An Unsupervised Learning Embedding Method Based on Semantic Hashing

Javad Hamidzadeh[1*], Mona Moradi [2]

***Abstract*— Embedding learning is an essential issue in Natural Language Processing (NLP) applications. Most existing methods measure the similarity between text chunks in a context using pre-trained word embedding. However, providing labeled data for model training is costly and time-consuming. So, these methods face downward performance when limited amounts of training data are available. This paper presents an unsupervised sentence embedding method that effectively integrates semantic hashing into the Kernel Principal Component Analysis (KPCA) to construct embeddings of lower dimensions that can be applied to any domain. The experiments conducted on benchmark datasets highlighted that the generated embeddings are general-purpose and can capture semantic meanings from both small and large corpora.**

Index Terms— Kernel Principal Component Analysis, Natural Language Processing, Semantic Hashing, Sentence Embedding.

I. INTRODUCTION

WORD embedding refers to learning distributed word representations to encode word semantics. It can be applied for various applications such as sentiment analysis, text classification, machine translation, named entity recognition, information retrieval, etc. [1-3]. Supervised word embedding achieves remarkable performance given large amounts of manually labeled data. However, they suffer some limitations: (1) Supervised word embedding approaches require large amounts of labeled data with word-to-meaning associations. Acquiring such labeled data can be expensive and time-consuming, especially for specialized domains or low-resource languages. (2) Supervised word embeddings are often specific to the domain in which they are trained. They may not generalize well to new or different domains, limiting their applicability in diverse contexts. (3) Supervised methods rely on a predefined vocabulary, meaning that they may struggle to handle Out-Of-Vocabulary (OOV) words or rare terms that are not present in the training data. This can limit their usefulness in scenarios where new or specialized vocabulary is encountered. (4) The quality and biases of the labeled data used to train supervised word embeddings can influence the resulting embeddings. If the labeled data contains biases or errors, the embeddings may inherit those biases, leading to biased representations [4].

By leveraging unlabeled data and extracting meaningful representations from it, unsupervised word embedding methods, e.g., Glove [5] and word2vec (Skip-Gram [6] version), overcome these limitations. These methods offer data efficiency, generalization, adaptability, and the potential to capture more nuanced semantic relationships, making them valuable for a wide range of NLP tasks and applications [7].

Although word embedding methods had been achieved great success, sentence embedding is necessary for natural language understanding systems. The methods for sentence representations can be grouped into two main categories [8]:

Neural network-based methods. Based on the idea that words in similar contexts have a similar meaning, Avg. GloVe [9], IS-BERT [10], BERT-flow [11], and SBERT-WK [12] have been proposed using various pre-trained models inspired by neural network language modeling. An unsupervised method for generating domain-independent embeddings [13] was proposed using several recurrent network variants. An unsupervised method for paraphrase detection based on recursive autoencoders [14] was proposed. Overall, by learning from large amounts of text data, these pre-trained models capture the co-occurrence patterns and distributional properties of words, resulting in embeddings that encode meaningful semantic information. Meanwhile, by utilizing pre-trained embeddings as input features, models can leverage the knowledge acquired during pre-training, which often leads to improved performance and faster convergence. However, these pre-trained models have shortcomings [15-17], such as (1) to obtain appropriate performance, extensive text collections are needed for the training phase, (2) the words must have been seen in the training data before embedding, (3) selecting appropriate values for tunable parameters is a time-consuming process, (4) providing high-quality training sets can be difficult, expensive, or even impossible to acquire for some applications. The majority of well-known word embeddings rely on a specified vocabulary, therefore when OOV words need to be processed, they are typically ignored. In addition, (5) the majority of these models do not or, if they do, only partially take into account the morphology of the words contained inside the word vectors. The quality of document representations based on them may suffer as a result. Consequently, it is advantageous to create vector representations without

1- Faculty of computer engineering and information technology, Sadjad University, Mashhad, Iran.

2- Faculty of Electrical and Computer Engineering, Semnan University, Semnan, Iran.
Corresponding author: j_hamidzadeh@sadjad.ac.ir

supervision.

statistical methods, e.g., averaging the embeddings of words in a sentence, is an easy way of obtaining sentence embeddings but some defects, like neglecting the word order, realizing the inexpedient meaning of the sentence, and failure to adopt an appropriate approach when dealing with long sentences, causing utilizing sentence embedding instead of word embedding. Hash-based methods have received much more attention in recent years. Methods presented in [18-20] generate hash codes for short text segments while preserving semantic-level similarities.

