International Journal of Grid and Distributed Computing
Vol. 11, No. 9 (2018), pp.65-80
http://dx.doi.org/10.14257/ijgdc.2018.11.9.06

Semantic Representations in Text Data

Triveni Lal Pa[1*], Madhu Kumari, Tajinder Singh and Mohammad Ahsan

Department of Computer Science and Engineering
National Institute of Technology Hamirpur, Hamirpur (H.P.), India
trivenipal@gmail.com, madhu.jaglan@gmail.com, nith2k14@gmail.com, ahsan.ch23@gmail.com

Abstract

Automatic text mining processes and other sophisticated natural language processing constructs need realistic representations of text/documents which embed semantics efficiently. All the representations work on the notion that every data contains different explanatory factors (attributes). In this article, we exploit these explanatory factors to study and compare various semantic representation methods for text documents. The article critically reviews recent trends in the area of semi-supervised semantic representations, covering cutting-edge methods in distributed representations such as embeddings. This article gives a broad and synthesized description of various forms of text representations, presented in their chronological order ranging from BoW models to the most recent embeddings learning. Conclusively, various findings taken together provide valuable pointers for researchers looking to work in the field of semantic representations. In addition, the article also shows that one need to develop a model for learning universal embeddings in unsupervised/semi-supervised settings that incorporate contextual as well as word-order information, with language independent features and which would be feasible for large dataset.

Keywords: *Word Embeddings, Vector Space Model, Long-Short-Term Memory, Recurrent Neural Networks*

1. Introduction

Broadly, text/documents have two representation approaches. One is syntax-based approach; based on linguistic theory, grammar and logic [1]. Another is semantic approach; based on statistical techniques, past history and its learning. These representations vary in the way they can incorporate and hide more or less the different attributes of variation behind the data. For example, one hot vector with binary entries only shows presence or absence of a word in a particular document and it ignores frequency information. While, vector space model is relatively more informative and depicts how frequent a particular word is, in a given document; as actual frequency of occurrence of a term does give better performance on various predictive methods in machine learning.

Some NLP applications such as grammar checking and statistical machine translation need support from the syntax of the input language. However, for more ambitious goals of text mining like abstractive summarization and text categorization, an essential requirement is a theory of semantic representation using background (contextual) knowledge, and how it is related to and interacts with this representation. This paper systematically surveys issues that researchers in the latter category (*i.e.*, Semantic approach) grapple with.

Received (January 18, 2018), Review Result (May 14, 2018), Accepted (May 25, 2018)
[*] Corresponding Author

Even though logicist approaches and cognitive science approaches are better in semantic representation, they use more linguistic features and need more human intervention. To automate the text processing, we need to look at some statistical approach. We, therefore, in this paper have considered statistical approaches for semantic representations (statistical semantics) with more emphasis.

In conventional approaches like Vector Space Model (VSM) [2], text has been represented as a bag of words (BoW). VSM, in its most basic form, use Boolean entries for each element in the vector to indicate presence or absence of the word in the document. Further, term-frequency (*tf*), term frequency-inverse document frequency (*tf-idf*), point-wise positive mutual information (*PPMI*), etc. were used as weighting factors to capture the notion that all words cannot be equally important in the document. The vector space model considers numerical feature vectors in a Euclidean space. Each word in VSM was treated as independent from other, thus losing the semantic relation between the words. BoW ignores word-order, thus missing important semantic relations between the words. Indeed, researchers, in the text mining community, proposed ingenious solutions to incorporate the semantic relations (word-order) in the vector space model. N-gram statistical language model [3] is one of such attempts. *N*-gram model intended to incorporate semantics by using context word in predicting the target word. The target word is predicted using conditional probability $P(w_n|w_{1:n-1})$. Where, w_n is the target word and words $w_{1:n-1} = w_1, \dots, w_{n-1}$ are called the context. Though, n-gram has been widely used, it has the curse of its dimensionality.

Alternatively, ontologies deal with semantic heterogeneity in structured data and as conceptual models, provide the necessary framework for the semantic representation of textual information. Ontology representation is based on the concept that is the principal link between texts. Ontology links concept labels to their interpretations, *i.e.*, specifications of their meanings, including concept definitions and relations to other concepts [4]. Ontologies find applications in text summarization, document clustering, text annotation, biomedicine, attitude analysis and entity and event Annotation *etc.*, Ontologies need hand-coded knowledge to semantics, so it is more labor intensive and systematically excluded from this study.

Graphs or semantic graphs are proven to be more prominent than ontologies [5]. In a semantic graph or semantic network representation, knowledge is represented by nodes and connections between nodes. Nodes represent things, concepts, and states; whereas labeled edges represent relationships among nodes. Semantic graph is based on the assumption that higher the degree of connection between the words or concepts, more similar the words be in the meaning. Graphical representation of a text document is powerful because it can helpful in most of the operations in text such as topology, relational, statistical *etc*. Though, a semantic network is intuitive, natural, and easy to understand; however, due to the absence of a quantifier in semantic networks, describing the complex relationship is difficult.

Another approach that has gain most attention of all semantic space models and known for its ability to incorporate hidden semantic relations between words/documents is Latent Semantic Analysis (LSA) [6]. Though LSA too uses BoW approach, it proved to be better than basic VSM because of its unique dimensionality reduction algorithm. LSA first forms a term-document matrix using a document collection and then finds its low-rank approximation using singular value decomposition (SVD). LSA has the capability of finding out hidden semantic relations (that's why the name 'latent') even if the two words never co-occurred in the document. The basic idea behind LSA's meaning induction of a word is the aggregate contexts (in which a word does or does not occur), that produce a set of constraints which generates the meaning of the word. Firth (1957) has put this idea as, "you shall know a word by the company it keeps."

LSA has a wide range of applications in psycholinguistic phenomena, from judgments of semantic similarity to judgments of essay quality and discourse comprehension.

Despite its success, LSA has been criticized for its BoW approach which ignores statistical information regarding word-order. As, semantics of a word is not only defined by its context but also it is the sequence of word occurrences that contribute to the overall meaning of the word.

In recent years, low dimensional, dense vectors called "Word embeddings" based on neural networks learning, gaining attention for semantic learning. These methods are quite successful in learning the semantic representation of words. The skip-gram model and continuous bag-of-word model (CBOW) [7] [8] are popular machine learning approaches for learning word representations that can be efficiently trained on large amounts of text data. CBOW has the training objective to combine the representations of surrounding (context) words to predict the word in the middle (target word). Whereas, the skip-gram model trained to predict the source context words based on the target word. Recently, [9] uses word embeddings in another approach that uses spatial distance to show word relatedness known as "Semantic word cloud". This approach better visualizes the semantic relatedness between the words by improving the aesthetic on word layout.

