is about the sentence level embedding approaches available for low resource language especially for Punjabi. Although, there are multilingual and cross-lingual approaches are available but few challenges are still in from of us. These are limited dataset, the effectiveness of pretrained models is also shown in the analysis. The paper concludes by presenting the important findings related to word as well sentence embeddings.

A. Artificial Intelligence

Can machines think? This question was asked by Alan Turing (1950) and the statement became the backbone for AI. His paper "Computing Machinery and Intelligence" forced the researchers to think about AI. Later on, AI applied in various real-life applications. A traditional chess program used hand crafted rules of AI but machine learning was absent in such programs.

The fundamental aim of AI is to develop expert systems by employing human intelligence in machines. So, when we see technological applications, AI is applied in every area of our daily life such as gaming, expert systems, robots, driverless cars and even aircrafts. Text processing applications come under Natural Language Processing (NLP). So, machine translation, paraphrase detection & generation, spell checker, summarization and speech recognition are the main applications of NLP.

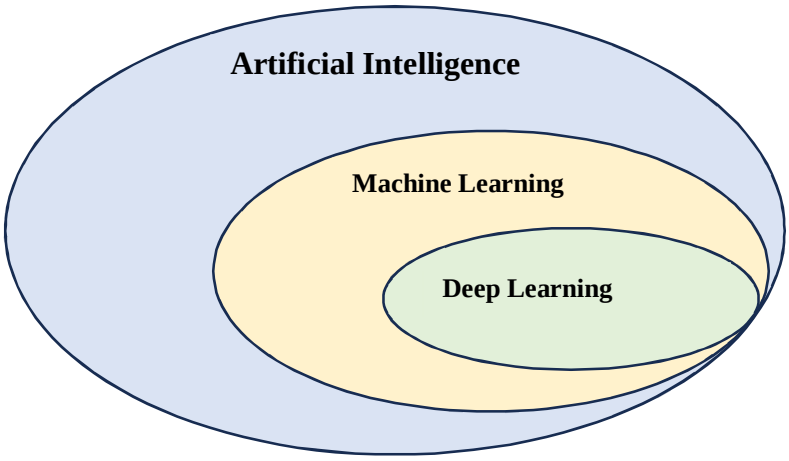


Figure 1: The relationship between Artificial Intelligence, Machine Learning and Deep Learning

B. Machine Learning

Machine learning algorithms performed well for developing AI based applications (Mitchel 1997). The researchers thought at that time the large data sets will be backbone for these models. Here large dataset means, millions of sentences in text data, millions of images etc. Another similarity of AI based models with human mind is that such models learn from past experiences. The models learn from input data as well as from patterns in data. Other than this, AI models learn from large data for better results. More data improves the results of machine learning algorithms.

C. Deep Learning

Deep Learning is a part of Machine Learning that works to understand complex datasets. Deep Learning based algorithms are so intelligent which work as per the human brain work. It also makes such type of models which can learn from patterns with in the unstructured data. Deep Learning follows multi layered neural network architecture where data processes through these layers. The multiple layers mean data processed iteratively for adjusting weights and biases to improve the results.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

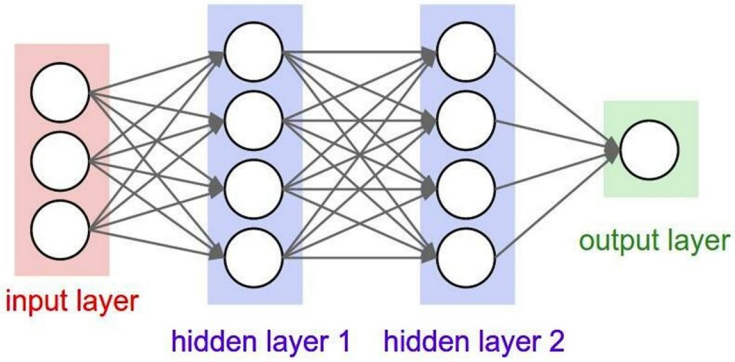


Figure 2: Multi layered architecture of Deep Learning models consisting of input, hidden and output layers (https://medium.datadriveninvestor.com/a-primer-into-neural-networks-a0bc02d8513b)

The basic architecture of multi layers is shown in the below image where Deep means to apply multiple layers of interconnected layers. Here, first is input layer, last is output layer and there may be multiple hidden layers between input and output layer (LeCun et al., 2015). In every model, there will be input and output layer but hidden layers may vary as per the requirement of the problem.
Due to the accuracy of the deep learning models for processing image, video, audio and text data, it is applied in every area of today's life. Out of those different areas, deep learning models are famous to develop natural language processing applications. This paper presents a brief introduction how text data is being processed by the deep learning models (Goldberg, 2017).

II. PROCESS TEXT DATA USING DEEP LEARNING

A. Introduction

The text data can be represented as sequence of characters, words, sentences, paragraphs and then documents. The characters are the smallest unit of text but the words are more common to use. So, we can feed characters, words or sentences to the deep learning models for further processing. Now the question is that deep learning models can read this type of data? The answer is "no". So, such models read numeric representation of the text. Such numeric representation of text data is called vectors or embeddings. So that there may be other preprocessing steps for cleaning text data etc. But this paper is presenting various methods to generate embeddings of the text data. The embeddings is just a numeric representation of the text and this paper is going to talk about the methods to convert words and sentences into embeddings.

B. One Hot Encoding

One hot encoding is a traditional approach to convert sequence of text into vectors. So, consider the following example to understand this. Here a sentence is written in English language. Here the sentence is converted into dictionary where unique words must be there present in the dictionary (Goldberg, 2017).
Input Sentence: Patiala is a beautiful royal city.
Dictionary: 'Patiala', 'is', 'a', 'beautiful', 'royal', 'city'
One hot vector of beautiful = [0, 0, 0, 1, 0, 0]

In the above example the vector is represented in 0 and 1. Now, 0 means word is not present at that location where as 1 means the word is present in the list.

C. Limitations of One Hot Encoding

1) This is not a better technique to detect similarity between words based upon their features. For example, apple and banana can be similar as per some features but in one hot encoding, apple can be similar to others also.
2) Dictionary means the total number of unique words in the input as shown in the previous example, so it will increase the dimension of the vector. The large size vector is difficult to compute.
3) In the previous example, there are maximum 0's in the feature vector i.e. is also known as sparse data which becomes difficult for machine to compute.
4) These problems solved by word embeddings i.e. discussed in the next section.

III. WORD EMBEDDINGS

Word Embedding is a way in Natural Language Processing to represent text as numerical vectors in a semantic space. It is also known as computational model of meanings which is further used to find the semantic similarity between text or words. The name of the model is also telling that this is for words and embeddings of the words help to detect meanings of the words within in a context.