Aiming to encode the sentences into the embedding vectors, the proposed method effectively combines the SimHash algorithm with the KPCA in an unsupervised manner; thereby, sentences that share semantic and syntactic properties are mapped into similar vector representations. SimHash is a computationally efficient method that finds applications in various fields, including information retrieval, plagiarism detection, duplicate content detection, and clustering of similar documents [21]. SimHash can be used to encode words as binary vectors (hash codes) in the context of word embedding, where words with similar semantic meanings have comparable binary representations. By comparing their hash values, this feature makes it beneficial for locating identical or almost identical things. It is possible to gauge how similar two SimHash values are by comparing the hamming distance (the number of different bits) between them. The smaller the hamming distance, the more similar the items are considered to be. Besides, KPCA is a nonlinear dimensionality reduction technique that can capture complex nonlinear relationships in the data. In the context of word embedding, KPCA can be used to transform the binary word representations obtained from SimHash into a continuous and low-dimensional space, while preserving their semantic similarities. KPCA enables finding low-dimensional representations that preserve the underlying structure of the data, even when the relationships are nonlinear. Here, KPCA is used to map the sentence embeddings to a lower-dimensional space that aligns better with a target task. This approach allows for the creation of word embeddings that capture semantic similarities between words. It provides a way to represent words in a continuous and low-dimensional space, facilitating downstream natural language processing tasks such as text classification, information retrieval, or semantic analysis.

The advantages of the proposed method are that it can be applied to any size of text chunks: sentences, paragraphs, and documents. The created embeddings are also unresponsive to uncommon or infrequent words. In addition, with respect to the fundamental ideas of locality-sensitive hashing (LSH) methods, with SimHash being a variant, the robustness of the proposed method against small variations or noise can be considered [22].

The remainder of this paper is organized as follows: Section II provides related work. In Section III, the proposed method is described. Several experiments have been conducted on several datasets to evaluate the performance of the proposed method. These results have been discussed in Section IV, followed by a conclusion in Section V.

II. RELATED WORK

Unsupervised methods for word and sentence embedding provide valuable tools for understanding and processing natural language data without relying on labeled annotations. They offer a way to extract meaningful representations from vast amounts of unlabeled text, opening up possibilities for various downstream NLP applications and facilitating research in areas where labeled data is scarce or non-existent. The recent successes of unsupervised embeddings in several applications demonstrate the capability of these methods. Generally, unsupervised learning embedding methods can be categorized into two groups [23].

Unsupervised learning embedding with adversarial training. For word embeddings, significant efforts have been made [24-26], but sentence embeddings remain to be discovered. Previous studies on sentence embedding have computed sentence embeddings by composing word embeddings [27, 28]. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTMs) [29], recursive neural networks [30, 31], and Convolutional Neural Networks (CNNs) as deep learning-based models have a strong ability to capture the hidden correlations among words. Still, high complexity makes them time-consuming, especially for training large datasets. The Word2vec [6] was introduced for learning word vector representations. For representing large textual chunks, such as sentences and paragraphs, a generalized version of Word2vec, named Doc2vec [32] was proposed. Authors in [33] studied on CNNs for learning sentences. Authors in [34] introduced a method for unsupervised learning of a distributed sentence encoder. They trained an encoder-decoder model that tried reconstructing the surrounding sentences of an encoded passage. A sentence encoder model [35] trained on a large corpus was presented. Authors in [36] employed RNN and LSTM architectures.

Although deep learning-based approaches generate suitable vectors for word representation, their performance is reduced when the model is trained on a small dataset due to several reasons [37-40]: (1) Deep learning models often have a large number of parameters, which allow them to capture complex patterns and relationships in the data. With a smaller dataset, there is a higher risk of overfitting, where the model memorizes the training examples instead of learning generalizable patterns. By giving a more varied and representative sample of the underlying data distribution, a larger dataset assists in reducing overfitting. (2) Effective generalisation of new data is a goal of deep learning models. The model can acquire more robust and nuanced representations, capturing a wider range of patterns present in the data, by being exposed to a larger variety of samples. This enhances the model's generalization and performance on fresh, novel examples. (3) Deep learning models often learn hierarchical representations of the input data. Each layer of the network learns to extract progressively more abstract and meaningful features. These representations become more accurate and informative as the model observes a larger and more diverse dataset. With a smaller dataset, the learned representations may not capture the full complexity of

the underlying data distribution. It is necessary to note that while there is no specific threshold for dataset size, it is generally observed that deep learning models tend to benefit from larger datasets. However, the precise impact of dataset size can vary depending on the complexity of the task, the model architecture, and other factors. It's important to strike a balance between the availability of data and the computational resources required to train deep learning models effectively.

Unsupervised learning with similarity distribution. In [41] an unsupervised method for extractive multi-document summarization was proposed based on the centroid approach and the sentence embedding representations. Gupta [42] proposed an unsupervised model that allows composing sentence embeddings using word vectors and *n*-gram embeddings. In [43], Sent2Vec1 was introduced which is an unsupervised model allowing composing sentence embeddings using word vectors and *n*-gram embeddings. Besides, an unsupervised sentence embedding method [44] was proposed based on the discourse vectors in the random walk model for generating text.