Tensors [10], alternatives to vectors, are multidimensional objects that have been used for information retrieval, document analysis, text categorization, etc. Tensors paved to be more prominent over vectors as dimensionality reduction in VSM is limited by the number of samples, whereas, TSM has no such limit. HOSVD under TSM can reduce the data to any dimension.

The paper is organized in following sections. Next section (Section 2), built the basis for the study, looking back brief history in the text representation. Various representation approaches with more emphasis on distributed representations are presented in Section 3. This section will be very helpful to an informed users as it is a comprehensive description of tasks specific representations with state-of-the-art methodologies in the tasks. Statement of comparison showing where and why one model lagging over other helps researchers in not only conceptual clarity but it will equip them to select appropriate method for their research. Subsection 3.2 is more imperative and it surveys recent trends in the area of semi-supervised semantic representation (*i.e.*, word embeddings). The section contains various neural network based methods like Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), Long-Short-Term Memory (LSTM) and their variants are represented chronologically. Sections 4 of the article contains some of tools/software packages that are publically available as open source or as proprietary software for semantic representations learning to help beginners in the field. Last section (Section 5) concludes, discussing features of different methodologies, open issues and forecasting further researches.

2. Early History

Semantic representation of text has always been a highly desirable and challenging task for researchers of linguistic, psychology, NLP and text mining community. Way back in the 1950s, Charles Osgood and George Miller worked for semantic representations. Though they worked independently on seemingly unrelated problems, Osgood (1952) tackled the problem of representing word meaning and Miller (1951), solves the problem of transition between the words through sequential dependencies between words. In other attempts in 1960s, psycholinguists try to create a representation of information contained in the text data. In the process of providing an explanatory theory for the nature of the representation, they proposed a hypothesis that "the mental encoding of a sentence corresponds to one of its linguistically motivated representations, its syntactic deep structure or its semantic representation". Though this hypothesis was quickly proved incorrect but it leads the foundation for models which showed that text representation is influenced by world knowledge (context) which contribute a lot to semantic knowledge.

3. Semantic Representation Approaches

The literature for semantic representations of text documents is very rich. It includes a number of approaches varying from domain specific semantic representations to more generic semantic representation. Though the categorization of these methods is very difficult, however, we have broadly categorized them on distributional approach (based on distribution words in or across the documents) and distributed approach (or learning perspective). To constrain the study, we have taken unsupervised and or semi-supervised distributed (embeddings) models in detail and left other models giving the only introduction. Thus, the next subsection (Section 3.1), is dedicated to the methods under the distributional approach for semantic representations. Recently, word embeddings are at the center of domain specific semantic representations. We briefly describe the methods under this category in the subsection 3.2.

3.1. Distributional Approaches

The edifice of distributional approach for meaning is built on distributional hypothesis (which states that words that appear in similar context have same semantics). Researchers also have given the name “vector semantics” [11] to the distributional approach of meaning, as it is generally based on VSM or co-occurrence matrix.

Moreover, on distributional perspective, there are a plethora of methods such as VSM, TSM, Phrase-based approach, LSA, Topic Models, Graph-based approach (semantic graphs), Ontology-based approach *etc*. The subsection 3.1.1 discusses the major works in VSM. Ontologies and semantic graphs are excluded from this study, giving few points as an introduction.

3.1.1. Vector Space Model: Text data, in very basic representation model, can be represented as a term-by-document matrix. Where, each row, in the matrix, is a document and each column represents a feature (word). Here, the collective set of features is typically called a dictionary. Therefore, each element in the matrix represents presence or absence of the word in the corresponding document. As a result of this representation, a matrix with binary entries is obtained. Though this model is very simple to understand and claimed less computation time to prepare the data, it uses all the words (weak or similar words) in the text data. As a consequence, a high dimensional sparse matrix is obtained.

Moreover, in many applications, it is good to work with reduced sparseness. In the process, one obvious reduction is stop words removal, because these words hardly contribute anything towards meaning. Frequency is other important information that can be used in reducing the dimension of the matrix. Most frequent words often are most important in terms of meaning representation (other than stop words).

The vector space model uses term frequencies (*tf*) in the document vectors. Raw co-occurrence frequency measure is simple to understand. However, it is skewed and not very discriminative. Therefore, it isn't the best measure of association between words. García-Hernández *et al.*, [12] use *tf-idf* representation in extractive text summarization, with weighted term-frequency and inverse sentence frequency (Table 1).

Table 1. Weighting Factors used in VSM

Weighted Term	**Formula**
Boolean	0 and 1
tf	$f_{ij} = \frac{f_{ij}}{max_i\{tf_{ij}\}}$
idf	$idf_i = \log\frac{N}{df_i}$
tf-idf	$w_{ij} = tf_{ij} \times idf_i = \frac{f_{ij}}{max_i\{tf_{ij}\}} \times \log\frac{N}{df_i}$
PPMI	$PPMI(w,c) = \max(\log\frac{p(w,c)}{p(w)p(c)},0)$

The *tf–idf* weighting scheme able to determine the relevance of a document by calculating the numerical (*tf–idf*) value of all the words in the document. Being frequency based method, it unable to identify the semantics, and it cannot deal with synonymy and polysemy. Synonymy may lead to decrease the cosine similarity value, whereas, polysemy tends to increase the similarity value, leading to a poor recall and poor precision respectively. High dimensionality is another problem which requires high memory space and computational time.

Despite the fact that VSM is not suitable for semantic representation, with some modification, [13] used Concept Vector Representation model for representing the semantics of text to compute word relatedness using Temporal Semantic Analysis (TSA). In the model, instead of representing a word with a vector of unit concepts, authors manipulated vectors of time series, where each time series describes concept dynamics over time. The model is based on the hypothesis that concepts with similar behavior over time, are semantically related. Authors further revealed that this representation with extra temporal dimension could help in extracting the semantic relation between words. Another similar work by [14] used Wikipedia based ESA to compute semantic relatedness for obtaining a more meaningful semantic interpretation of a text fragment.

Since VSM can represent an unformatted text document as simple and formulaic notation, various algorithms which had been used in data mining can be applied without any modification. Because of this advantage, lots of researches on VSM are being actively carried out. Though VSM (in its basic form) is easy to understand and effectively model the text document, it fails in capturing the semantic relation between the terms in the document.