The words are represented in a n-dimensional vectors and then the similarity is calculated as angles between vectors. In section 8, various techniques are discussed to detect similarity between embeddings. Here n can be an integer that can vary from 1 to a large number. To represent words in a large space means that if the words are close in space, they will have the same meanings,

IV. CONVERT TEXT DATA INTO WORD EMBEDDINGS

In this section, frequency and prediction-based embeddings are discussed.

A. Frequency based

1) Count Vectors

A count vector word embedding represents each document using the frequency of every word it contains (Jurafsky & Martin, 2026). For example, suppose we have two documents (D1 and D2) and a vocabulary of 10 distinct words (N).

In this case, we can construct a count vector matrix of size D × N, where each cell indicates how many times a particular word appears in a specific document.

D1: Patiala is a royal city. It is located in Patiala.

D2: Patiala is famous for its culture.

Dictionary = [‘Patiala’, ‘is’, ‘a’, ‘royal’, ‘city’, ‘it’, ‘located’, ‘in’]

Documents	Words											
	Patiala	is	a	royal	city	It	located	in	famous	for	its	culture
D1	2	2	1	1	1	1	1	1	0	0	0	0
D2	1	1	0	0	0	0	0	0	1	1	1	1

← document vector

↑ Word vector

Figure 3: An example of Count Vectors Approach

In the above Figure 3, there are columns are considered as word vectors and rows are considered as document vectors.

2) Term Frequency-Inverse Document Frequency (TFIDF)

TFIDF is another approach to represent text as vectors. This approach differs from the count vector’s technique in that it focuses on word's occurrence in the whole text or document (Salton et. 1988). There can be very common words in the given text or document. The English phrase may include, "the," "is," "are," and "a" as common words whereas the Punjabi has "ਹੈ" (hai) (is), "ਹਨ" (han) (are), "ਉਹ" (uh) (they), "ਸੀ" (si) (was), and "ਸਨ" (n) (were) are commonly used words. These words are used frequently within the documents but provide less important information. Therefore, the main advantage of TFIDF is that it gives high weight to the important terms and less weight to the common words in the document. Then the following formula is used to calculate the similarity score.

TF = (Number of times term t appears in a document) / (Number of terms in the document)

3) Co-Occurrence Matrix Word by Word

Sometimes, the words are used in the same context then they will mean the same thing. So, consider the following example to grasp the concept:

Swift is a car. Nexon is a car.

In the above example, ‘Swift’ and ‘Nexon’ come under the same context i.e. ‘car’. So, these words have the same semantic meaning. The first step is to construct cooccurrence matrix for the given text using window size. As an example, is provided as below:

Patiala is a royal city

A list for each word can be defined row-wise or column-wise from cooccurrence count matrix. For example, the list for Patiala is (0, 1, 0, 0, 0) this list is also considered as context vector on the basis vocabulary and window size. Thus, this process applies continuously for getting the vectors for each word. Then the similarity between words is calculated by determining the angles between vectors. There are several approaches to determine similarities between words (section 8) using word vectors but cosine similarity is a prominent approach to do this.

Other methods that are highly helpful for forming text as vectors are Random Indexing (RI), Latent Semantic Analysis (LSA), and Latent Dirichlet Allocation (LDA).

B. Prediction Based

Due to the issues in frequency-based approaches, a Google team created the first prediction-based method "Word2Vec" for creating word embeddings in 2013. Now days, a number of additional methods available for creating word embeddings which are discussed in this section.

1) Word2Vec

Word2Vec (Mikolov, Chen, Corrado, and Dean 2013) is a way to represent words as vectors. This is a famous word embedding model and is also known as a beginner for deep learning. It is a predictive model to learn the word embedding from a large unlabelled data. The fundamental objective of the model is to group comparable items and also to preserve the semantic links between words as illustrated in Figure 4. This methodology allows us to locate comparable words, describe the relationships between words, and cluster the words. The model may be trained using two approaches i.e. Continuous Bag of Words Model (CBoW) and Skip Gram (SG). As seen in Figure 4, this model depicts the connections between various terms.

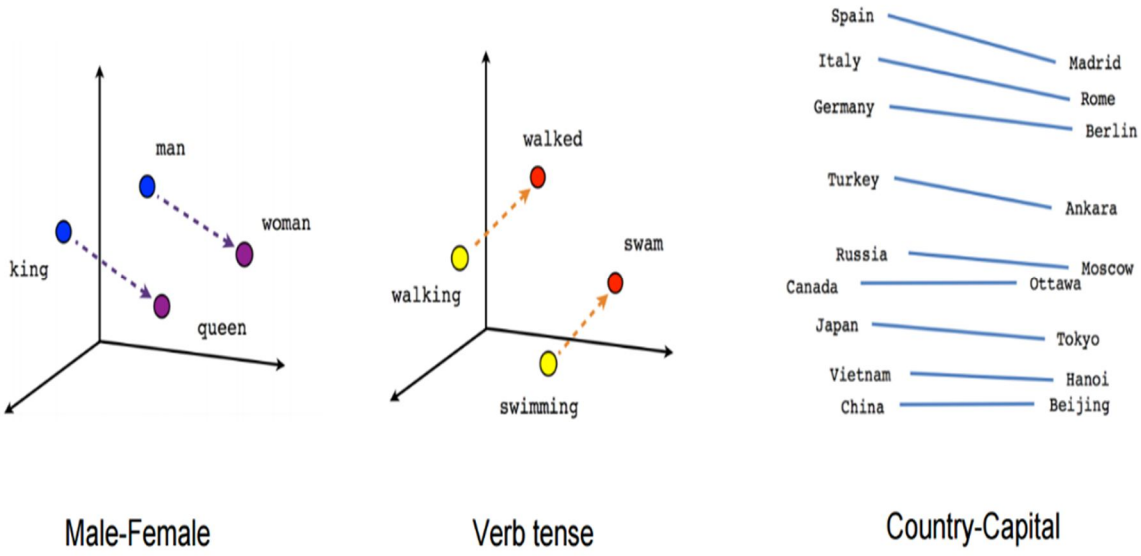


Figure 4: Word2Vec: An Illustration (https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-coUnt-word2veec/)

- Continuous Bag of Words Model (CBoW): This model is to predict the term from its context (Mikolov, Chen, Corrado, and Dean 2013). The context may be a single word or a combination of words. The input for the model is w(t2), w(t1), w(t+1), Diagram 5 explains the expected words w(t+2) and w(t). The prediction of the target term depends upon its context. This model generates results with high precision, but a lot of data is needed. For frequently occurring terms, CBOW performs better. The model's primary benefit is its reduced memory usage.
- Skip-gram: The skip-gram model works as opposite to CBOW. This model uses the given words to consider the context (Mikolov, Chen, Corrado, and Dean 2013). For instance, the provided word is w(t), while the output is w(t2), w(t1), w(t+1), and w(t+2).