In conclusion, pre-trained language models based on neural networks have achieved significantly better results in various NLP tasks compared to traditional statistical methods and have revolutionized the field of NLP in recent years [45, 46]. There are some reasons why pre-trained language models have achieved better results: (1) pre-trained language models have the ability to capture a contextual understanding of words and sentences [47]. They learn to represent words based on their surrounding context, allowing them to capture the nuances of language and better handle word sense disambiguation. Traditional statistical methods often rely on simple word co-occurrence statistics or bag-of-words representations, which lack the contextual understanding captured by neural language models. (2) Pre-trained language models are trained on massive amounts of text data, typically on large-scale corpora such as Wikipedia or web text [48]. This extensive pre-training enables the models to learn rich representations of words, phrases, and even entire sentences. In contrast, statistical methods often require manual feature engineering and rely on relatively smaller datasets, limiting their ability to capture complex language patterns. (3) Pre-trained language models can be fine-tuned for specific downstream tasks with smaller task-specific datasets [49]. This transfer learning approach allows the models to leverage the knowledge learned during pre-training and adapt it to specific tasks [50]. This is particularly beneficial when labeled data for a specific task is limited. Traditional statistical methods often require task-specific feature engineering or training from scratch, which can be time-consuming and less effective with limited data. (4) Pre-trained language models can be adjusted or tailored to certain domains or specialized datasets, enabling them to excel even in tasks that are peculiar to those areas. They are extremely adaptable across a variety of fields and applications thanks to this versatility. Traditional statistical methods may require extensive manual feature engineering or specialized models for different domains, making them less flexible and scalable. (5) Pre-trained language models excel at capturing semantic relationships between words and understanding various aspects of language, such as synonymy, antonymy, analogies, and sentence-level coherence. They can generate more meaningful and contextually appropriate word embeddings or sentence representations. Statistical methods, while useful for some tasks, often struggle to capture these higher-level semantic relationships effectively.

Statistical methods have been used in NLP for several reasons, despite the advent and success of neural network-based approaches [51]. Statistical methods often provide more interpretability compared to neural network-based models. They allow researchers and practitioners to understand the underlying statistical principles and assumptions used in the models [52]. This interpretability can be crucial in domains where explainability and transparency are required, such as legal or regulatory contexts [53]. Statistical methods are often simpler in terms of model architecture and implementation. They tend to have fewer parameters and require fewer computational resources compared to neural network-based models. This simplicity makes them computationally efficient and suitable for scenarios with limited computational power or memory constraints. Statistical methods can work well with limited amounts of data. Even with limited datasets or in situations where obtaining huge amounts of labeled data is difficult or expensive, they can nevertheless produce good results. Contrarily, for neural network-based models to perform to their full potential, large-scale training datasets are often necessary. Manual feature engineering is frequently used in statistical approaches, where domain-specific knowledge and skill can be used to create useful features [54]. This feature engineering approach enables users to add certain language or contextual data to the model, improving performance for some jobs. Contrarily, models built on neural networks may automatically learn characteristics from raw data, which may not always capture the desired domain-specific information. Besides, certain applications have specific constraints or requirements that make statistical methods more suitable. For example, in some low-resource or real-time applications, the simplicity, speed, and low memory footprint of statistical methods make them preferable over complex neural network models.

It's important to note that while statistical methods have their advantages, neural network-based approaches have made significant advancements and achieved state-of-the-art results in various NLP tasks. The choice of method depends on factors such as the specific task, available resources, interpretability requirements, and the trade-off between performance and complexity.

III. PROPOSED METHOD

In this section, a new unsupervised method for sentence embedding is introduced by incorporating the SimHash algorithm into the KPCA.

As shown in Fig. 1, the pre-processed sentences are extracted from each document and accumulated into a comprehensive set of unique sentences. The sentence pre-processing involves several steps such as tokenization, stemming, removing stop words, and lowercasing each

sentence. Next, the task of generating hash binary codes (fingerprints) is done for each word of a sentence using the SimHash algorithm. SimHash, as one of the extensions of the Locality Sensitive Hashing (LSH) [55] algorithm for large-scale real-valued data, was first introduced by Charikar [56].

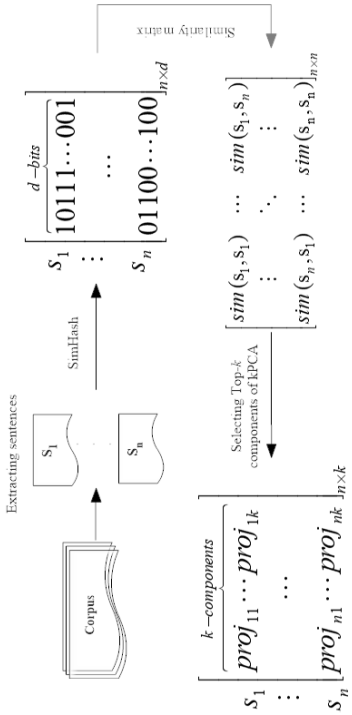

Fig. 1. The proposed sentence embedding pipeline.