3.1.2. Latent Semantic Analysis: LSA [6] is a semantic space model that addresses the problems with *tf-idf,* of not capturing semantic relations between words. In the process of finding hidden semantics, LSA extract and represent the contextual-usage meaning of words by applying statistical methods on large text corpora. Based on word usages LSA compares two text pieces/documents by calculating semantic similarity between them.

LSA is based on the assumption that words in the documents have some latent semantic structure and that structure must be same in all the synonyms of the word. Also, polysemy must have a variety of different semantic structures. LSA first forms a document-term matrix, using a document collection and then finds its low-rank approximation using Singular Value Decomposition (SVD). SVD decomposes and maps high-dimensional document and query vectors to fewer dimensional (say k-dimensional) semantic space. Semantic similarity between two documents can be determined by the distance between their projected vectors in the concept or semantic space.

For, a given *word*×*contex* (or *word*×*document*) matrix, A $\in \mathbb{R}^{I\times T}$ with, T>I, SVD factorizes A into U, ∑ and V such that [10]:

$$\mathrm{A} = \mathrm{U}\Sigma\mathrm{V}^{T} \tag{1}$$

Where, the matrix U = [u_1, u_2, …, u_I] $\in \mathbb{R}^{I\times I}$ contains the I left singular vectors, V = [v_1, v_2, …, v_T] $\in \mathbb{R}^{T\times T}$ represents the T right singular vectors and $\Sigma \in \mathbb{R}_{+}^{I\times T}$ with nonnegative elements on the main diagonal.

LSA improves the accuracy of text representation by retaining the semantic relations between words and mitigating the influence of synonymy and polysemy. Also, reduced rank by SVD able to map high-dimensional document vectors in VSM to low dimensional latent semantic space, reducing the dimensionality problem for large corpus.

LSA has some limitations too. LSA basically designed for semantic analysis purposed. Therefore, its performance on other downstream tasks is not up to the mark. Secondly, LSA uses SVD for dimensionality reduction which is a mathematical treatment of the matrix and whose physical interpretation is not clearly defined. Also, controlling the SVD processes in large-scale data is difficult due to its high time and space complexity. Thirdly, LSA assumes Gaussian distribution between words and document, whereas reported distribution by is Poisson. An alternative to LSA, Probabilistic Latent Semantic Analysis (PLSA) [15] is reported with better results than LSA.

3.1.3. Probabilistic Latent Semantic Analysis (PLSA): Probabilistic Latent Semantic Analysis (PLSA) [15], inspired and influenced by Latent Semantic Analysis (LSA) [6], used for learning a semantic representation of a word in its context of use, without using any thesaurus. Thus, it is helpful in disiguating polysemous words. It groups together the words with common context to reveals topical similarity.

Unlike matrix transformation in LSA, PLSA uses generative model. Thus, it tries to approximate the exact solution using approximate calculation method. PLA uses iterative Expectation Maximization (EM) algorithm for constructing mapping which makes it more efficient. All probabilistic topic models work on the assumption that a document is mixed distribution of topics and each topic is probability distribution of words. Assumptions in PLSA topic f_k is a distribution over a fixed size of vocabulary V. Thus, f_k is essentially a vector that each element $f(k,w)$ represents the probability that term w is chosen by topic k, namely:

$$p(w \mid k) = f(k \mid w) \tag{2}$$

PLSA effectively solve the problem of synonyms and polysemes by using EM algorithm to train latent classes. EM algorithm reduces time complexity thus, raising computing speed [16] and perform better than SVD. The disadvantage of PLSA is that it suffers from overfitting. Also, due to unsupervised nature, convergence rate while EM algorithm iterates in PLSA, is very slow.

3.2. Distributed Approaches

Distributional approaches, works on word count on co-occurrence window based on distributional hypothesis. These models are simple to train being count-based, but rare n-grams can be poorly estimated. Moreover, most of these methods represent a semantic vector of a word treating word in isolation and failed to incorporate complex semantic relations between the words. Another class of semantic representations that work on word prediction is often called as distributed representations due to their distributed features in vector space. This section of the paper covers methods under distributed representations.

Word embeddings have been successfully implemented in language models, extractive summarization, machine translation, named entity recognition, disambiguation, parsing, semantic word cloud and social media mining [17].

3.2.1. Character Embeddings: Rather than using vectors corresponding to words, many researchers proposed character level vector representations. Mikolov *et al.*, [18] used characters, syllables and frequent words in subword language model. This model is useful in handling out-of-vocabulary words. Another model character to word (C2W) proposed by Ling *et al.*, [19] used for word representations. Words are composed of vectors of character, helpful in language modeling and POS tagging. Character-Aware Neural Language Model [20] is another approach that employs a convolutional neural network (CNN) and a highway network over characters with LSTM neural network language model. Authors analytically proved that their character model been able to encode both semantic and orthographic information.

3.2.2. Word Embeddings (Word2Vec): Word embeddings popularized by word2vec, a family of energy-based modes [7] [8]. Word2vec internally use the continuous bag-of-word (CBOW) model and skip-gram model for learning high-quality word vectors that can be efficiently trained on large amounts of text data. The target word in the CBOW has been predicted by combining the representations of surrounding (context) words and predicts the word in the middle. Alternatively, the skip-gram model trained to predict the source context words based on the target word.