For uncommon terms, this method performs better. The context can be one word or multiple words.

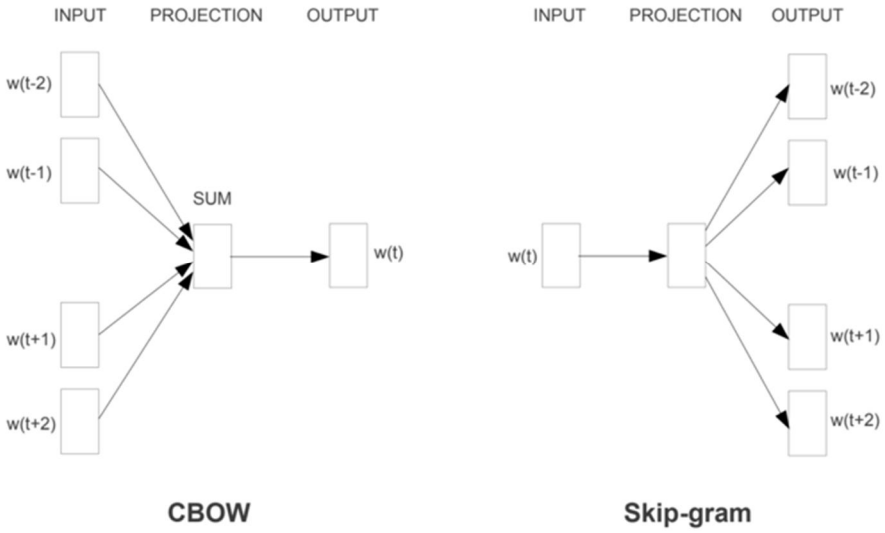


Figure 5: Architecture of CBOW and Skip-gram (Mikolov, Chen, Corrado, and Dean 2013)

2) Glove

Pennington, Socher, and Manning (2014) presented the global vector technique of word representations, which is identical to Word2vec. While GLOVE employs cooccurrence matrix and word2vec is a prediction-based model. As explained above, the vector shows the associations between words by displaying how they appear together. It is calculated by counting the number of words in a corpus that appear together.

3) BERT

Word2Vec uses a fixed representation for every word, instead of knowing the situation of the context in which it is used. The BERT was first presented by (Devlin, Chang, Lee, and Toutanova 2019) which has been considered as a revolutionary shift in a number of NLP tasks. It creates word embeddings based on the context of the words. The following example can explain the difference:

- The man was going to the bank.
- The man went fishing by the bank of the river.

The word ’bank’ in both statements has a different meaning. While the BERT would provide distinct embeddings for each phrase, the Word2Vec would produce the same embedding for the word "bank" in both.

The BERT is an extension to the transformer model (Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, and Polosukhin 2017) as it stacks the multiple encoders. The authors proposed two BERT models i.e. base and big. Twelve encoder layers, twelve attention heads, and seven hundred and sixty-eight hidden units make up the base BERT. However, huge BERT makes use of 1024 hidden units, 16 attention heads, and 23 encoder layers.

4) Analysis of Word Embedding Approaches

This section is providing an analysis about the word embedding approaches those discussed in the previous section. As we discussed in the Table 1, there are number of approaches where we can evaluate our word embeddings models.

Method	Type	Context Awareness	Dimensionality	Sparsity	Semantic Capture	Scalability	Typical Use
One-Hot Encoding	Traditional	None	Very High	Sparse	None	Poor	Baseline
Count Vector (BoW)	Frequency-based	Limited	High	Sparse	Weak	Moderate	Text classification
TF-IDF	Frequency-based	Limited	High	Sparse	Moderate	Good	IR systems
Co-occurrence Matrix	Frequency-based	Partial	Very High	Sparse	Moderate	Poor	Early embeddings
Word2Vec	Prediction-based	Local context	Low	Dense	Good	Excellent	NLP tasks
GloVe	Prediction-based	Global context	Low	Dense	Very Good	Good	Semantic tasks

Table 1: Analysis of Word Embedding Approaches

Some of the analysing points as discussed in the table

- Traditional approaches suffer from the dimensionality problems
- Frequency based approaches are able to find the statistical importance instead of meanings
- Prediction-based methods are able to find the semantic meanings as well as relationship
- One important analysis shown in the table is the flow of processes from sparse to dense representation.

V. CONVERT TEXT DATA INTO SENTENCE EMBEDDINGS

Sentence embeddings are helpful to capture the semantic similarity between the sentences. Sentence embeddings is a highly useful and recommended area in NLP which has number of applications. The sentences from input text can be defined as sentence embeddings using both conventional and neural network-based methods. Now days, the transformer models are being used for converting sentences as embeddings. This section will provide a brief introduction about the various sentence embeddings approaches.

A. Average Word Embeddings

Average word embedding approach follows two steps to generate sentence embeddings. First step is to create word embeddings and then the second step is to create sentence embeddings. Actually, here sentence embedding is created by averaging the word embeddings. One of the architectures of sentence embeddings generation with Average Word Embeddings is presented in paper (Singh A, Josan GS, 2021). After reading the input text as a sentence, SentVecAve model divides it into words (w1, w2,..., wn). Next step, each word's embedding from the trained word2Vec model is read by the model. At the last step, all the word vectors are then averaged to generate a single vector which is considered as sentence vector.

B. RNN Based Sequence-to-Sequence

The advancements in Neural network helped to make the generative models (Bowman, Vilnis, Vinyals, Dai, Jozefowicz, and Bengio 2016; Chung, Kastner, Dinh, Goel, Courville, and Bengio 2015), which further provided an opportunity for solving complicated sequence-related problems. Recurrent Neural Network models were very helpful for handling common NLP problems such as machine translation, question answering, and text summarization also known as Seq2Seq (Sutskever, Vinyals, and Le 2014). Seq2Seq models are frequently implemented using the encoder-decoder architecture. Since LSTMs (Hochreiter and Schmidhuber, 1997) maintain vector representations as internal states, where both the encoder and decoder are LSTM models. The input sequences read by the encoder are then used as internal state vectors. The decoder generates output sequences after receiving the internal state vector as an input encoded by the encoder.

C. Skip-Thought

Skip-gram approach works for words whereas Skip-Thought (Kiros, Zhu, Salakhutdinov, Zemel, Urtasun, Torralba, and Fidler 2015) is used to generate sentence vectors and is also employed to forecast the following sentence. The authors of this approach followed an encoder-decoder methodology to construct sentence vectors.

D. FastSent

This approach is similar to Skip-Thought technique but it followed the BOW algorithm to detect sentences that are very next to the input sentence. This technique is also very effective for creating sentence vectors (Hill et. 2016).