SimHash is designed to produce similar hash values for similar inputs, making it suitable for identifying similar items based on their hashes. It produces a compact representation of high-dimensional vectors by using random projections. Then, it applies a weighting scheme to each feature and combines them to create a binary fingerprint. The fingerprint is obtained by comparing the weighted feature values with a threshold and setting the corresponding bit in the hash value based on the result of the comparison.

For word w, the SimHash function is defined as follows.

$$h(w) = sign(v^t w) = \begin{cases} +1 & if\ v^t w \geq 0 \\ -1 & Otherwise \end{cases} \quad (1)$$

where v^t is the transpose of the randomly generated vector v. Now, given a set containing w words extracted from a sentence, SimHash is used to generate d-bit fingerprints $h \in H \equiv \{-1,1\}^d$; meanwhile, simple word averaging is performed to obtain sentence embeddings.

Afterward, given a set containing n sentences extracted from a document, the similarities between sentences in the original spaces and transformed spaces are calculated using similarity function presented in Eq. (2):

$$sim(s_i, s_j) = e^{-dist(s_i, s_j)};\ \forall i, j \in \mathbb{R}^n \quad (2)$$

where s_i represents a vector $s_i = [s_{i1}, s_{i2}, \cdots, s_{id}]$, where d is the number of the bits of the fingerprints and *dist* is the Hamming distance of two fingerprints. The similarity matrix **sim** contains n n-dimensional elements. This matrix is obtained by computing Eq. (2) to all fingerprint pairs, as shown below.

$$\mathbf{sim} = \begin{bmatrix} sim(s_1, s_1) & \cdots & sim(s_1, s_n) \\ \vdots & \ddots & \vdots \\ sim(s_n, s_1) & \cdots & sim(s_n, s_n) \end{bmatrix}_{n \times n}$$

To find the distributed representation of the sentences, KPCA is employed. By exploiting PCA, obtained fingerprints from the previous step can be converted to the low dimensional-vector space where the dimensions are features that describe semantic properties.

Using the kernel trick and nonlinear mapping function ϕ, PCA maps each element of the similarity matrix **sim** onto a high-dimensional feature space to obtain the principal components from that space [57]. The term 'kernel' refers to the dot product of the projections of the $sim(s_i, s_j)$ under ϕ.

$$\kappa(s_i, s_j) = \phi(s_i)\phi(s_j)^T \quad (3)$$

The kernel matrix $\boldsymbol{\kappa}$ is computed by applying a nonlinear kernel function to the similarity matrix **sim**. The proposed method employs the Gaussian Radial Basis Function (RBF) as a kernel.

$$\kappa(s_i, s_j) = e^{-\gamma \|s_i - s_j\|_2^2} \quad (4)$$

$$\gamma = \frac{1}{2\sigma^2} \quad (5)$$

where σ is the variance of the data. If the data has a high variance, it suggests that the data points are spread out and have a wide range of values. In such cases, a smaller γ value may be suitable to capture the global structure and smooth out the representations. This prevents overfitting to local patterns and ensures that the embeddings capture the broader trends and variations in the data. On the other hand, if the data has low variance, it implies that the data points are more tightly clustered or have limited variation. In such scenarios, a larger γ value may be appropriate to capture finer details and localized patterns in the data.

The RBF kernel offers several benefits when used in KPCA:
(1) The RBF kernel has the property of universal approximation, which means it can approximate any continuous function given enough basis functions. This property allows the RBF kernel to represent complex and diverse data patterns accurately. By employing a suitable number of basis functions, KPCA with RBF kernel can effectively model and approximate the underlying structure of the data.
(2) The RBF kernel allows KPCA to capture nonlinear relationships in the data. It maps the original data into a higher-dimensional feature space, where linearly inseparable patterns can be separated more effectively. This flexibility is particularly useful when dealing with complex and nonlinear data distributions, enabling KPCA to discover hidden structures and capture intricate relationships.
(3) The RBF kernel produces smooth and continuous representations in the feature space. It assigns higher weights to points that are closer together, gradually decreasing their

influence as the distance increases. This characteristic ensures that nearby data points have similar representations, preserving local relationships and maintaining the overall continuity of the embeddings.

(4) The RBF kernel is less sensitive to outliers compared to other kernels, such as the linear kernel. Outliers typically have less influence on the resulting embeddings when using the RBF kernel, allowing KPCA to focus more on the underlying data distribution. This robustness to outliers helps in obtaining more reliable and stable representations, particularly when dealing with noisy or contaminated data.