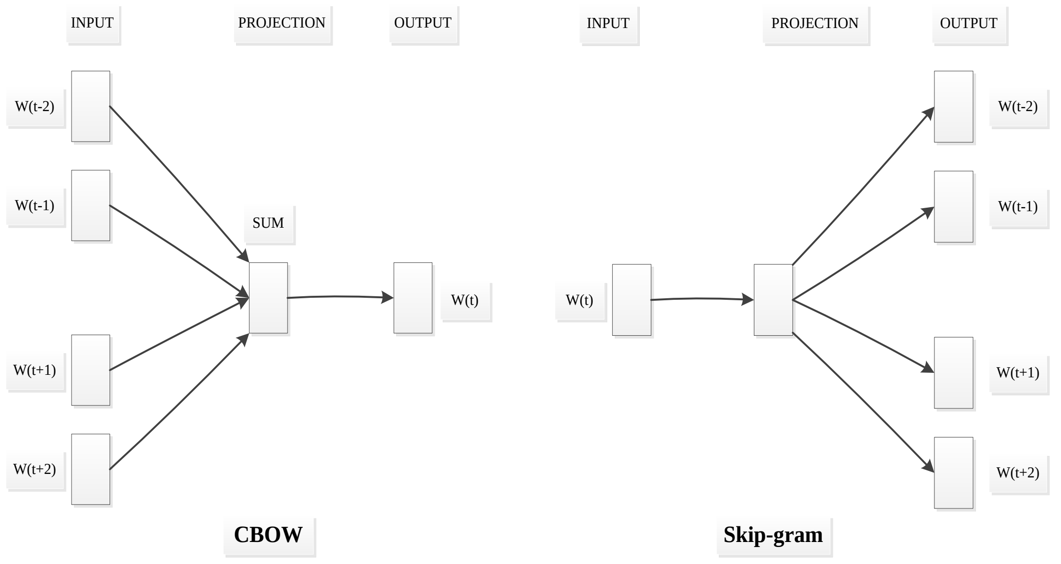

Figure 1. A simple CBOW and Skip-gram Model

Algorithmically, both of these models are same, the difference in the way they trained to predict the target words. Figure 1 is an illustration of CBOW and Skip-gram models. The training objective of the CBOW model is to combine the representations of surrounding (context) words to predict the word in the middle (target word). Whereas, skip-gram model trained to predict the source context words based on the target word. These models were implemented in Word2vec, a google project led by Tomas Mikolov and claimed to be a computationally-efficient predictive model for learning word embeddings from raw text.

Skip-gram successfully works on word analogies, but poorly utilizes the global co-occurrence statistics as it trained on separate local context windows. GloVe [21], a specific weighted least squares model that trained on global word-word co-occurrence matrix, thus, can be seen as an efficient model that uses global statistics. Eventually, GloVe can be advantageous over word2vec in that it is easier to parallelize the implementation. Consequently, this type of models can be trained over more data which is an advantage over word2vec. The disadvantage of GloVe over word2vec is that it is context counting model, whereas its counterpart is context predicting model.

In [22], authors extend skip-gram model [7] and proposed topical word embeddings (TWE) and tried to incorporate word relations by associating topical information with words. They follow the notion that each word has different embeddings under different topics. Though TWE is successful in capturing topical information and complex word correlations, it suffers from the loss of word-order and contextual information.

3.2.3. Multiple Embeddings per Word: The single vector representation corresponding to each word that most of the word embedding methods follow, suffers in handling homonymy and polysemy. As these models use only local context and sometimes local context are not sufficient to capture the exact semantics of word as words are often polysemous. Reference [23] follows the approach of multi-prototype vector space model and proposed a Global Context-Aware Neural Language Model capable of learning word embedding that better capture the semantics by incorporating both local and global document context. Authors handled homonymy and polysemy by learning multiple embedding per word. Thus the score for next word computed using local as well as global context. This multi-prototype word embeddings model mitigates the problem of homonymy and polysemy by generating multiple embeddings corresponding to a single word. However, the model ignores the complex correlation that most of the words have, as it generates multi-prototype vectors for each word in isolation. In reality, words may have semantic relations, causing their semantic cluster to overlap. Thus, there is not a clear semantic boundary between words.

3.2.4. Embeddings of Longer Word Sequences: Researchers come up with embeddings of longer word sequences to incorporate word-sequence information. Consequently, phrase representation [24], sentence embeddings [25] and paragraph vectors [26] were developed that capture the semantics of word (or distance between words). Paragraph vectors, an unsupervised learning method, successfully learn fixed-length feature representations from variable-length pieces of texts. Paragraph vector is an extension of work by Mikolov *et al.*, [7] [8] in [26]. Though the model gives better performance on sentiment analysis, it lacks in capturing fine-grained sentence structure [25]. Sentence embedding model (Skip-Thoughts) shows great performance on BookCorpus. Authors used an encoder-decoder pair for semantics. Encoder encodes the current sentence $s(t)$, and based on $s(t)$ two separate decoders decode previous ($s(t\text{-}1)$) and next ($s(t+1)$) sentences. RNNs with Gated Recurrent Unit (GRU) are used for encoder-decoders. Skip-Thought Vectors need extensive training on a large corpus of contiguous text. Recently autoencoder is (Context-sensitive Autoencoder) [27] used in representation learning for computing text pair similarity. In Context-sensitive Autoencoder contextual information is fed into deep autoencoders as low dimensional vectors.

Word embeddings suffer from ignoring word-order, contextual and morphological information. Tensors [10], alternatives to vectors, are also gaining attention in semantic representation, for information retrieval [28], document analysis, text categorization.

Convolutional Neural Networks (CNN) is another machine learning approach that is highly used in image processing and text mining.

3.2.5. RNN Based Models: Feedforward Neural Network (FNN) used by Bengio *et al.*, [29] in their neural language model, trained to predict the next word in the given sequence of words as well as learn vector representations for each word in the dictionary. However, it has limitation in its structure of not having memory to remember previous words.

Hence, the number of context words taken to predict the next word has been limited by a fixed number of input neurons, thereby missing the necessary contextual information in determining the target word. Researchers introduced some sort of memory in FNN architecture by incorporating extra neurons (context neurons) connected to hidden layer, which further increased to theoretically infinite size by introducing Recurrent Neural Network (RNN). Consequently, RNN can handle arbitrary context length and can efficiently represent more complex patterns than the shallow neural networks.

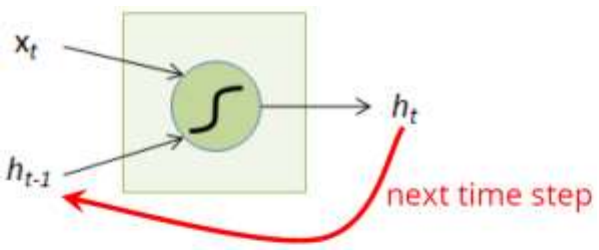

Figure 2. Single RNN Unit

Figure 2 is a single RNN unit. Unfolding this unit one can get complete architecture of RNN model which consists of input, hidden and output layers. The recurrent matrix in the RNN that connects hidden layer to itself, using time-delayed connections, contributes to the memory component in RNN [7]. The value of hidden layer gets updated based on the current input and the state of the hidden layer in the previous time step.

$$h_t = f(W_h \mathrm{h}_{t-1} + W_x x_t) \tag{3}$$

Where, x_t is input at time t and h_{t-1} is the hidden state at time t-1 and f is a transformation function. Input and output of the network are series of vectors. Learning in RNNs is defined by optimizing objective functions. Table 2 is common choices of activation functions. Out of these, most commonly used activation function in feedforward neural nets is *tanh*. Figure 3 demonstrates the use of activation function to get output activations