E. Universal Sentence Encoder

A next method presented by Google researchers (Cer, Yang, Kong, Hua, Limtiaco, John, Constant, Guajardo Cespedes, Yuan, Tar, Sung, Strope, and Kurzweil 2018) for creating sentence embeddings. In this technique the encoder is responsible to take the tokenized sentence as input and transforms it into a 512-dimensional vector. The Universal Sentence Encoder uses two different kinds of encoders. First is transformer encoder and the second is DAN. After collecting unigram and bigram embeddings, the DAN averages them to create a sentence vector, which is subsequently transmitted to the neural network for sentence embedding.

F. SentenceBERT

Another Transformer-based sentence embeddings model presented by (Devlin, Chang, Lee, and Toutanova 2018; Reimers et. 2019). The sole encoder of the BERT model, a transformer model with RNN, was employed in the suggested SentenceBERT method. The

BERT encoder is used to encode the input text into a large-dimensional vector. The encoder output is then passed through the pooling layer to get fixed vectors (u, v). The similarity is then calculated by applying cosine similarity on vectors u and v.

G. SimCSE

SimCSE (Simple Contrastive Learning of Sentence Embeddings) (Gao et. 2021) is a technique used in natural language processing to generate high-quality sentence embeddings. It is also known as numerical vector representations of sentences that capture their semantic meaning. The SimCSE is built on top of transformer models like BERT. SimCSE applies contrastive learning which means the same sentence is passed through the model twice may be with dropout to create two similar embeddings. Here the similar embeddings are put closer and embeddings of different sentences are farther. This approach of sentence embedding models helps to detect the meanings of sentences effectively. SimCSE can be trained in both unsupervised and supervised ways. Here, supervised means using the labelled sentence pairs for better performance. Due to the effectiveness of SimCSE, it is widely used in various NLP tasks such as semantic similarity, search systems, clustering, and recommendation system. The SimCSE is trained on English dataset so this is not powerful for Punjabi text.

H. Multilingual MiniLM

Multilingual MiniLM is another efficient sentence embedding model (Reimers et 2020; Wang et. 2020) and the fundamental advantage of this model is that it designed to support multiple languages, including Punjabi. This approach is also part of the Sentence Transformers family and based on transformer architectures like XLM-RoBERTa. The model is trained on large parallel and multilingual datasets to generate semantically meaningful embeddings for sentences of different languages. The main advantage of this model is the balance between speed and performance which making it suitable for real-time NLP applications such as semantic search, clustering, and recommendation systems. Multilingual MiniLM can understand and compare sentence embeddings even if they are written in different languages, which makes it especially useful in multilingual environments like Indian education systems.

I. LaBSE (Language-agnostic BERT Sentence Embedding)

LaBSE is developed by Google which is a powerful multilingual sentence embedding model (Feng et. 2022). This model supports more than 100 languages including Punjabi. It is also built on a BERT-based architecture. The model trained on large parallel dataset to align sentence meanings across languages into a shared embedding space. This allows LaBSE to generate accurate semantic representations which further useful for various tasks such as translation, multilingual search, and duplicate detection. As compare to other models, LaBSE offers very accurate but demands more computational resources. The main advantage of this model is to map semantically similar sentences on large semantic space from different languages. This technique is more interesting as similar sentences of different languages are closer than the dissimilar sentences.

J. Analysis of sentence Embeddings Approaches

Model	Architecture	Language Support	Speed	Accuracy	Training Type	Key Feature	Limitation
RNN Seq2Seq	RNN / LSTM Encoder-Decoder	Limited (data-dependent)	Slow	Moderate	Supervised	Early sequence modelling approach	Poor long-range context handling
Skip-Thought	RNN Encoder-Decoder	Mostly English	Slow	Moderate	Unsupervised	Learns from surrounding sentences	Computationally heavy, outdated
FastSent	Bag-of-Words	English	Very Fast	Low	Unsupervised	Simple and efficient	Weak semantic understanding
Universal Sentence Encoder (USE)	Transformer / DAN	Multilingual	Fast	Good	Supervised + Unsupervised	Easy to use, general-purpose embeddings	Less accurate than newer models
Sentence-BERT (SBERT)	Siamese BERT	Mostly English	Medium	High	Supervised	Optimized for semantic similarity	Limited multilingual support
SimCSE	BERT + Contrastive Learning	Mostly English	Medium	Very High	Self/Supervised	Simple and powerful embeddings	Weak for non-English languages

Model	Architecture	Language Support	Speed	Accuracy	Training Type	Key Feature	Limitation
MiniLM (Multilingual)	Transformer (XLM-R based)	50+ languages	Very Fast	Good	Distillation + Fine-tuning	Lightweight and multilingual	Slightly less accurate than LaBSE
LaBSE	BERT-based	100+ languages	Slow	Very High	Supervised (parallel data)	Strong cross-lingual embeddings	Large and resource-intensive

Table 2: Analysis of Sentence Embedding Approaches

VI. AN EXAMPLE TO UNDERSTAND THE DIFFERENCE OF WORD AND SENTENCE EMBEDDING APPROACHES

To understand the difference between word and sentence embedding approaches, consider a following Punjabi sentence:
ਬਿੱਲੀ ਚਟਾਈ ਉੱਤੇ ਬੈਠੀ ਹੈ" (The cat is sitting on the mat)

Now, look at different word embedding approaches, how they create relationship among different words.

1) One-hot encoding: this method represents each word (ਬਿੱਲੀ (cat) & ਕੁੱਤਾ (dog)) independently without preserving any semantic relationship within words.
2) Frequency based approaches: This technique also does not capture the semantic relationship as it just gives weights based on their occurrence in the text.
3) Word2Vec: This approach creates vectors for words according to the context in which they occur. In the above example, "ਬਿੱਲੀ" (cat) and "ਕੁੱਤਾ" (dog) can be mapped to nearby vectors due to the context and so that words are semantically similar.
4) Bert: As explained in the example of section 5, Bert is more reliable and accurate to understand the actual meaning of the word.
5) Sentence Embeddings: The various approaches of sentence embeddings discussed in the previous section are useful for generating sentence embeddings for the given sentences. For example, we have the following sentences:

Example 1:

a) ਬਿੱਲੀ ਚਟਾਈ ਉੱਤੇ ਬੈਠੀ ਹੈ (The cat is sitting on the mat)
b) ਇੱਕ ਬਿੱਲੀ ਫਰਸ਼ ਤੇ ਬੈਠੀ ਹੈ (A cat is sitting on the floor)

Example 2:

a) ਮੌਸਮ ਬਾਰਿਸ਼ ਵਾਲਾ ਹੈ (The weather is rainy)
b) ਅੱਜ ਮੀਂਹ ਪੈ ਰਿਹਾ ਹੈ (It is raining today)

Sentence embedding techniques are able to generate dense embeddings for complete sentences. These dense embeddings are then used for NLP tasks. The semantically similar sentences will get nearby embeddings and will also be mapped nearby on semantic space. As shown in the above example 1, sentence a and b will be mapped nearby but sentences in example 2 will be mapped far from each other. Also see Figure 8 for more details where similar and dissimilar sentences are shown in semantic space. The current sentence embedding approaches can also detect the similarity even though words are not similar in the sentences.