To obtain principal components in the kernel space, the covariance of the kernel matrix $\boldsymbol{\kappa}$ should be maximized by solving Eq. (6).

$$C\nu = \lambda\nu \tag{6}$$

where C is the covariance matrix (see Eq. (7)), λ is an arbitrary eigenvalue, and ν is its corresponding eigenvector.

$$C = \frac{1}{n}\boldsymbol{\kappa}\boldsymbol{\kappa}^T \tag{7}$$

Combining Eqs. (7) and (4), and substituting into Eq. (6), Eq. (8) is taken as follows:

$$\boldsymbol{\kappa}\alpha = n\lambda\alpha \tag{8}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_n)^T$ denotes the column vector such that the orthogonal eigenvector ν of the covariance matrix C satisfies

$$\nu = \sum_{i=1}^{n} \alpha_i \phi(x_i) \tag{9}$$

To obtain top-k principal components, let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k \geq \cdots \geq \lambda_n$ show the first k eigenvalues of $\boldsymbol{\kappa}$ and $\alpha_1, \alpha_2, \cdots, \alpha_n$ the corresponding complete set of eigenvectors obtained by Eq. (8). The corresponding eigenvectors $\nu_1, \nu_2, \cdots, \nu_k$ of matrix C are calculated by Eq. (9). At the end of the proposed method, the employed projection matrix $\boldsymbol{P}$ is obtained by selecting k eigenvectors and dividing them by the corresponding eigenvalues as follows:

$$\boldsymbol{P} = \left[\frac{\nu_1}{\lambda_1}, \frac{\nu_2}{\lambda_2}, \cdots, \frac{\nu_k}{\lambda_k}\right] \tag{10}$$

Now, arriving a new sentence s_t, first, it is converted to the d-bit SimHash code; then, the similarity function of this sentence against all other sentences is calculated by considering the kernel function as follows:

$$\tau_t = \kappa(sim(s_t, s_i)); \ \ \forall i \in \mathbb{R}^n \tag{11}$$

The product of τ_t with the projection matrix $\boldsymbol{P}$ generates the k-dimensional KPCA embedding $\rho_t = \boldsymbol{P}^T\tau_t$.

IV. EXPERIMENTAL RESULTS

In this section, the experiments for the evaluation of the proposed method are described. Some experiments have been conducted over well-known real-world datasets. The selected datasets are introduced in subsection *A*. The evaluation procedure is covered in subsection *B*. The experimental results and their interpretations are described in subsection *C*.

A. Datasets

The proposed method is evaluated on widely-used datasets STSB (Semantic Textual Similarity Benchmark) [36], SICK (Sentences Involving Compositional Knowledge) [37], and STS (Semantic Textual Similarity) tasks 2012 - 2016 [38-42]. These datasets consist of sentence pairs with labeled semantic similarity scores ranging from 0 to 5, where a higher score means higher semantic similarity. The STSB dataset has been used as part of shared tasks in the SemEval (Semantic Evaluation) series, which is a series of workshops focused on semantic analysis and NLP tasks. It contains around 7,500 sentence pairs. The SICK dataset consists of approximately 10,000 sentence pairs across multiple tasks, including semantic relatedness, paraphrase detection, and textual entailment. The STS datasets have been utilized in various SemEval shared tasks. The STS 2012, STS 2013, and STS 2014 datasets contain 1,500 sentence pairs. The STS 2015 dataset comprises a total of 5,000 sentence pairs across the various subtasks including STS MSRvid (video descriptions), STS MSRpar (online article snippets), STS OnWN (WordNet glosses), STS SMTeuroparl (Europarl sentences), and STS Tweet (Twitter messages). The STS 2016 dataset comprises a total of 12,304 sentence pairs across the different subtasks including answer (sentence pairs from community question-answering websites), headlines (news headlines), plagiarism detection, and question-question similarity.

B. Evaluation Procedure

Inspired by [58], Procedure 1 is carried out to evaluate the performance of the proposed method. Accordingly, the cosine similarity of each pair of the sentence embedding and the gold label is calculated. For a pair, golden semantic similarity ranges from 0.0 to 5.0, while cosine similarity ranges from -1.0 to 1.0. To overcome the range difference, rank-based Spearman's correlation coefficient is employed.

PROCEDURE I
EVALUATION PROCEDURE

For each sentence pair **do**
1. Perform pre-processing stage (tokenization, stemming, removing stop words, and lowercasing each sentence).
2. Derive sentence embeddings by the proposed method.
3. Compute the cosine similarity score as the predicted similarity.
4. Compute Spearman's rank correlation coefficient between the predicted similarity and gold standard similarity scores as the evaluation metric.

End for

Spearman's rank correlation is a statistical measure that quantifies the strength and direction of the monotonic relationship between two variables. It focuses on the ordinal relationship, where the variables may not have a linear association but still exhibit a consistent ordering pattern. Spearman's correlation coefficient is calculated by first ranking the values of each variable and then calculating the Pearson

correlation coefficient between the ranks. It ranges between -1 and 1, where:

- A coefficient of +1 indicates a perfect monotonic increasing relationship, where higher ranks of one variable are consistently associated with higher ranks of the other variable.
- A coefficient of -1 indicates a perfect monotonic decreasing relationship, where higher ranks of one variable are consistently associated with lower ranks of the other variable.
- A coefficient of 0 indicates no monotonic relationship between the variables.

Spearman's correlation coefficient is useful when dealing with non-linear relationships or variables measured on an ordinal or ranked scale. It is less sensitive to outliers compared to Pearson's correlation coefficient and can provide insights into the direction and strength of the relationship, even if the relationship is not strictly linear.