Table 2. Activation Functions

Activations	**Formula**
Step Function	$A_1 = 1$ $A_0 = 0$
Linear Combination	$y = \varsigma + b$
Sigmoid Function	$\sigma(t) = \frac{1}{1+e^{-\beta t}}$
Hyperbolic Tangent Function	$\phi(t) = \tanh(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}}$
ReLU	$l_j(t) = \max(0, t)$
Softmax Function	$\sigma(t) = \frac{e^{\varsigma_i}}{\sum_{j\varepsilon L} e^{\varsigma_i}}$

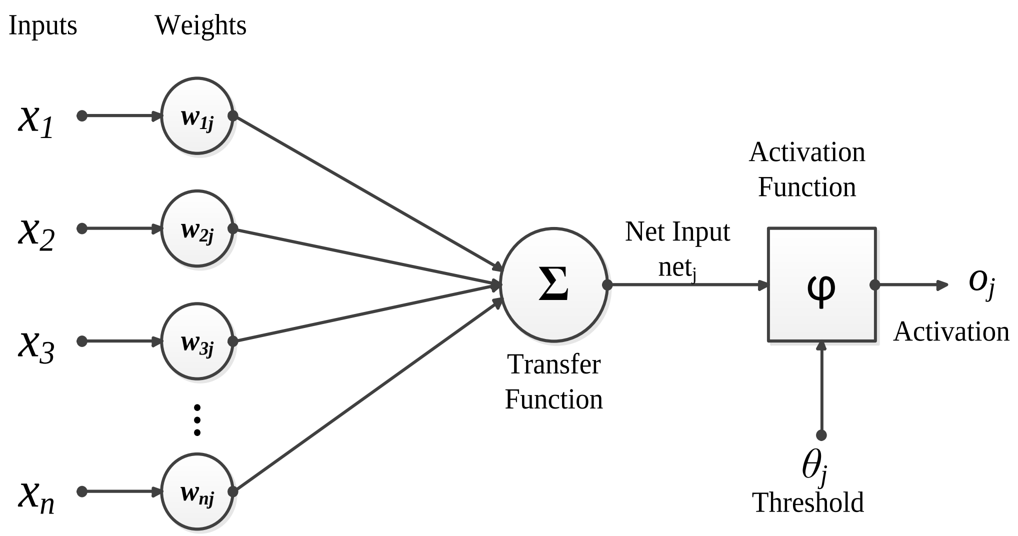

Figure 3. Use of Activation Function to get Output Activations

Mikolov [30] added recurrence to hidden layer to improve neural net language model and reported perplexity reductions against Good-Turing trigram baseline over 50% and against modified Kneser-Ney smoothed 5-gram over 40%. Another similar kind of work by [31] improves perplexity and BLEU scores on statistical machine translation. A generalization of recurrent neural networks presented as recursive auto-encoders (RAE) by [32], been useful in full sentence paraphrase detection beating the state-of-the-art techniques in paraphrase detection almost doubling the F1 score on MSRP paraphrase corpus. Unsupervised RAE can be used to learn feature vectors for phrases from parse trees. The state of the art embedding model is word2Vec, which uses full neural network and trained on simple single-layer architecture based on the inner product between two word vectors.

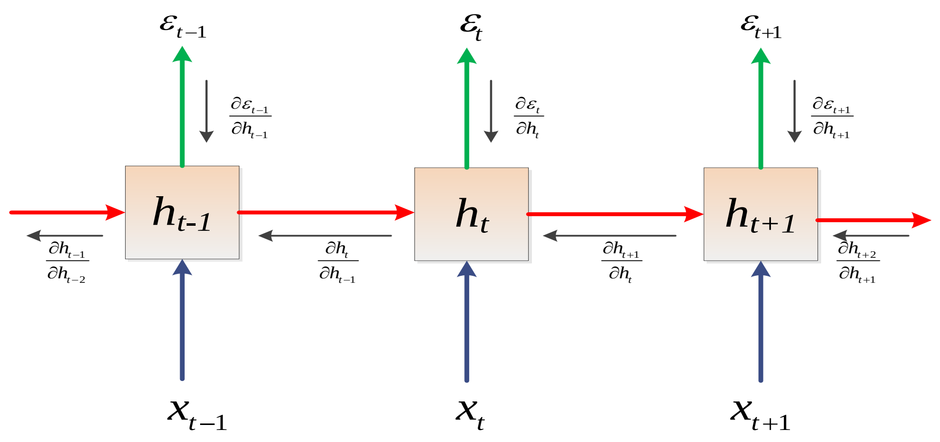
ε_{t-1}
ε_t
ε_{t+1}
$\frac{\partial \varepsilon_{t-1}}{\partial h_{t-1}}$
$\frac{\partial \varepsilon_t}{\partial h_t}$
$\frac{\partial \varepsilon_{t+1}}{\partial h_{t+1}}$
h_{t-1}
h_t
h_{t+1}
$\frac{\partial h_{t-1}}{\partial h_{t-2}}$
$\frac{\partial h_t}{\partial h_{t-1}}$
$\frac{\partial h_{t+1}}{\partial h_t}$
$\frac{\partial h_{t+2}}{\partial h_{t+1}}$
x_{t-1}
x_t
x_{t+1}

Figure 4. Gradients through Backpropagation

Recently, Mikolov *et al.*, [33] presented a gradient descent optimization based RNN architecture to learn word sequences in natural language. By slight structural modification in RNN model, they get improved performance equivalent to LSTM on language modeling tasks.

The extent, to which RNNs can be exploited, is limited by the effectiveness of training algorithms. Gradient based methods (Figure 4) suffer to exploding and vanishing gradient problem; as back-propagated error quickly either blows up or vanish. Hence standard RNNs are not suitable for capturing long term-dependencies.

3.2.6. CNN Based Models: RNN based embeddings are helpful in making a prediction within local context windows (shallow window-based method). However, these embeddings are task and domain specific. A unified model of multitask learning [34] was proposed to use convolutional neural network and learn a vector for multiple tasks. The network was trained jointly on all the tasks using weight-sharing. Researchers in the text mining community have exercised on various size of context window. To achieve the advantage of full context window [35] used the full context of a word for learning the word representations. SENNA system [35] uses convolutional neural networks to extract more generic representations that can be shared across the tasks of language modeling, named entity recognition, part-of-speech tagging, semantic role labeling, syntactic parsing and chunking.