VII. ESTIMATING SEMANTIC SIMILARITY

The previous section explored various word as well as sentence embeddings techniques. The embeddings further used to detect the semantic similarity of words and sentences. There are various techniques for identifying semantic similarity between embeddings (Manning et al., 2008; Jurafsky et al. 2023) as explained in this section.

A. Cosine Similarity

Cosine similarity is a famous approach to determine the semantic similarity between embeddings. It is the cosine of the angle between two vectors of n dimension. It is just a dot product between two vectors split by the product of the lengths of the two vectors. The mathematical expression of cosine similarity is as follows:

$$sim = \frac{(a \cdot b)}{(||a|||b||)}$$

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

B. Jaccard Similarity

Jaccard similarity is used to find the lexical similarity between two strings. This can be determined as the ratio of the intersection to the union of the elements in two strings as stated in the following equation. The issue with this method is that it cannot detect semantic similarity.

$$Jaccard(S^1, S^2) = \frac{(S^1 \cap S^2)}{(S^1 \cup S^2)}$$

C. Euclidean Distance

This method is used to find the shortest path between two places. So, this method is also helpful for determining the following commonalities between two embeddings:

$$dist_E(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)}$$

D. Manhattan Distance

Another technique for calculating the separation between two vectors is the Manhattan Distance. This is actually the total of vertical and horizontal distance between two places.

$$d(M, P) = |M_x - P_x| + |M_Y - P_Y|$$

E. Minkowski Distance

The generalized version of Manhattan and Euclidean distances is called the Minkowski Distance. It can be expressed mathematically as the following equation:

$$dist_M(\vec{x}, \vec{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^N \right)^{\frac{1}{N}}$$

F. Combination of Cosine and Jaccard Similarity

The authors in Singh, A. and Josan, G., S. (2020) used the combination of Cosine and Jaccard similarity for finding similar pairs from Punjabi newspapers. They said that this approach is more reliable to find semantically as well as lexically similar pairs.

VIII. EVALUATION MATRIX

This is an important phase for every model especially in machine translation, paraphrase detection & generation and sentiment analysis. This paper is talking about word and sentence embeddings so it becomes more important to measure the models because there may be different words in sentences as shown in the previous section. Intrinsic and extrinsic are two main evaluation methods are available for embeddings. Intrinsic evaluation finds the embedding's quality without considering the appropriate application. It relies on word similarity and further cosine similarity applies on vectors.

The second approach is extrinsic that understands embeddings according to the real-world applications such as sentiment analysis, machine translation, question-answering and text classification. Accuracy, F1-score and precision-recall are commonly used performance metrics. Important, the embeddings models are considered good if their intrinsic and extrinsic evaluation is fine (Wang et al., 2019; Jurafsky et al., 2023).

IX. EXAMPLES OF PUNJABI TEXT ON EMBEDDING SPACE

This section is explaining how we put word and sentence embeddings on semantic space. Two sentences are shown in Table 3 and then as in Figure 6 & 7. Figure 6 is explaining that words or sentences are given to the system as input text. Then the preprocessing is applied on that text which includes normalization, splitting is done. The next step is to create a tokenizer of given text. The next important part is to encode the words to id's. The last step is to create word or sentence embeddings.

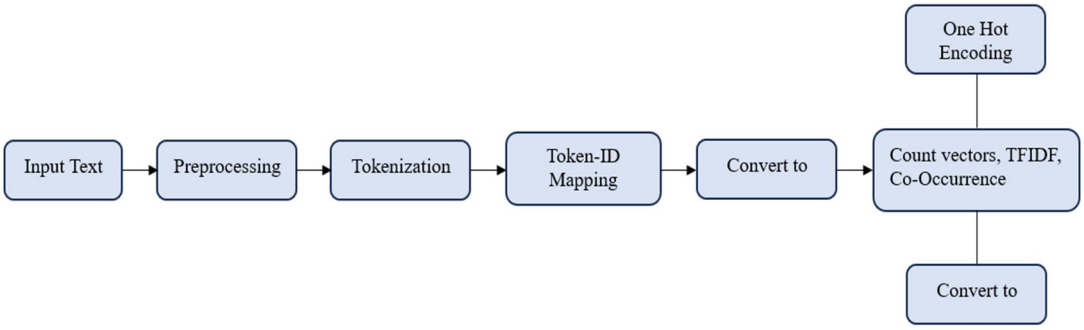


Figure 6: Process of converting plain text into embeddings

	Process	Example 1	Example 2	Description
Raw Text	Input sentence	ਬਿੱਲੀ ਚਟਾਈ ਉੱਤੇ ਬੈਠੀ ਹੈ	ਰਾਸ਼ਟਰੀ ਪ੍ਰਦੂਸ਼ਣ ਰਹਿਤ ਦਿਵਸ ਮਨਾਇਆ	Original Punjabi text
Preprocessing	Cleaning & normalization	ਬਿੱਲੀ ਚਟਾਈ ਉੱਤੇ ਬੈਠੀ ਹੈ	ਰਾਸ਼ਟਰੀ ਪ੍ਰਦੂਸ਼ਣ ਰਹਿਤ ਦਿਵਸ ਮਨਾਇਆ	Standardize text, remove noise
Tokenization	Split into words	[ਬਿੱਲੀ, ਚਟਾਈ, ਉੱਤੇ, ਬੈਠੀ, ਹੈ]	[ਰਾਸ਼ਟਰੀ, ਪ੍ਰਦੂਸ਼ਣ, ਰਹਿਤ, ਦਿਵਸ, ਮਨਾਇਆ]	Break into tokens
Encoding	Word → index mapping	[12, 45, 78, 33, 9]	[101, 56, 77, 88, 23]	Convert words to numeric form
Embedding	Dense vectors	ਬਿੱਲੀ → [0.21, -0.34, …]	ਰਾਸ਼ਟਰੀ → [0.11, 0.52, …]	Semantic vector representation
Sentence Embedding	Whole sentence vector	[0.45, -0.12, …]	[0.62, 0.08, …]	Context-aware representation
Application	NLP tasks	Similarity / Chatbot	Classification / Topic detection	Real-world usage

Table 3: Steps of converting plain text into embeddings

The process is explained using examples as below.