C. Results

For evaluation, the following sentence embedding methods are considered competing methods.

- TF-IDF-SVD. A sentence is represented by a sparse vector of TF-IDF. Then, the vector is reduced by SVD. This method can uncover latent semantic relationships in the text data. The reduced-dimensional representation may reveal hidden associations and similarities between terms and documents that are not immediately apparent in the original space.
- Smooth Inverse Frequency (SIF) [44]. It is an unsupervised sentence embedding that employs a weighted average of the word vectors. Then, modify them using PCA/SVD. SIF incorporates Inverse Document Frequency (IDF) weighting to down-weight frequently occurring words in the embedding process. This helps to mitigate the impact of common and less informative words, enabling the focus on more important and meaningful terms in the text. SIF does not explicitly consider the word order or sentence structure in the embedding process. It treats each sentence as a bag of words, potentially limiting its ability to capture nuanced semantic relationships that depend on word order or syntax. SIF focuses primarily on individual word embeddings and their IDF weighting. It does not explicitly capture the contextual information or relationships between words within a sentence, which can be crucial for certain text-understanding tasks.

Table I displays Spearman's rank correlation of the methods. As observed, the proposed method attains the highest Spearman's rank correlation for STS12 and the lowest Spearman's rank correlation for STSB. In Table II, the Pearson correlation values of the methods are presented. The results demonstrate that the proposed method achieves the highest correlation values for all datasets when compared with the competing methods.

TABLE I
Spearman's Rank Correlation

Method	STSB	SICK	STS12	STS13	STS14	STS15	STS16
TF-IDF-SVD	31.03	42.13	52.92	49.57	41.31	31.23	52.12
SIF(PCA)	56.97	51.04	57.31	58.03	58.27	50.82	59.65
Proposed method	61.25	62.35	62.18	69.63	61.33	61.15	62.19

TABLE II
Pearson Correlation Values

Method	STSB	SICK	STS12	STS13	STS14	STS15	STS16
TF-IDF-SVD	36.05	44.28	57.29	48.82	41.18	34.83	53.23
SIF(PCA)	58.37	50.16	60.26	58.29	57.97	52.17	58.82
Proposed method	63.17	64.23	64.03	70.38	62.60	63.20	61.17

D. Time complexity

The time complexity of SimHash can be approximated as $O(d)$. This is because SimHash involves iterating over each feature or dimension of the data item and performing bitwise operations. It scales linearly with the number of dimensions, making it computationally efficient for high-dimensional data.

The time complexity of KPCA depends on the number of data points (n) and the dimensionality of the data (d). The primary computational bottleneck of KPCA lies in the computation of the kernel matrix. The time complexity for computing the kernel matrix in KPCA is $O(n^2 d)$. This is because, for each pair of data points, a kernel function needs to be evaluated, resulting in n^2 evaluations. Furthermore, each evaluation typically involves computing the dot product between two d-dimensional vectors, which has a time complexity of $O(d)$. After computing the kernel matrix, KPCA performs eigenvalue decomposition or singular value decomposition (SVD) on the matrix. The time complexity of eigenvalue decomposition or SVD is $O(n^2 d)$.

Overall, the time complexity of the proposed can be approximated as $O(d) + O(n^2 d) + O(n^2 d)$.

V. CONCLUSION AND FUTURE WORK

For embedding learning tasks, techniques like Word2Vec, GloVe, or BERT are commonly used, as they can preserve semantic and syntactic relationships in a more nuanced way. However, they cannot represent well on rare words with little contextual information. To solve this issue, this paper proposes a new unsupervised method for learning sentence embeddings

and generating new vectors for unseen sentences or even for sentences that contain deliberately obfuscated words. The proposed method simplifies the sentence embedding computation by integrating SimHash with the KPCA. Regarding the KPCA, the sentence representations are transformed into a lower-dimensional space while preserving nonlinear relationships.

Employing SimHash is beneficial since it is computationally efficient, especially when dealing with large datasets. It can accurately estimate the similarity between two inputs based on the hamming distance between their hash values. However, SimHash generates fixed-length hash values, which may lead to a loss of granularity in representing the original data. Fine-grained differences between inputs might be disregarded in the hash representation. Also, SimHash does not capture the semantic meaning of words or the context in which they appear. It treats each word or feature independently, potentially leading to limitations in applications where a deeper understanding of language semantics is required. While SimHash can tolerate some level of noise, excessive noise or significant perturbations in the data may adversely affect its performance. It's essential to preprocess the data appropriately and ensure that the noise level is within reasonable limits for SimHash to provide accurate similarity measurements.

Experiments performed on seven benchmark datasets have shown that it gains the highest Spearman's rank for STS12 and the lowest for STSB datasets.

In future work, the authors intend to focus on weighted averaging for representing short phrases. Furthermore, since different kernels may produce diverse embeddings and capture various aspects of the data, kernel learning will be another issue that should be investigated.