Convolutional Neural Network (CNN) have wide applications in computer vision. CNNs are particularly a variant of feedforward neural network. Using CNN is advantageous over traditional neural networks in terms of parameter sharing. Unlike traditional neural networks, CNN can reuse filter parameters in convolution operations to calculate the output.

Given input sentence matrix $W \in R_{n \times d}$ and filter $f \in R_{m \times d}$ with sliding window size m, convolutional output of W and f is n-dimensional vector o:

$$o_i = \sum_{k=0}^{m-1} \sum_{j=0}^{d-1} f_{m-k-1,j} W_{i-k,j} \tag{4}$$

Advantage of using CNN over RNN is that CNN can extract hierarchical features over large contexts by stacking it to represent large context windows. Besides, CNN can be parallelized as input does not depend on word sequences.

3.2.7. LSTM Based Models

Unrolling RNNs works better in local context and suitable for capturing relationships in sequential data *i. e.*, word order information in case of text data. However, it failed in capturing long-term dependencies due to vanishing and exploding gradient problem. Long Short-Term Memory (LSTM), introduced in [36] resolves vanishing gradient problem, by introducing special units called memory blocks in the recurrent hidden layer. Memory blacks include memory cells and special multiplicative units called gates. Memory cells are capable of storing the temporal state information of the network. Additional multiplicative units control the flow of information in the network by regulating what to add and what to remove in the cell state. Memory cells maintain more constant error, by preserving the error that can be back propagated through different time steps and layers. Thus, allowing recurrent nets to continue learning over time steps and layers.

Unlike RNN that impose bias towards recent observations, LSTM tries to solve the vanishing gradient problem by keeping constant error flowing back through time. Every cell in a simple LSTM consists of three nonlinear gates namely input (i), forget (f) and output (o). The input gate controls the flow of input activations into the memory cell. The forget gate selectively discards the long term dependencies information from going through into the network. The output gate selects the information need to pass to the next cell/layer. LSTM cell can be defined with a following set of equations:

$$i_t = \sigma(x_t W^i + h_{t-1} U^i + b_i) \tag{5}$$

$$f_t = \sigma(x_t W^f + h_{t-1} U^f + b_f) \tag{6}$$

$$o_t = \sigma(x_t W^o + h_{t-1} U^o + b_o) \tag{7}$$

$$g_t = \tanh(x_t W^g + h_{t-1} U^g + b_g) \tag{8}$$

$$c_t = c_{t-1} \circ f_t + g_t \circ i_t \tag{9}$$

$$h_t = \tanh(c_t) \circ o_t \tag{10}$$

$$y_t = \phi(h_t U^y + b_y) \tag{11}$$

Where, i_t, f_t and o_t are input, out and forget vectors respectively. c_t is cell memory vector. σ is sigmoid function. W, U and b: parameter matrices (W is for weight, U is for update, and b is for bias)

LSTM have several variations. Amongst other, [37] proposed a model with "peephole connections" which allows the gate layers to look at the cell state. Eqns 5-7 can be modified to get Eqns 12-14.

$$i_t = \sigma(x_t W^i + c_{t-1} V^i + h_{t-1} U^i + b_i) \tag{12}$$

$$f_t = \sigma(x_t W^f + c_{t-1} V^f + h_{t-1} U^f + b_f) \tag{13}$$

$$o_t = \sigma(x_t W^o + c_{t-1} V^o + h_{t-1} U^o + b_o) \tag{14}$$

Other researchers rule out the concept of separately deciding the activations at forget and input gates and presented a model which coupled forget and input gates. The forget gate updated only when input gate is need to be updated and vice versa. Thus Eqn 9 can be redefined as:

$$c_t = c_{t-1} \circ f_t + g_t \circ (1 - f_t) \tag{15}$$

Gated Recurrent Unit (GRU) by [38] combines the forget and input gates into a single "update gate." GRU adaptively capture dependencies on different time scales. It merges the cell state and hidden state to get Eqns 16-19. This results in a simpler model than basic LSTM model.

$$z_t = \sigma(W^z x_t + U^z h_{t-1}) \tag{16}$$

$$r_t = \sigma(W^r x_t + U^r h_{t-1}) \tag{17}$$

$$j_t = \tanh(x_t W^j + (r_t \circ h_{t-1}) W^j) \tag{18}$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ j_t \tag{19}$$

Depth Gated RNNs proposed in [39] is another variation the LSTM have. To introduce a linear dependence between lower and upper recurrent units, memory cells in adjacent layers are connected by a new gate called "depth-gate." Authors claimed that this new model improve the performance on machine translation and language modeling.

$$i_t^{L+1} = \sigma(W_{xi}^{L+1} x_t + W_{hi}^{L+1} h_{t-1}^{L+1} + W_{ci}^{L+1} c_{t-1}^{L+1}) \tag{20}$$

$$f_t^{L+1} = \sigma(W_{xf}^{L+1} x_t + W_{hf}^{L+1} h_{t-1}^{L+1} + W_{cf}^{L+1} c_{t-1}) \tag{21}$$

$$d_l^{L+1} = \sigma(b_d^{L+1} + W_{xd}^{L+1} x_t^{L+1} + W_{cd}^{L+1} \circ c_{t-1}^{L+1} + W_{ld}^{L+1} \circ c_t^L) \tag{22}$$

$$c_t^{L+1} = d_t^{L+1} c_t^L + f_t^{L+1} \circ c_{t-1} + i_t^{L+1} \circ \tanh(W_{xc} x_t + W_{hc} h_{t-1}^{L+1}) \tag{23}$$

$$o_t^{L+1} = \sigma(W_{xo}^{L+1} x_t + W_{ho}^{L+1} h_{t-1} + W_{co}^{L+1} \mathrm{c}_t^{L+1}) \tag{24}$$

$$h_t^{L+1} = o_t^{L+1} \circ \tanh(c_t^{L+1}) \tag{25}$$

where i_t^{L+1}, f_t^{L+1}, o_t^{L+1} and d_l^{L+1} are the input gate, forget gate, output gate and the depth gate.

In Figure 5 the depth-gate d_l^{L+1} connects cell c_t^{L+1} in layer L+1 to cell c_l^L in layer L and controls the flow of information from lower to upper layer.