Example Sentences

1. *"ਬਿੱਲੀ ਚਟਾਈ ਉੱਤੇ ਬੈਠੀ ਹੈ"* (The cat is sitting on the mat)
2. *"ਰਾਸ਼ਟਰੀ ਪ੍ਰਦੂਸ਼ਣ ਰਹਿਤ ਦਿਵਸ ਮਨਾਇਆ"* (National Pollution Control Day celebrated)

The Table 3 explains the step-by-step process that how Punjabi sentences are converted into embeddings such as numeric representation. Pre-processing is applied first to normalize the input text. Then tokenizer is created of sentences. These tokens are then converted into numerical indices using appropriate encoding technique. At the end, each word is converted into dense vector in word embeddings. On the other hand, sentence embedding techniques generate a single vector for whole sentence. This process is shown in Table 3 and then also in Figure 7. Here important thing is that word as well as sentence vectors preserve semantic meanings so that similar words and sentences will be close to each other as shown in Figure 8.

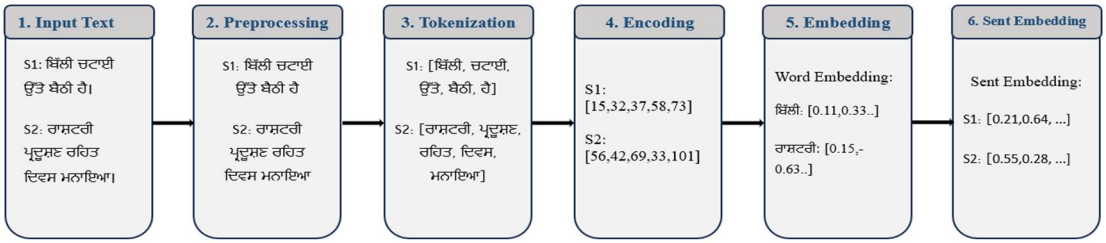


Figure 7: Process to convert Punjabi sentences into Embeddings

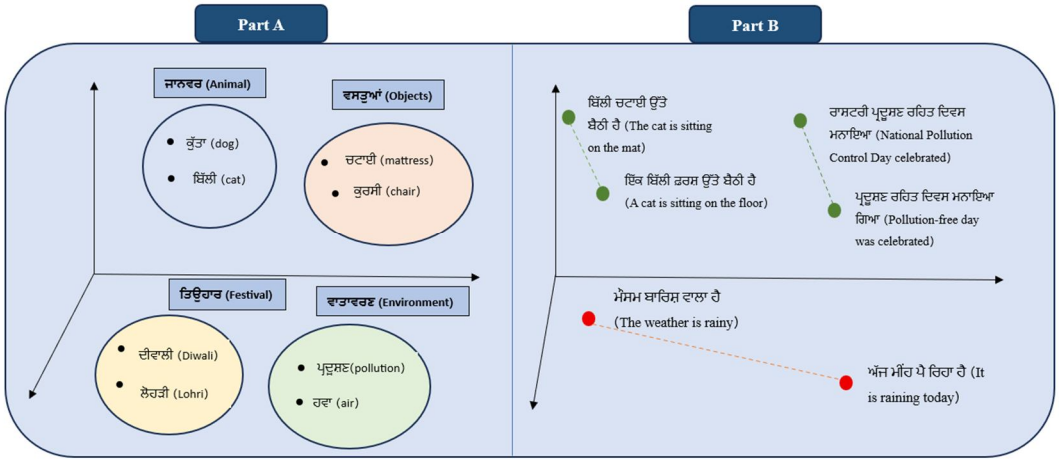


Figure 8: Representation of Punjabi sentences in semantic space

Figure 8 shows visualization of words (part A) as well as sentence embedding (part B) in semantic space. Here we can clearly see that how semantically similar Punjabi words and sentences are placed closer to each other. Figure 8, Part A is showing 'ਬਿੱਲੀ' (cat) and 'ਕੁੱਤਾ' (dog) are close as they semantically similar, here semantically similar means these words appears in same context. In the Figure 8, Part A different four groups/clusters are shown such as ਜਾਨਵਰ (animals), ਵਸਤੂਆਂ (objects), ਤਿਉਹਾਰ (festivals) and ਵਾਤਾਵਰਣ (environments).

To understand Part B of Figure 8, consider the following examples first:

Example 1:

a) ਬਿੱਲੀ ਚਟਾਈ ਉੱਤੇ ਬੈਠੀ ਹੈ (The cat is sitting on the mat)
b) ਇੱਕ ਬਿੱਲੀ ਫਰਸ਼ ਉੱਤੇ ਬੈਠੀ ਹੈ (A cat is sitting on the floor)

Example 2:

a) ਮੌਸਮ ਬਾਰਿਸ਼ ਵਾਲਾ ਹੈ (The weather is rainy)
b) ਅੱਜ ਮੀਂਹ ਪੈ ਰਿਹਾ ਹੈ (It is raining today)

Example 3:

a) ਰਾਸ਼ਟਰੀ ਪ੍ਰਦੂਸ਼ਣ ਰਹਿਤ ਦਿਵਸ ਮਨਾਇਆ (National Pollution Control Day celebrated)
b) ਪ੍ਰਦੂਸ਼ਣ ਰਹਿਤ ਦਿਵਸ ਮਨਾਇਆ ਗਿਆ (Pollution-free day was celebrated)

The B part of the Figure 8 is clearer where three different sentence pairs are shown. The semantically similar sentences are shown closer to each other or we can say in same cluster even though words are different. Both the sentences in example 1 and 3 are presenting similar meanings, so they are shown with green link between them. The sentences in example 2 are not saying the same meaning so that the pair is linked with red line.

This section provides a processing as well as visual information of converting text into embeddings through Table 3 and Figure 7 & 8. The processing pipeline explains technical steps to convert plain Punjabi text into embeddings whereas the relationship between embeddings is shown in semantic space.

X. RECENT TRENDS IN TEXT EMBEDDINGS

The advancements in NLP changed the text embedding techniques from static models to LLM-based approaches. The earlier approaches such as Word2Vec and Glove assign a single vector to the words without understanding the context in deep. The recent LLM-based models are able to generate dynamic embeddings for capturing the contextual meanings. The big achievement is happened with the use of transformer-based models such as BERT and its variants.

Table 4 is showing the recent models of text embeddings. Here in the table, type of the model, their architecture & features are provided. The strength and limitations of the models are also presented in the table.