VI. REFERENCES

[1] J.E. Font, M.R. Costa-Jussa, Equalizing gender biases in neural machine translation with word embeddings techniques, arXiv preprint arXiv:1901.03116, (2019).

[2] R.A. Stein, P.A. Jaques, J.F. Valiati, An analysis of hierarchical text classification using word embeddings, Information Sciences, 471 (2019) 216-232.

[3] E. Biswas, K. Vijay-Shanker, L. Pollock, Exploring word embedding techniques to improve sentiment analysis of software engineering texts, in: Proceedings of the 16th International Conference on Mining Software Repositories, IEEE Press, 2019, pp. 68-78.

[4] F. Incitti, F. Urli, L. Snidaro, Beyond word embeddings: A survey, Information Fusion, 89 (2023) 418-436.

[5] R. JeffreyPennington, C. Manning, Glove: Global vectors for word representation, in: Conference on Empirical Methods in Natural Language Processing, 2014.

[6] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in neural information processing systems, 2013, pp. 3111-3119.

[7] D.S. Asudani, N.K. Nagwani, P. Singh, Impact of word embedding models on text analytics in deep learning environment: a review, Artificial Intelligence Review, (2023) 1-81.

[8] J. Qiang, F. Zhang, Y. Li, Y. Yuan, Y. Zhu, X. Wu, Unsupervised statistical text simplification using pre-trained language modeling for initialization, Frontiers of Computer Science, 17 (2023) 171303.

[9] J. Pennington, R. Socher, C.D. Manning, Glove: Global vectors for word representation, in, pp. 1532-1543.

[10] Y. Zhang, R. He, Z. Liu, K.H. Lim, L. Bing, An unsupervised sentence embedding method by mutual information maximization, arXiv preprint arXiv:2009.12061, (2020).

[11] B. Li, H. Zhou, J. He, M. Wang, Y. Yang, L. Li, On the sentence embeddings from pre-trained language models, arXiv preprint arXiv:2011.05864, (2020).

[12] B. Wang, C.C.J. Kuo, Sbert-wk: A sentence embedding method by dissecting bert-based word models, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 28 (2020) 2146-2157.

[13] J. Wieting, M. Bansal, K. Gimpel, K. Livescu, Towards universal paraphrastic sentence embeddings, arXiv preprint arXiv:1511.08198, (2015).

[14] R. Socher, E.H. Huang, J. Pennin, C.D. Manning, A.Y. Ng, Dynamic pooling and unfolding recursive autoencoders for paraphrase detection, in: Advances in neural information processing systems, 2011, pp. 801-809.

[15] B. Min, H. Ross, E. Sulem, A.P.B. Veyseh, T.H. Nguyen, O. Sainz, E. Agirre, I. Heinz, D. Roth, Recent advances in natural language processing via large pre-trained language models: A survey, arXiv preprint arXiv:2111.01243, (2021).

[16] S. Li, X. Puig, C. Paxton, Y. Du, C. Wang, L. Fan, T. Chen, D.-A. Huang, E. Akyürek, A. Anandkumar, Pre-trained language models for interactive decision-making, Advances in Neural Information Processing Systems, 35 (2022) 31199-31212.

[17] R.K. Kaliyar, A multi-layer bidirectional transformer encoder for pre-trained word embedding: a survey of bert, in, IEEE, pp. 336-340.

[18] M.S. Charikar, Similarity estimation techniques from rounding algorithms, in, 2002, pp. 380-388.

[19] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, science, 313 (2006) 504-507.

[20] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing, Advances in neural information processing systems, 21 (2008).

[21] Y. Li, F. Liu, Z. Du, D. Zhang, A simhash-based integrative features extraction algorithm for malware detection, Algorithms, 11 (2018) 124.

[22] J. Leskovec, A. Rajaraman, J.D. Ullman, Mining of massive data sets, Cambridge university press, 2020.

[23] F. Hill, K. Cho, A. Korhonen, Learning distributed representations of sentences from unlabelled data, arXiv preprint arXiv:1602.03483, (2016).

[24] T. Mikolov, W.-t. Yih, G. Zweig, Linguistic regularities in continuous space word representations, in: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2013, pp. 746-751.

[25] O. Levy, Y. Goldberg, Linguistic regularities in sparse and explicit word representations, in: Proceedings of the eighteenth conference on computational natural language learning, 2014, pp. 171-180.

[26] S. Arora, Y. Li, Y. Liang, T. Ma, A. Risteski, Linear algebraic structure of word senses, with applications to polysemy, Transactions of the Association of Computational Linguistics, 6 (2018) 483-495.

[27] W. Blacoe, M. Lapata, A comparison of vector-based representations for semantic composition, in: Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning, Association for Computational Linguistics, 2012, pp. 546-556.

[28] J. Mitchell, M. Lapata, Vector-based models of semantic composition, proceedings of ACL-08: HLT, (2008) 236-244.