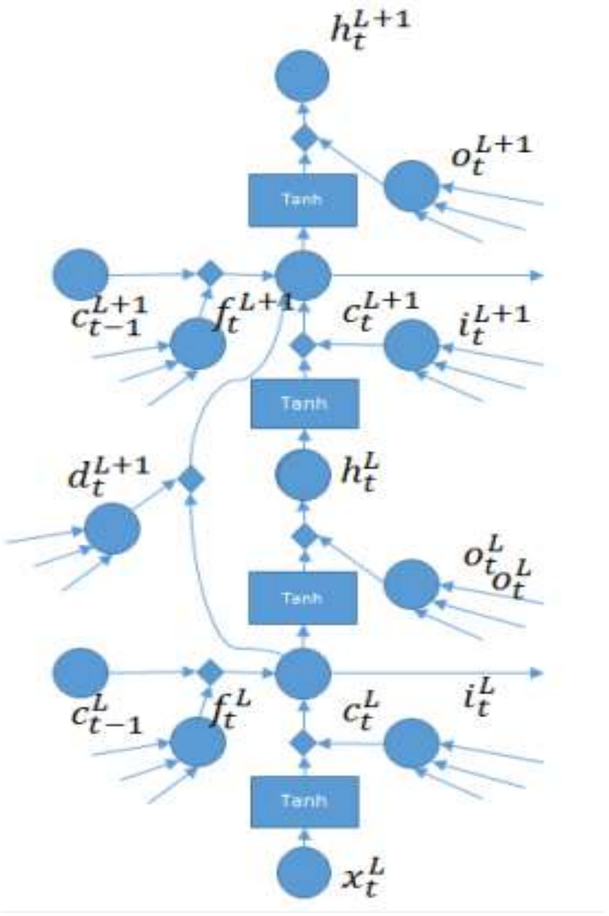

Figure 5. The Depth-Gated LSTM by Yao, et al. (2015)

3.3. Tensor Space Model

Embeddings are task and domain specific. Therefore, learning universal embeddings whilst keeping word-order and contextual information intact, is still a challenging task. Tensors [10], alternatives to vectors, are also gaining attention in semantic representation, for information retrieval [28], document analysis, text categorization. Dimensionality reduction in VSM is limited by the number of samples, whereas, TSM has no such limit. HOSVD under TSM can reduce the data to any dimension [28].

4. Tools for Semantic Representation Learning

The success and popularity of embeddings is also due to open-source software packages word2vec (including pre-trained word-embeddings), GloVe and Deeplearning4j. In this section, we present some of tools/software (Table 3) packages that are publically available as open source or as proprietary software for semantic representation learning. This may help a beginner choosing proper tool for her/his research.

Table 3. Tools/Software Packages for Semantic Representations

Tools	Open source	Developed by	Written in	Support for	license	Applications
Deeplearning 4j	Yes	DL4j community led by Adam Gibson; Skymind	Java	Java, Scala, Clojure, Python (Keras)	Apache 2.0	Anomaly Detection & Cyber Security
TensorFlow	Yes	Google Brain team	C++, Python	Python (Keras), C/C++, Java, Go, R	Apache 2.0	Machine Learning; to detect and decipher patterns and correlations, Used in Google products
Torch	Yes	Ronan Collobert, Koray Kavukcuoglu, Clement Farabet	C, Lua	Lua, LuaJIT, C, utility library for C++/OpenCL	BSD 3-clause "New" or "Revised"	Deep Machine Learning, used by the Facebook AI Research Group, IBM, Yandex and the Idiap Research Institute
Theano	Yes	machine learning group at the Université de Montréal	Python	Python	BSD license	Define, optimize, and evaluate mathematical expressions, especially with multi-dimensional arrays
Caffe	Yes	Berkeley Vision and Learning Center Originally Yangqing Jia	C++	Python, MATLAB	BSD license	For deep learning
ND4J	Yes	DL4j community, led by Adam Gibson	Java	Java, Scala, Clojure, Python (Keras)	Apache 2.0	for performing linear algebra and matrix manipulation in a production environment; used by Nasa JPL for tasks such as climatic modeling;
Gensim	Yes	RaRe Technologies originally Radim Řehůřek	Python	NumPy, SciPy, Cython	LGPL	information Retrieval, Vector Space Modeling and Topic Modeling
RapidMiner	No	AI Unit of the Technical University of Dortmund	Java	GUI, R and Python scripts	Proprietary (Limited free edition under a GPL license)	data preparation, machine learning, deep learning, text mining, and predictive analytics
Neural Designer	No	Artelnics	C++	Graphical User Interface	Proprietary	Data mining, machine learning, predictive analytics
Wolfram Mathematica	No	Wolfram Research	C/C++, Java, Wolfram Language	Command line, Java, C++	Proprietary	Text Mining (SA), Computer algebra, numerical computations, information visualization, statistics

5. Conclusions

This paper is first (to our knowledge), presenting systematic survey of methods for semantic representations. In the survey, we found that no approach is perfectly suitable for all text mining and NLP applications. Some is suitable for one set of applications others are for some other set. Therefore, the optimal approach for semantic representations depends on intended application. Moreover, it also depends on whether representations would be applied in supervised or unsupervised environment. In unsupervised environment, fast, shallow, log linear BoW models can still achieve the best performance. Models based on deep learning should be preferred for representations in supervised environment.

It has been observed informally that longer embeddings (*e. g.*, sentence and phrase rather than word and character) are more appropriate for capturing internal sentence structure, word-order and context which are suitable for applications that require deep language processing like machine translation and speech recognition. Word2Vec vectors works well on majority of supervised settings. However, for paraphrase identification, SDAEs is a better approach.

Conclusively, these findings taken together provide valuable pointers for researchers looking to work in the field of semantic representations. Also, the survey shows that one need to develop a model for learning universal embeddings in unsupervised/semi-supervised settings that incorporate contextual information as well as word-order information, with language independent features and which would be feasible for large dataset.

References

[1] K. R. Logical, J. E. Sowa, P. Grove and C. a B. Cole, “Knowledge representation: logical, philosophical, and computational foundations”, Anticancer Res., vol. 32, no. 5, pp. 2217, 2012.

[2] A. W. and C. S. Y. G. Salton, “Vector Space Model for Automatic IndexingNo Title”, ACM, **(1975)**, pp. 613-620.

[3] B. Croft and J. Lafferty, Eds., Language Modeling for information retrieval, **(2013)**.

[4] M. Uschold, M. King, S. Moralee and Y. Zorgios, “The Enterprise Ontology”, Knowl. Eng. Rev., vol. 13, no. 1, **(1998)**, pp. 31-89.

[5] J. Wu, Z. Xuan and D. Pan, “Enhancing text representation for classification tasks with semantic graph structures”, Int. J. Innov. Comput. Inf. Control, vol. 7, no. 5 B, **(2011)**, pp. 2689-2698.

[6] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer and R. Harshman, “Indexing by Latent Semantic Analysis”, J. Am. Soc. Inf. Sci., vol. 41, no. 6, **(1990)**, pp. 391-407.

[7] T. Mikolov, G. Corrado, K. Chen and J. Dean, “Efficient Estimation of Word Representations in Vector Space”, Proc. Int. Conf. Learn. Represent. (ICLR 2013), **(2013)**, pp. 1-12.

[8] T. Mikolov, K. Chen, G. Corrado and J. Dean, “Distributed Representations of Words and Phrases and their Compositionality”, Nips, **(2013)**, pp. 1-9.

[9] J. Xu, Y. Tao and H. Lin, “Semantic word cloud generation based on word embeddings”, IEEE Pacific Vis. Symp., vol. 2016, **(2016)** May, pp. 239-243.

[10] A. Cichocki, R. Zdunek, A. H. Phan and S. I. Amari, "Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation", **(2009)**.

[11] D. Jurafsky and J. H. Martin, "Speech and Language Processing", **(2015)**.

[12] R. A. García-Hernández and Y. Ledeneva, “Word sequence models for single text summarization”, Proc. 2nd Int. Conf. Adv. Comput. Interact. ACHI 2009, **(2009)**, pp. 44-48.

[13] K. Radinsky, E. Agichtein, E. Gabrilovich and S. Markovitch, “A word at a time: computing word relatedness using temporal semantic analysis”, Proc. 20th Int. World Wide Web Conf. WWW’11, **(2011)**, pp. 337-346.

[14] E. Gabrilovich and S. Markovitch, “Computing semantic relatedness using wikipedia-based explicit semantic analysis”, IJCAI Int. Jt. Conf. Artif. Intell., **(2007)**, pp. 1606-1611.

[15] T. Hofmann, “Unsupervised learning by probabilistic Latent Semantic Analysis”, Mach. Learn., vol. 42, no. 1-2, **(2001)**, pp. 177-196.

[16] W. Buntine, “Estimating Likelihoods for Topic Models”, Asian Conference on Machine Learning, **(2009)**, pp. 51-64.

[17] T. Singh, M. Kumari, T. L. Pal and A. Chauhan, “Current Trends in Text Mining for Social Media”, Int. J. Grid Distrib. Comput., vol. 10, no. 6, **(2017)**, pp. 11-28.

[18] T. Mikolov, I. Sutskever, A. Deoras, H. S. Le, S. Kombrink and J. Cernocky, “Subword language

modeling with neural networks", **(2012)** March.

[19] W. Ling, "Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation", Proc. 2015 Conf. Empir. Methods Nat. Lang. Process., **(2015)** September, pp. 1520-1530.

[20] Y. Kim, Y. Jernite, D. Sontag and A. M. Rush, "Character-Aware Neural Language Models", Aaai, **(2016)**, pp. 2741-2749.

[21] J. Pennington, R. Socher and C. D. Manning, "GloVe: Global Vectors for Word Representation", Proc. 2014 Conf. Empir. Methods Nat. Lang. Process., **(2014)**, pp. 1532-1543.

[22] Y. Liu, Z. Liu, T.-S. Chua and M. Sun, "Topical Word Embeddings", Proc. 29th AAAI Conf. Artif. Intell., vol. 2, no. C, **(2015)**, pp. 2418-2424.

[23] E. H. Huang, R. Socher, C. D. Manning and A. Y. Ng, "ImprovingWord Representations via Global Context and MultipleWord Prototypes", Proc. 50th Annu. Meet. Assoc. Comput. Linguist., **(2012)** July, pp. 873-882.

[24] R. Socher and C. Lin, "Parsing natural scenes and natural language with recursive neural networks", Proc. …, **(2011)**, pp. 129-136.

[25] H. Palangi, "Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval", IEEE/ACM Trans. Audio, Speech, Lang. Process., vol. 24, no. 4, **(2016)**, pp. 694-707.

[26] Q. Le and T. Mikolov, "Distributed Representations of Sentences and Documents", Int. Conf. Mach. Learn. - ICML 2014, vol. 32, **(2014)**, pp. 1188-1196.

[27] H. Amiri, P. Resnik, J. Boyd-graber and H. Daum, "Learning Text Pair Similarity with Context-sensitive Autoencoders", **(2015)**.

[28] L. Ning, "Text representation: From vector to tensor", Proc. - IEEE Int. Conf. Data Mining, ICDM, **(2005)**, pp. 725-728.

[29] Y. Bengio, R. Ducharme, P. Vincent and C. Janvin, "A Neural Probabilistic Language Model", J. Mach. Learn. Res., vol. 3, **(2003)**, pp. 1137-1155.

[30] J. Mikolov, T. Deoras, A. Kombrink, S. Burget and L. Černocký, "Empirical Evaluation and Combination of Advanced Language Modeling Techniques", Interspeech, **(2011)** August, pp. 605-608.

[31] H. Schwenk, A. Rousseau and M. Attik, "Large, Pruned or Continuous Space Language Models on a GPU for Statistical Machine Translation", NAACL-HLT Work. Futur. Lang. Model. HLT, **(2012)**, pp. 11-19.

[32] R. Socher, E. Huang and J. Pennington, "Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection", Adv. Neural Inf. Process. Syst., **(2011)**, pp. 801-809.

[33] T. Mikolov, A. Joulin, S. Chopra and M. Mathieu, "Learning Longer Memory in Recurrent Neural Networks", Iclr, **(2015)**, pp. 1-9.

[34] R. Collobert and J. Weston, "A unified architecture for natural language processing", Proc. 25th Int. Conf. Mach. Learn. - ICML '08, vol. 20, no. 1, **(2008)**, pp. 160-167.

[35] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa, "Natural Language Processing (almost) from Scratch", **(2011)**.

[36] S. Hochreiter and J. J. Schmidhuber, "Long short-term memory", Neural Comput., vol. 9, no. 8, **(1997)**, pp. 1-32.

[37] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count", Proc. IEEE-INNS-ENNS Int. Jt. Conf. Neural Networks. IJCNN 2000. Neural Comput. New Challenges Perspect. New Millenn., vol. 1, **(2000)**, pp. 189-194.

[38] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling", arXiv, **(2014)**, pp. 1-9.

[39] K. Yao, T. Cohn, K. Vylomova, K. Duh and C. Dyer, "Depth-Gated Recurrent Neural Networks", Jelinek Summer Workshop on Speech and Language Technology, **(2015)**, pp. 1-5.