Model	Type	Architecture	Key Feature	Strength	Limitation	Reference
BERT	Contextual	Transformer Encoder	Bidirectional context	Strong semantic understanding	Computationally heavy	Devlin et al. (2019)
SBERT	Sentence-level	Siamese BERT	Optimized for similarity	Fast similarity computation	Limited multilingual (base version)	Reimers & Gurevych (2019)
MiniLM	Lightweight	Transformer (distilled)	Efficient and fast	Good performance with low cost	Slightly lower accuracy	Wang et al. (2020)
LaBSE	Multilingual	BERT-based	Cross-lingual embeddings	Supports 100+ languages	Large model size	Feng et al. (2022)
mBERT	Multilingual	Transformer	Multilingual representation	Wide language coverage	Uneven performance	Devlin et al. (2019)
GPT-based Embeddings	LLM-based	Transformer Decoder	Deep contextual understanding	Very high semantic quality	Expensive, API-dependent	Brown et al. (2020)

Table 4: Recent Trends in Text Embeddings

XI. PERFORMANCE OF SENTENCE EMBEDDINGS MODELS ON SEMANTIC SIMILARITY

This section explores the performance of various sentence embeddings model on sentence similarity tasks. Word2Vec and Glove are word embedding models but they were used as sentence embeddings by averaging the word vectors and these are also considered as baseline models for comparison. The authors in paper Singh, A. and Josan, G., S. (2021) compared Word2Vec and Seq2Seq on paraphrase detection for Punjabi which proved Seq2Seq was better than average models. The Seq2Seq model was able to capture sequential dependencies which was better than the static embeddings. The following Table is also showing the recent sentence embeddings models and there is huge gap between the score of traditional and latest models. This gap is due to the use of transformer-based models which are able to understand the contextual representations.

S No	Model	Type	Method	Dataset	Metric	Score	References	Strength	Limitation
1.	Word2Vec (avg)	Word	Averaging word vectors	STS-B	Pearson	~0.60	Mikolov et al. (2013)	Simple, fast baseline	No context awareness
2.	GloVe (avg)	Word	Averaging word vectors	STS-B	Pearson	~0.62	Pennington et al. (2014)	Captures global stats	Static embeddings
3.	Seq2Seq (Encoder-Decoder)	Sentence	Seq2Seq with attention	From newspapers & Quora paraphrase dataset translated in Punjabi	BLEU/ Human eval	0.45 / 70%	Singh, A. and Josan, G., S. (2021)	Captures sequential context	Less accurate for complex sentences
4.	BERT	Contextual	CLS / pooled output	STS-B	Pearson	~0.63	Devlin et al. (2019)	Context-aware	Not optimized for similarity
5.	RoBERTa	Contextual	Sentence pooling	STS-B	Pearson	~0.66	Liu et al. (2019)	Improved BERT	Computationally heavy
6.	SBERT	Sentence	Siamese architecture	STS-B	Pearson	~0.85	Reimers & Gurevych (2019)	Optimized for similarity	Needs fine-tuning
7.	SRoBERTa	Sentence	Fine-tuned SBERT	STS-B	Pearson	~0.86	Reimers et al. (2019)	High accuracy	Large model
8.	SimCSE	Sentence	Contrastive learning	STS-B	Pearson	~0.82	Gao et al. (2021)	Efficient & strong	Needs training
9.	LaBSE	Multilingual	Transformer-based	Tatoeba / STS	Accuracy	~0.88	Feng et al. (2022)	Strong cross-lingual	Large size

Table 5: Performance of Sentence Embedding Models Sentence Similarity Tasks

XII. THE SENTENCE EMBEDDING MODELS FOR LOW RESOURCE LANGUAGES (PUNJABI)

The below table presents the various sentence embedding models available for low resource language like Punjabi. The traditional models require huge dataset for training. The latest models are good for similarity related tasks as shown in the previous section. mBERT, MiniLM and LaBSE are trained on huge dataset of Punjabi and user can follow the models. The finetuning a particular model is open for improving the results in low resource languages.

S. No	Model	Type	Dataset (Pretraining / Usage)	Multilingual Support	Pretrained for Punjabi	Fine-tuning Required	Reference	Strength	Limitation
1	Word2Vec	Word	Requires large Punjabi corpus (news, Wikipedia)	No	No	Yes	Mikolov et al. (2013)	Simple, fast baseline	Needs large data, no context awareness
2	GloVe	Word	Large co-occurrence corpora	No	No	Yes	Pennington et al. (2014)	Captures global statistics	Static embeddings
3	FastText	Word	Wikipedia + Common Crawl	Yes	Partial	Optional	Joulin et al. (2017)	Handles subwords (good for Punjabi)	Limited context
4	Seq2Seq (Encoder–Decoder)	Sentence	Punjabi paraphrase / translation datasets	No	No	Yes	Singh & Josan (2021)	Captures sequential context	Task-specific, data-heavy
5	BERT (English)	Contextual	BooksCorpus + Wikipedia	No	No	Yes	Devlin et al. (2019)	Strong contextual understanding	Not suitable for Punjabi directly
6	mBERT	Contextual	Multilingual Wikipedia (100+ languages)	Yes	Yes	Optional	Devlin et al. (2019)	Multilingual capability	Uneven performance
7	SBERT	Sentence	NLI + STS datasets	Mostly English	No	Yes	Reimers & Gurevych (2019)	High similarity accuracy	Needs multilingual tuning
8	MiniLM (Multilingual)	Sentence	Multilingual corpora (distilled models)	Yes	Yes	Optional	Wang et al. (2020)	Lightweight and efficient	Slightly lower accuracy
9	LaBSE	Sentence	Large multilingual parallel corpora	Yes	Yes	Optional	Feng et al. (2022)	Strong cross-lingual embeddings	Large model

Table 6: The Sentence Embedding Models for Low Resource Languages (Punjabi).