[29] K.S. Tai, R. Socher, C.D. Manning, Improved semantic representations from tree-structured long short-term memory networks, arXiv preprint arXiv:1503.00075, (2015).

[30] R. Socher, B. Huval, C.D. Manning, A.Y. Ng, Semantic compositionality through recursive matrix-vector spaces, in: Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning, Association for Computational Linguistics, 2012, pp. 1201-1211.

[31] R. Socher, A. Perelygin, J. Wu, J. Chuang, C.D. Manning, A. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the 2013 conference on empirical methods in natural language processing, 2013, pp. 1631-1642.

[32] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: International conference on machine learning, 2014, pp. 1188-1196.

[33] N. Kalchbrenner, E. Grefenstette, P. Blunsom, A convolutional neural network for modelling sentences, arXiv preprint arXiv:1404.2188, (2014).

[34] R. Kiros, Y. Zhu, R.R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, S. Fidler, Skip-thought vectors, in: Advances in neural information processing systems, 2015, pp. 3294-3302.

[35] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, A. Bordes, Supervised learning of universal sentence representations from natural language inference data, arXiv preprint arXiv:1705.02364, (2017).

[36] S.R. Bowman, G. Angeli, C. Potts, C.D. Manning, A large annotated corpus for learning natural language inference, arXiv preprint arXiv:1508.05326, (2015).

[37] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.

[38] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Communications of the ACM, 60 (2017) 84-90.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems, 30 (2017).

[40] T.J. Sejnowski, The unreasonable effectiveness of deep learning in artificial intelligence, Proceedings of the National Academy of Sciences, 117 (2020) 30033-30038.

[41] S. Lamsiyah, A. El Mahdaouy, B. Espinasse, S. El Alaoui Ouatik, An unsupervised method for extractive multi-document summarization based on centroid approach and sentence embeddings, Expert Systems with Applications, 167 (2021) 114152.

[42] P. Gupta, Unsupervised learning of sentence embeddings using compositional n-gram features, in, 2018.

[43] M. Pagliardini, P. Gupta, M. Jaggi, Unsupervised learning of sentence embeddings using compositional n-gram features, arXiv preprint arXiv:1703.02507, (2017).

[44] S. Arora, Y. Liang, T. Ma, A simple but tough-to-beat baseline for sentence embeddings, (2016).

[45] A. Roshanzamir, H. Aghajan, M. Soleymani Baghshah, Transformer-based deep neural network language models for Alzheimer's disease risk assessment from targeted speech, BMC Medical Informatics and Decision Making, 21 (2021) 1-14.

[46] J. Lu, X. Zhan, G. Liu, X. Zhan, X. Deng, BSTC: A Fake Review Detection Model Based on a Pre-Trained Language Model and Convolutional Neural Network, Electronics, 12 (2023) 2165.

[47] Z. Dai, J. Callan, Deeper text understanding for IR with contextual neural language modeling, in, pp. 985-988.

[48] N. Azzouza, K. Akli-Astouati, R. Ibrahim, Twitterbert: Framework for twitter sentiment analysis based on pre-trained language model representations, in, Springer, pp. 428-437.

[49] H. Christian, D. Suhartono, A. Chowanda, K.Z. Zamli, Text based personality prediction from multiple social media data sources using pre-trained language model and model averaging, Journal of Big Data, 8 (2021).

[50] V. Suresh, D.C. Ong, Using knowledge-embedded attention to augment pre-trained language models for fine-grained emotion recognition, in, IEEE, pp. 1-8.

[51] L.K. Şenel, I. Utlu, V. Yücesoy, A. Koc, T. Cukur, Semantic structure and interpretability of word embeddings, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 26 (2018) 1769-1779.

[52] J.J. Lastra-Díaz, J. Goikoetxea, M.A.H. Taieb, A. García-Serrano, M.B. Aouicha, E. Agirre, A reproducible survey on word embeddings and ontology-based methods for word similarity: Linear combinations outperform the state of the art, Engineering Applications of Artificial Intelligence, 85 (2019) 645-665.

[53] A. Bakarov, A survey of word embeddings evaluation methods, arXiv preprint arXiv:1801.09536, (2018).

[54] V. Lampos, B. Zou, I.J. Cox, Enhancing feature selection using word embeddings: The case of flu surveillance, in, pp. 695-704.

[55] P. Indyk, R. Motwani, Approximate nearest neighbors: towards removing the curse of dimensionality, in: Proceedings of the thirtieth annual ACM symposium on Theory of computing, ACM, 1998, pp. 604-613.

[56] M.S. Charikar, Similarity estimation techniques from rounding algorithms, in: Proceedings of the thirty-fourth annual ACM symposium on Theory of computing, ACM, 2002, pp. 380-388.

[57] B. Schölkopf, A. Smola, K.-R. Müller, Kernel principal component analysis, in: International conference on artificial neural networks, Springer, 1997, pp. 583-588.

[58] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, arXiv preprint arXiv:1908.10084, (2019).