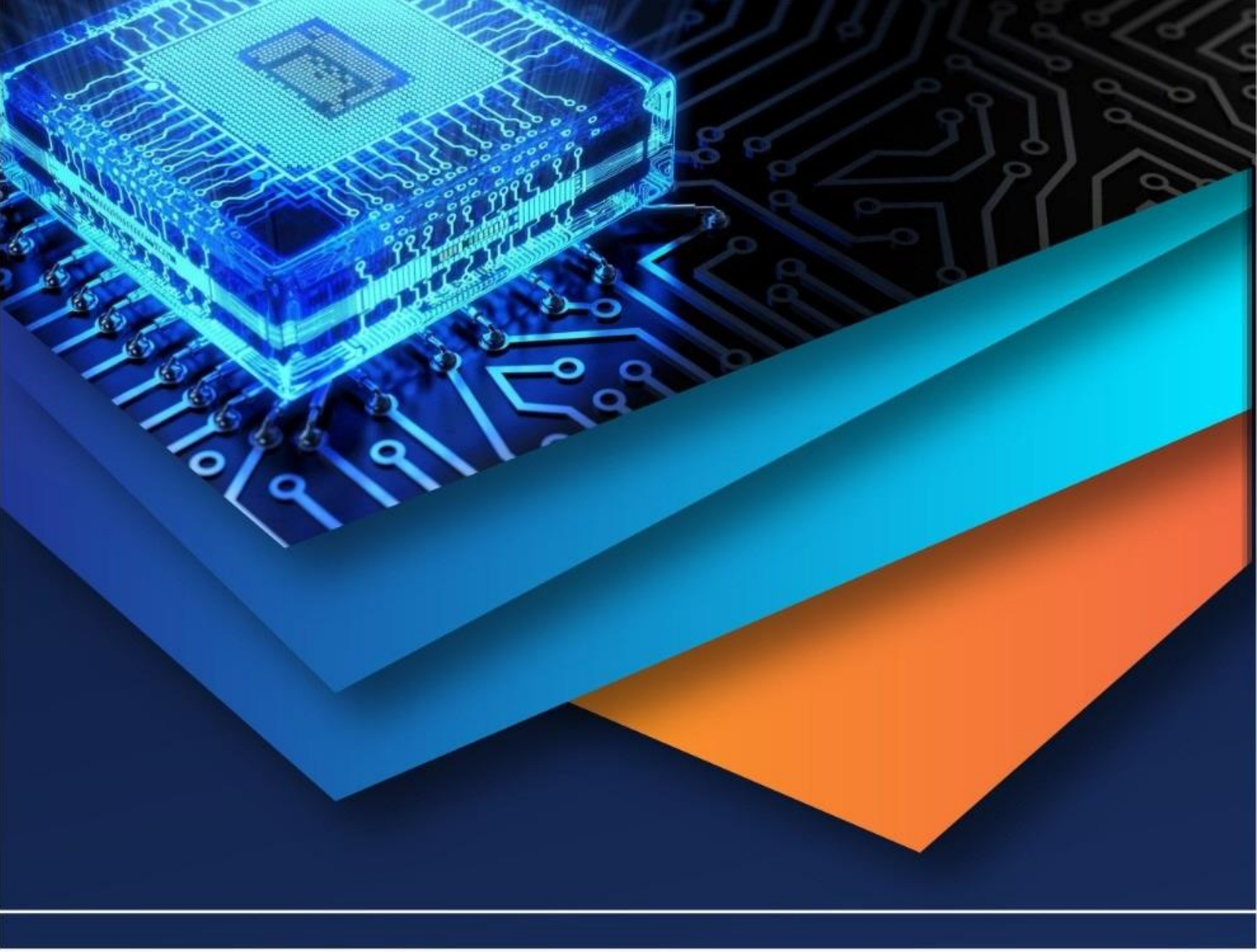
XIII. CONCLUSION AND FUTURE DIRECTION

The embeddings play an important role in semantic similarity tasks such as sentiment analysis, question-answering, text retrieval and paraphrase detection. This paper presented a review of word and sentence embedding approaches from traditional to modern deep learning-based approaches. The traditional approaches are one-hot encoding, count vectors and TF-IDF which are known as the base approaches to convert text as vectors but these are suffered from high dimensionality and lack of semantic understanding. The paper then presented prediction-based models such as Word2Vec and GloVe which was the big advancement in the field of NLP. These models captured the semantic relationship between the text as they produced dense vectors. The study further discussed sentence embedding approaches, where models such as RNN-based Seq2Seq, Universal Sentence Encoder, Sentence-BERT, and SimCSE explained which are capable to capture contextual and semantic relationship at the sentence level. In another section, a comparative analysis and evaluation metrics demonstrated that transformer-based and contrastive learning approaches significantly outperform traditional methods in semantic similarity tasks. An advantage of this paper is that it included practical example and a case study using Punjabi sentences. The section just focused on the process of transforming raw text as embeddings. The analysis of models shown that traditional models are les powerful for obtaining semantic representation and faced issues like dimensionality. Whereas mBERT, MiniLM and LaBSE are the multilingual pretrained models are better for preserving semantic representation. A section on recent trends shown that there are various available models for sentence embeddings. The latest models are good enough for various NLP tasks but with some fine-tuning options available for better results.

As the study explored, traditional embedding approaches was useful as the baseline techniques, on the other hand, transformer based and multilingual models provide an effective solution to capture semantic relationship. Such models are also good for low resource language like Punjabi. The future directions for the development of scalable, efficient and context-aware embedding models for low-resource as well as for multilingual applications.

REFERENCES

[1] A. M. Turing. 1950. Computing Machinery and Intelligence. Mind 59, 236 (1950), 433–460.
[2] Vaibhav Kumar, Jagdish Prasad, and Baldev Singh. 2021. Convolutional Neural Network for Classification for Indian Jewellery. In Proceedings of the International Conference on Sustainable Computing in Science, Technology & Management (SUSCOM-2019).
[3] Tom M. Mitchell. 1997. Machine Learning. McGraw-Hill.
[4] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. Nature 521, 7553 (2015), 436–444.
[5] Yoav Goldberg. 2017. Neural Network Methods for Natural Language Processing. Morgan & Claypool.
[6] Daniel Jurafsky and James H. Martin. 2026. Speech and Language Processing (3rd ed. draft). Stanford University.
[7] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. Information Processing & Management 24, 5 (1988), 513–523.
[8] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In Proceedings of ICLR Workshop.
[9] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In Proceedings of EMNLP.
[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of NAACL-HLT.
[11] A. Singh and G. S. Josan. 2021. A deep network model for paraphrase detection in Punjabi. In Recent Innovations in Computing (ICRIC 2020), Lecture Notes in Electrical Engineering, Vol. 701. Springer, Singapore, 173–185.
[12] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In Proceedings of CoNLL, 10–21.
[13] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems (NeurIPS), 3104–3112.
[14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural Computation 9, 8 (1997), 1735–1780.
[15] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In Advances in Neural Information Processing Systems (NeurIPS), 3294–3302.
[16] Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In Proceedings of NAACL-HLT.
[17] Daniel Cer, Yinfei Yang, Shengyi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder. arXiv:1803.11175.
[18] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Proceedings of EMNLP-IJCNLP.
[19] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In Proceedings of EMNLP.
[20] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In Advances in Neural Information Processing Systems (NeurIPS).
[21] Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In Proceedings of EMNLP.
[22] Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic BERT sentence embedding. In Proceedings of ACL.
[23] Daniel Jurafsky and James H. Martin. 2023. Speech and Language Processing.
[24] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. Introduction to Information Retrieval. Cambridge University Press.
[25] Lingfeng Wang, Vivek Kulkarni, and Sergio Verdú. 2019. On the intrinsic and extrinsic evaluation of word embeddings. In Proceedings of AAAI.
[26] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, et al. 2020. Language models are few-shot learners. In Advances in Neural Information Processing Systems (NeurIPS).
[27] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. arXiv:1907.11692.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY