IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 12 March 2023, accepted 9 April 2023, date of publication 11 April 2023, date of current version 14 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3266377

SURVEY

A Survey of Text Representation and Embedding Techniques in NLP

RAJVARDHAN PATIL[1], SORIO BOIT[1], VENKAT GUDIVADA[2], AND JAGADEESH NANDIGAM[1]
[1]School of Computing, Grand Valley State University, Allendale, MI 49401, USA
[2]Department of Computer Science, East Carolina University, Greenville, NC 27858, USA

Corresponding author: Rajvardhan Patil (patilr@gvsu.edu)

ABSTRACT Natural Language Processing (NLP) is a research field where a language in consideration is processed to understand its syntactic, semantic, and sentimental aspects. The advancement in the NLP area has helped solve problems in the domains such as Neural Machine Translation, Name Entity Recognition, Sentiment Analysis, and Chatbots, to name a few. The topic of NLP broadly consists of two main parts: the representation of the input text (raw data) into numerical format (vectors or matrix) and the design of models for processing the numerical data. This paper focuses on the former part and surveys how the NLP field has evolved from rule-based, statistical to more context-sensitive learned representations. For each embedding type, we list their representation, issues they addressed, limitations, and applications. This survey covers the history of text representations from the 1970s and onwards, from regular expressions to the latest vector representations used to encode the raw text data. It demonstrates how the NLP field progressed from where it could comprehend just bits and pieces to all the significant aspects of the text over time.

INDEX TERMS Natural language processing, embeddings, text representation, word vectors, survey, word embeddings, literature review, NLP, language models.

I. INTRODUCTION

The topic of NLP broadly consists of two main steps: first is the representation of the input text (raw data) into numerical format (vectors or matrix), and second is the design of models for processing the numerical data to achieve a desired goal or task. This paper focuses on the first part and shows how because of the change in the text representation, the NLP field progressed from just being able to comprehend bits and pieces to all the aspects of the text. As shown in Figure 1, we broadly classify the embedding learning techniques into rule-based, statistical, and neural-network-based approaches.

In hand-based techniques, the rules and features are derived by experts, such as trees, graphs, grammar rules, etc. Statistical and mathematical formulas are used to derive the feature. In neural networks, the neural model learns the features automatically, which are categorized into context-sensitive, context-insensitive, and pre-trained embeddings.

The associate editor coordinating the review of this manuscript and approving it for publication was Sunil Karamchandani.

The NLP field began with the exact matching technique where Context Free Grammar (CFG) was used for analysis. The search engines were mainly based on the complex nested if-then rules represented by CFG. Further advancements led to approximate matching, where errors up to a certain threshold were ignored. However, because of the ambiguous nature of natural languages, using such techniques and designing complex rules could have been more convenient, time-consuming, and error-prone. More statistical approaches then followed these pattern-matching approaches.

As the research evolved, statistical approaches were sought, where the focus was given to the frequency of the words. Several techniques, such as One Hot Encoding (OHE), Bag of Words (BoW), Term-Frequency (TF), Inverse-Document-Frequency (IDF), etc., fall under this category. Compared to grammar-based, these techniques were easy to implement and improved the models' accuracy. However, such representations suffered from the curse of dimensionality, and also, because of computational power limitations, their performance suffered on large-scale datasets. As a

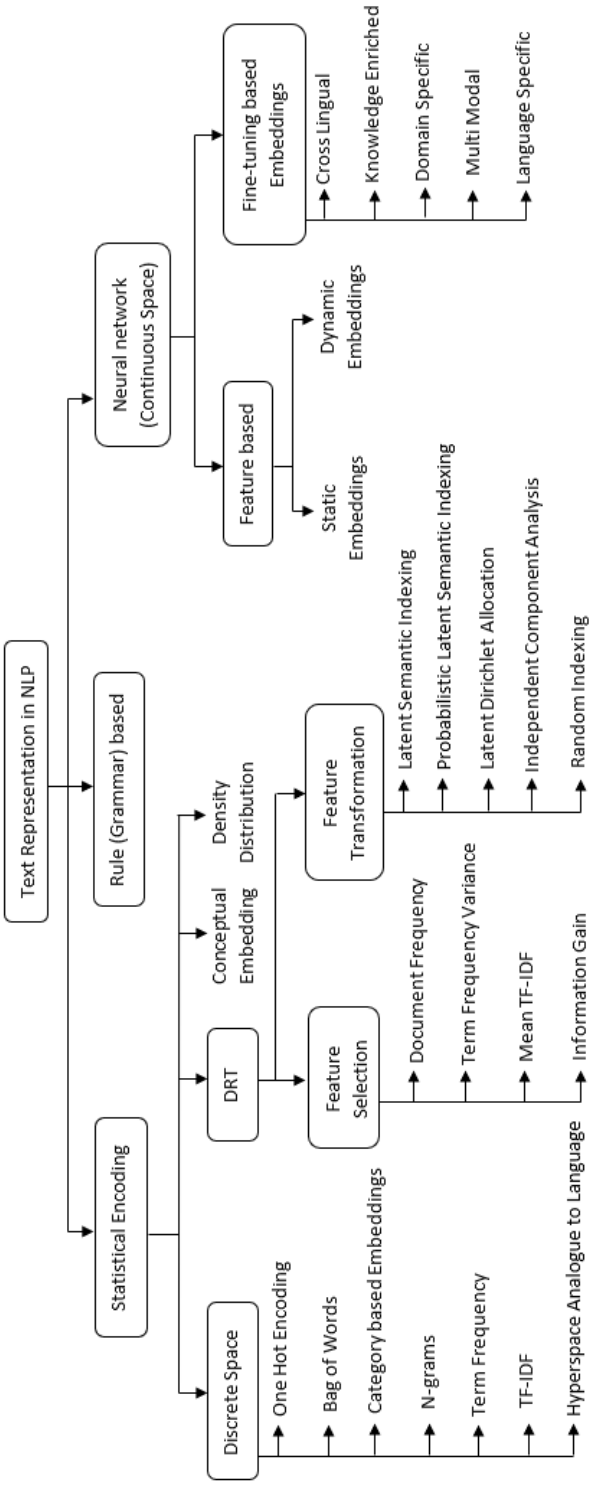


FIGURE 1. Embedding taxonomy.

result, they needed to be more scalable, and their usage was primarily limited to small-scale datasets.

Dimensionality Reduction Technique (DRT) was used to overcome the curse of the dimensionality problem. In these reduction techniques, the key idea is to find a new space or coordinate system in which input data can be expressed with fewer features, less information loss, and minimal error. Usually, two techniques are employed to reduce the dimensions. The first approach is feature selection, and the other is feature transformation. In feature selection, top-k terms or features are selected based on certain criteria or threshold values, and the rest are simply discarded. In feature transformation, linear or non-linear transformation is applied on the dimensions of the original dataset and mapped to a lower space with fewer dimensions. In both approaches, care is taken not to lose any information and that the content in the original data is preserved or captured. Such discrete representation does help in classification and categorization applications. However, it doesn't perform well for tasks such as information retrieval, chatbots, machine translation, and language generation, which require a deeper understanding of concepts such as polysemy or identifying analogies, synonyms, antonyms, etc. These limitations were overcome by advanced representations, called word vectors, which were derived using neural networks.

The embeddings derived from neural networks are mapped to continuous vector space, where each word vector is represented by an array of real numbers. In embeddings derived from such continuous vector space, the focus is more on computing the semantics of individual words by looking at the context in which they appear. This technique considers the effect that neighboring words can have on the given word and how these relationships affect the meaning of a word. These new word vectors are context-sensitive, can identify synonyms and antonyms, and construct analogies and categories of words, which was impossible in earlier approaches. Word vectors capture words' meanings (literal and implied) and represent them using dense floating-point values. They represent the semantics as well as the syntactic aspects of the word. Their length typically ranges between 100 to 500 dimensions.

Additionally, the earlier approaches required labeling of input data, whereas deducing the model processes word-vectors unlabeled data in a self-supervised manner. However, much more training data is required to improve the accuracy of word vectors. In these word vectors, the rows are the number of words in the corpus, and the columns are the number of categories, concepts, qualities, etc., where each feature represents some aspect of the word. For example, for a word such as 'City,' the important aspects could be friendliness, criminalness, socialness, economicalness, etc. Because of their preciseness, the word-embedding vectors if added or subtracted, result in a new word vector that means something semantically. If such a 'resultant' vector is plotted onto the continuous-vector space, it will point or be close nearby to some word's embedding. Such word embeddings can therefore be used to predict the words around them and capture the relationship between them.

As per [1], the field of NLP can be dated back to 1950s. Although there have been surveys done in the past, for example [41] that covered some of the Word Embeddings approaches, to the best of our knowledge this is the first

survey paper which exhaustively covers all the major text representation and word embeddings techniques that are used by the NLP models for processing the text. The outline of this survey paper is as follows: In Section II, the paper describes the NLP approach based on Context Free Grammar (CFG) and pattern matching. Section III enlists seven primary statistical methods that project the text onto the discrete space. In Section IV, two dimensionality reduction categories are introduced, followed by the description of techniques in each category that help reduce the dimensions and optimize the computational process. Section V explains conceptual embeddings, followed by density distribution techniques in Section VI. Finally, Section VII enumerates the popular techniques where embeddings are derived using neural network techniques, followed by the conclusion and future work in Section VIII.

II. REGULAR EXPRESSIONS

The initial development of the NLP field was mostly restricted to search engines and question-answering systems, which heavily relied on pattern matching [2]. Patterns could be a sequence of characters, words, or high-level abstract patterns such as parts of speech. Because of the computational resource limitations, the NLP field involves hard coding of regular expressions and complex logical rules, such as nested conditional if-else statements. The patterns would either be used to search the sequence in documents and then rank those documents in order of frequency or in the case of chatbots, they would generate a scripted response (answer) for the given sequence (question).

Programming languages tend to be context-free and can therefore be parsed or analyzed using Context-Free Grammar (CFG), where the context is unimportant or can be ignored. However, natural languages being context-sensitive; their meaning is subjective and is prone to change as per the context. Therefore, attempting to capture or parse the natural language using context-free grammar offers less flexibility in the rules. In addition, such algorithms often need to be completed, and more clear [3]. As a result, this technique of using regular expressions (represented by CFG) to capture natural language came across as more rigid, tightly controlled, and involved hard coding of the complex logical rules to match or extract information from a given text. They were based upon exact sequence or pattern matching and the rigid response rules embedded in the nested conditional if statements.

To overcome the above limitations, fuzzy expressions were introduced, performing approximate matches instead of exact matches. Such Fuzzy-CFG helped incorporate uncertainties and vagueness in NLP [4]. Additionally, as stated in [5], fuzzy grammar theory also helped capture the ambiguity and impreciseness that occur in natural language queries, as it can be parsed in different ways. Such systems overcame the brittleness of traditional regular expressions, where it found the closest grammar matches among a list of rules by ignoring a certain number of errors. Another approach to improve over the traditional CFG approach was to hard code the semantics of certain words to improve the accuracy of NLP results. For example, Keyword Search on Databases (KWS) is one of the NLP subfields. The task is to understand and represent the given natural language query in its equivalent SQL format. Authors from [6] and [7] demonstrated how the KWS systems give more accurate results by giving particular importance to logical operators (and, or, not); similarly, [8] focused on words such as ''average, count, total'' etc., to deal with advanced NLP queries. However, such attempts were limited in scope and domain, as the set of words to focus upon was domain specific.

Overall, in regular expressions, the word ordering and matching (exact or approximate) were considered, but the meaning, context, and frequency of words were ignored. These systems based on CFG can capture the breadth of the knowledge associated with the application and, therefore, can perform a simple keyword search, implement simple answering systems, and task-execution assistant bots; however, because of their rigidity, they are shallow, and so their capabilities of depth-wise understanding are limited, and therefore do not perform well for applications requiring a semantic understanding of language, such as text summarization, machine translation, etc. To overcome some of the limitations of regular expressions, the researchers explored and sought statistical approaches.

III. DISCRETE VECTOR SPACE

In statistical methods, words are represented using vectors of numbers, and the corpus is represented as a collection of such vectors, forming a matrix. Such statistical methods reduce documents of arbitrary length to fixed-length lists of numbers. These vector representations were helpful since they enabled researchers to use linear algebra operations to manipulate the vectors and compute distances and similarities. This helped address a much more comprehensive range of problems that otherwise required additional hand coding of regular expressions and nested conditional rules. Statistical approaches can broadly be categorized into three parts: (1) discrete vector space, (2) density vector space, and (3) continuous vector space.

In the discrete-vector space representations described below, the given corpus or text first goes through preprocessing, where the stop-words are eliminated, and stemming is performed to get the root or stem of each word. Stemming and lemmatization help normalize word endings so that words that differ only in their last few characters get collected under the same token. Also, the matrices in these techniques are named based on what the rows and columns represent. For instance, if the rows represent documents and columns represent words or terms, the matrix is termed a document-term matrix.

A. ONE HOT EMBEDDING (OHE)

In this technique, first, the vocabulary dictionary of the given corpus is constructed, which is a set of unique words. These sets of unique words are then sorted and indexed. The

TABLE 1. One hot encoding example.

	22	driving	he	old	started	was	when	years
he	0	0	1	0	0	0	0	0
started	0	0	0	0	1	0	0	0
driving	0	1	0	0	0	0	0	0
when	0	0	0	0	0	0	1	0
he	0	0	1	0	0	0	0	0
was	0	0	0	0	0	1	0	0
22	1	0	0	0	0	0	0	0
years	0	0	0	0	0	0	0	1
old	0	0	0	1	0	0	0	0

vocabulary is represented by V and its size by |V|. The rows in the resulting matrix are equal to the number of words in the given sentence or document, and the number of columns is equal to the vocab size, |V|. Here, the row index depicts the position or place of a word in the corpus.

In contrast, the column index represents the dictionary position or the index of that word in the vocabulary V. The resulting matrix is termed a ''word-term'' or ''word-word'' matrix. To build the matrix, the given text is traversed from left to right. The current-word position in the text acts as a row index in the matrix, and its index in the vocabulary acts as a column index in the matrix. The value at these row and column indexes is then marked as '1' to indicate the presence of the current word. This representation is called a one-hot encoding because, in a given row, only one column index has a non-zero value of '1'. Consider the following example:

Sentence = ''he started driving when he was 22 years old''

Here, vocab = ['22', 'driving', 'he', 'old', 'started', 'was', 'when', 'years']

The one-hot matrix for the above sentence is shown in Table 1, having dimensions (9,8).

If logical operations are performed on these one-hot vectors, they can be used to do a basic keyword search, which looks for the presence of one or more query words in a given document or corpus. Such representation also retains the ordering of words in the sentence, and it does not lose any information from the original text. Given the matrix, the original text can be reconstructed, which is impossible in other embeddings described below.

However, such encoding ignores the context and hence fails to capture the relationships and meanings of the words. The dot product between such OHE vectors does not convey any meaningful information, as a result, is always a zero. Also, the distance between such word vectors does not give any meaningful insights as they represent differences in alphabetic ordering. That is, vectors of words with similar meanings ('cold', 'freezing') get projected at more considerable distances. In contrast, vectors of words with opposite meanings ('freezing', 'hot') have a comparatively smaller distance.

Furthermore, such representations come with a cost of many dimensions and more storage space and are sparse, making the models costlier to train. As a result, using such a representation does not do any better than regex-pattern matching since the distance between vectors fails to measure the difference or similarity in meaning between the sequences.

TABLE 2. Bag of words example.

	Sam	started	driving	...	preferred	outdoor	activities
Sent1	1	1	1	...	0	0	0
Sent2	0	0	0	...	0	0	0
Sent3	0	0	0	...	1	1	1

B. BAG OF WORDS (BOW)

An early reference to ''bag-of-words'' in a linguistic context can be found in Zellig Harris's 1954 article on Distributional Structure [9]. As one-hot vectors were sparse, a more concise representation called bag-of-words was proposed. In this approach, the corpus is represented by a matrix, where each row represents a sentence, and each column represents a unique word from vocabulary. The number of rows is equal to the number of sentences in the corpus, and the number of columns is equal to the vocabulary size of the corpus. This matrix is termed a 'document-term' with a dimension equal to | the number of sentences| x |V|.

While processing each sentence in the corpus, the frequency or the absence/presence (indicated by 0 or 1) of each occurring word in the sentence is counted and updated in the respective row and column of the matrix. Here, the row index is the sentence's position in the corpus, and the column index is the current word's dictionary position. If the values are binary (0,1) then such BoW is termed binary BoW, whereas if the count or frequency of each word is recorded, it is termed non-binary BoW. Consider the following example:

corpus = ''Sam started driving when he was 22 years old.

He was passionate about swimming, and won several competitions.

He didn't like watching television, as he preferred outdoor activities.''

The bag-of-words representation for the above corpus is shown in Table 2, having dimensions(3,25).

Compared to OHE, the matrix's row size is significantly smaller and requires less time for model training. However, like OHE, this matrix is sparse as each sentence only has a few words from the corpus, resulting in '0' entry for most of the vocabulary words (columns). Here, as each sentence or document within a corpus is represented using a row, it enables finding similarity scores among sentences or between documents. A dot product, shown in 1, is typically used to compute document similarity scores.

$$A.B = \sum_{i=1}^{n} A_i B_i \tag{1}$$

where, 'A' and 'B' are documents and 'n' is the dimension of the vector space. Such dot product measures the overlap between two vectors and results in a scalar value or metric, giving a good estimate (approximation) of how similar they are in their words. To perform a keyword search on the corpus,

logical operations (OR, AND) are performed between the query and document vectors.

This representation is, however, based on exact matching. Therefore, document similarity is correct only if the exact words are used. Therefore, it will give an inaccurate similarity score for documents conveying the same information but using different words or synonyms. Furthermore, such representation comes at the cost of ignoring the word order originally preserved in OHE. Different sentences with different semantics can have the same BoW representation. The ordering of words becomes important as they convey logical relationships and dependencies between the words. Extracting such syntactical information between the words helps solve problems such as text generation (predicting the next word), text summarizing, etc. Additionally, the context and the relationship between words need to be addressed, which makes BoW fail to comprehend the semantic information of the words. Because of such shortcomings, its usage is limited to spam filters and sentiment analysis applications.

C. CATEGORY BASED EMBEDDING (CBE)

In this approach, the focus is on the types of categories that exist across the documents. Instead of dynamically identifying these categories, in this approach, they are pre-decided and hard-coded. Information about such categories is represented using columns. Therefore, the number of columns is equal to the number of possible categories in the corpus. The resulting matrix is termed a 'document-category' matrix. Such representation is helpful when a given document needs to be classified into one of these categories. Here, the matrix's rows equal the number of sentences or documents in the corpus.

In OHE and BoW approach, a separate column was dedicated to each word in the vocabulary V. However, the dimensionality in CBE is significantly reduced since the number of categories (represented by the columns) is way lower than the vocabulary size. For example, in the sentiment classification task, the documents can be grouped mainly into positive, neutral, and negative polarities. In this example, the number of categories and columns is just three. While iterating through each document, the polarity of each word is found by looking into user-defined lexicons, and the corresponding polarity or category count for the given document is incremented in the matrix. This matrix representation is dense as it has very few columns. Also, it is not sparse because most of the rows have non-zero values. As a result, the resulting matrix requires comparatively less storage space than its counterparts. Furthermore, because of fewer dimensions, the training of models using such representation is much faster. Consider the following example:

corpus = ''I am feeling fresh and happy because I had gone on vacation.

I did not enjoy reading his novel.

I hated the movie as the plot was missing.

The meal at this restaurant is delicious.''

TABLE 3. A category based embedding example.

Reviews	Positive	Negative	Overall
Sentence-1	2	0	Positive
Sentence-2	1	1	Neutral
Sentence-3	0	2	Negative
Sentence-4	1	0	Positive

The CBE matrix having dimensions (4, 3) for the above corpus is shown in Table 3.

To compose the lexicon, two approaches are normally used. In the heuristic approach, polarity scores are provided by humans. Reference [10] demonstrates popular hand-composed lexicons used for sentiment analysis. This predominantly includes six lexicons, namely: Vader [11], AFINN [12], Emotion Lexicon [13], Bing Liu's Lexicon [14], MPQA Lexicon [15], and General Inquirer [16]. However, compiling such a lexicon is labor intensive and might only cover some of the words from the corpus that represent or indicate some sentiment. Also, it becomes difficult to implement this approach if the language is foreign to the user, as the polarities of unfamiliar words are unknown.

The other approach is the non-sentiment-based approach. Given the input data with the labels, a frequency dictionary is generated that maps a given the word to its count of occurrences in each class. For example, the word 'happy' might occur too often in text with positive labels than negative ones. So the generated mapping would be: {happy : (15, 2)}, where 15 represents the number of times 'happy' appeared in positive documents, and 2 represents the number of times 'happy' appeared in negative sentiment documents.

The documents are then represented with these frequency-dictionary mapping, which are then used to train the model. In the heuristic unsupervised approach, the lexicons were referred to decide the polarity of the given word. In contrast, in the non-sentiment supervised approach, labels of the input data are used to deduce the sentiment of the word.

In this CBE representation, the vectors consist of frequency or count values, where the categorical (semantic) information is considered. However, the syntactic, contextual, and word-ordering information is ignored. Such representation is mostly used in applications that deal with short sentences. Since in short sentences, a change in word order does not change the meaning drastically. It also does better in applications requiring category-based counting, such as sentiment analysis of reviews, junk-email detections, finding troll messages, measuring sentiment or niceness in chat messages, etc.

D. N-GRAMS

In the above techniques, individual words represented the tokens that were then used for processing the documents. As the focus was on finding the presence or counting the frequency of individual tokens, the word ordering was ignored. N-gram was the first technique proposed in [17] that made an attempt towards imposing a window to capture the ordering among words. Here, ''n'' represents the size of window or context. So instead of focusing on individual words, n-gram

looks at multiword tokens and captures the ordering present in the window of the context. For instance, the 2-gram ''not playing'' captures correct ordering than if they were considered separately. Here, instead of paying attention to the occurrence of individual words, focus is on capturing group of words. Similarly for sentence-2 of the corpus presented in CBE subsection, as the focus was on individual words, the 'did not enjoy' words were treated separately. As a result, the sentence was incorrectly classified as neutral (instead of negative).

N-gram don'' necessarily capture the contextual information but is successful in capturing the word ordering among words. These words when occurred together might mean completely different than when they appear in isolation. For example, ''ice'' and ''cream'' versus ''ice cream'', or ''not'' and ''playing'' versus ''not playing''. If n-gram is not used, then ''ice'' and ''cream'' will be associated individually with the entire sentence or document, implying that the document is about ''ice'' and/or ''cream'', leading to wrong interpretation. That is, sometimes the semantic information is inherent in the ordering of words. Instead of individual words, extending the concept to incorporate n-grams (multiword tokens), enables the model to retain the meaning, which is implicit in the ordering of words or phrases. N-grams therefore become useful when certain group of words carry more weight or different meaning when appeared together. Consider the following example:

corpus = ''She can'' stand her.
She was fed up of the assignments.
He screwed up the exam.
He did not like trekking.
He could not bear the pain of separation.''

Here, bi-grams such as (can'' stand, screwed up), and tri-grams such as (did not like, could not bear) convey meaning when they are considered in group. Otherwise, individually, they might convey the incorrect (opposite or neutral) sentiment. N-grams are normally used in applications such as document classification, clustering, and sentiment analysis, where they capture important group of words (representing topic or concept) that occur frequently across the documents, and then classifying the documents based on the occurrence of n-grams in them. Authors from [18] and [19] demonstrated the use of n-grams in language modeling.

However, if n-grams are extremely rare in appearance, then they don'' carry much importance, as they cannot be used to help identify topics or themes that can connect or classify documents. On the other extreme, frequent n-grams might also not represent any important meaning, or might carry least information content. For example, n-grams such as ''of the'', ''so that'' etc., which occur too frequently don'' possess the ability to discriminate or classify the documents. So filtering mechanism is applied to eliminate n-grams which occur rarely or too frequently. Additionally, the n-gram that was used to train the model might not be in the same sequence in the test data. Furthermore, the possible combination of n-grams grow exponentially with value of 'n'. For example, to model the joint distribution of 5-word sequences in a vocabulary of 100,000 words, the possible number of combinations is $100000^5 - 1$. Due to these limitations, other techniques such as Term-Frequency, Document Frequency, and Inverse Document Frequency came to the forefront.

TABLE 4. Term frequency embedding example.

	'House' TF	...	Doc-Length	'House' Normalized-TF
Doc-A	50	...	150	50/150 = 0.33
Doc-B	1000	...	1,000,000	1000/1000000 = 0.001

E. TERM FREQUENCY EMBEDDING

Although the non-binary BoW approach did consider the frequency of words, it did not consider the importance of a word relative to the other words in the document or the corpus. To get more refined or precise results, finding out which words are more relevant to a particular document than others and their importance across the corpus as a whole becomes vital. Such a value of 'relevance' can be used to find more relevant documents based on the occurrence of query keywords in the individual documents and across the corpus.

In this approach, proposed by Salton [20], Term-Frequency (TF) counts the frequency of words in a document. However, simply counting the frequency can give misleading results. For instance, if the word 'house' appears in document A 50 times and in document B 1000 times, then just by considering the term frequency, it would be concluded that the word 'house' is more relevant to document B than document A. However, what if document A had only 150 words, whereas document B had 1 million words? In that case, the word 'house' seems more important to document A than document B.

So instead of solely relying on term frequency, Normalized Term Frequency (NTF) is often used, where the term frequency of the word is divided by the total number of words in that document. This normalization helps compute the importance of a word relative to other words in that document. Table 4 shows how document A gets a higher ranking after normalizing the term frequency than document B to the query word 'house.'

The matrix is a document-term matrix, where rows represent sentences or documents and columns represent unique words in the corpus (|V|). Unlike previous approaches, the values in the matrix are not binary values or frequency count integers; instead, they are normalized term-frequency floating point values or scores. In NTF, the importance of a word for other words in that document is taken into account.

When the NTF vectors are projected into the vector space, it is possible to compute similarity scores between query and document, or between documents, by looking at the cosine angle between the respective vectors. The cosine value between the vectors is nothing but a portion of the longer vector covered by the shorter vector's (perpendicular) projection onto the longer one. The cosine similarity between two

TABLE 5. TF-IDF embedding example.

	house	apartment	repair	maintenance	labor-cost	safety	commute	school	grocery	env
doc-A	125	210	100	200	32	0	0	0	0	0
doc-B	150	235	0	0	0	71	55	81	32	64
doc-C	175	261	150	350	21	0	0	0	0	0
doc-D	140	284	0	0	0	98	44	76	28	53
IDF	1	1	2	2	2	2	2	2	2	2

vectors 'x' and 'y' is computed using the equation 2.

$$\cos(x, y) = \frac{x.y}{\|x\|\|y\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} (x_i)^2}\sqrt{\sum_{i=1}^{n} (y_i)^2}} \quad (2)$$

The closer a cosine similarity value is to 1, the closer the two vectors are, and the smaller is the angle between them. That is, if two vectors have a cosine similarity closer to 1, it implies that the documents are using similar words in proportion. If two documents use similar words in proportions, the resulting vectors will have the same directions irrespective of their length or magnitude.

The documents whose vectors are close to each other will likely discuss the same thing. The length or magnitude of the vector represents the size of the document. The cosine similarity of 0 represents perpendicular vectors, sharing nothing in common. That is, these vectors have a completely different set of words. However, that does not mean they are talking about entirely different things since they can use a different set of words (synonyms) to discuss the same topic. It will be discussed later on how word-embedding vectors address or resolve this issue by successfully capturing the semantics aspects of words.

In this NTF approach, as the importance of words concerning other words in the document is taken into account, it leads to more precise and accurate results than previous techniques. However, while computing the frequencies, words were considered independently. Also, to cluster the documents together, the same group of words (exact match) needs to occur in them in a similar proportion to increase their similarity score. So similar to previous approaches, this approach also fails to consider the semantic aspects of words, such as synonyms, antonyms, analogies, etc. This approach was again count or frequency based, but a better version than BOW and OHE since it could find the relative importance of words to a document.

F. TF-IDF EMBEDDING

Although NTF does tell the importance of a word relative to the other words in the document, it does not capture the importance of a word relative to the rest of the documents in the corpus. This might lead to erroneous results since there might be different documents. However, just because they are using the words prevalent across the corpus (such as: 'and, of, the, that,' etc.), NTF will end up assigning high similarity scores and hence clustering them together. To compute similarity, we want to consider frequent words in a document (measured by NTF) and demonstrate their importance or uniqueness by occurring in fewer documents across the corpus.

Inverse Document Frequency (IDF) metric, which was proposed by Jones [21], gives importance or weightage to such words which are unique to a certain set of documents (representing a topic or concept), and that can therefore help distinguish and classify documents easily. IDF indirectly penalizes the similarity score if the words, which are not unique to the documents, are considered while clustering those documents. As shown in equation 3, IDF is the ratio of the total number of documents in the corpus to the number of documents the term appears in.

$$IDF_i = log\frac{N}{df_i} \quad (3)$$

where 'N' is the total number of documents in the corpus, and df_i is the number of documents (document frequency) containing term t_i. NTF was used to increase the similarity between vectors if they shared similar words in similar proportions. In contrast, IDF reduces the similarity between vectors if they share similar words that are not unique to those documents and might frequently occur in the corpus. As shown in equation 4, taking the product of TF and IDF (tf * idf) assigns a numeric value to the importance of that word in the given document, given its usage across the entire corpus. In addition, several papers, such as [22] and [23], have provided detailed explanations on the working of TF-IDF.

$$w_{i,j} = tf_{i,j} idf_i = tf_{i,j}\left(log\frac{N}{df_i}\right) \quad (4)$$

where, $tf_{i,j}$ is the term frequency of term i in document j. idf_i is the inverse document frequency of term i, and df_i is the document frequency or the number of documents in which the term appears. TF-IDF helps compute the importance of a word in a document relative to the rest of the corpus. TF-IDF, in a way, implicitly embeds and upscales the weights of the words into the document vector that are important in highlighting and classifying that document vector. Using such TF-IDF metric, vectors can bring out the concepts and topics in the document. References [24], [25], and [26], highlight the applications of tf-idf in domains such as text classification, information retrieval, etc. Table 5 shows an example of TF-IDF vectors.

Consider a corpus of four documents (A, B, C, and D), using NTF metric all these documents are termed equally similar as they use words such as ''house'' and ''apartment'' in proportion. So this does not yield or result in any meaningful insight. What if, instead of focusing on these prevalent words, by using TF-IDF metric more focus or importance is given to the words unique to documents A and C, such as: ''repair'', ''maintenance'', ''labor cost'', and the words that

are unique to documents B and D such as ''safety, ''commute'', ''schools'', ''grocery'', ''environment'' etc. This tells us that documents A and C discuss the cost incurred in repairing homes or apartments, and B and D talk about neighborhood communities. In short, using TF-IDF helps select or focus on words that are important to the documents and can therefore help distinguish the document from the rest of the corpus.

In terms of accuracy, TF-IDF vectors outperform the previously listed approaches. However, the TF-IDF matrices are high-dimensional and sparse. To address this issue, dimensionality reduction techniques were applied to the TF-IDF (or BoW) vectors. Furthermore, sometimes two lemmatized TF-IDF vectors close to each other are not similar in meaning. Also, synonyms with different spellings produce TF-IDF vectors that are not close to each other in the vector space. Therefore, even a state-of-the-art TF-IDF similarity score, such as cosine or Okapi BM25, would fail to connect the synonyms or push apart the antonyms.

G. HYPERSPACE ANALOGUE TO LANGUAGE (HAL)

HAL was proposed in [27], where a co-occurrence matrix is built, which provides coordinates in a high dimensional semantic space enabling analysis of word relationships. As stated in [28], such numeric vector representation in HAL captures enough information to make a semantic, grammatical, and abstract distinction. Before HAL, such semantic spaces were constructed by manually defining the number of axes and the meanings for each axis in the space. However, such an explicit effort was judgment-based and, therefore, error-prone, as the deduced set of axes might not represent or capture the desired level of details in the built space. So instead, in HAL, the lexical co-occurrence technique was used to automatically derive these axes in semantic space.

The co-occurrence matrix in HAL is a square matrix, where the rows and columns are equal to the vocabulary size, |V|. To build this matrix, a context window spans a couple of words. For a given word (represented by a row), all the words appearing before it in the current context window are considered co-occurring. For such co-occurring words, the values in the corresponding columns are incremented by 1. It records the entries corresponding to the number of times a word occurs in the context of another word. This context window slides over the source corpus by one-word increment. Each cell in such a co-occurrence matrix is the co-occurrence count for the given word pair, represented by a row and column. Consider an example: ''He ran towards the grocery store,'', where window size = 4.

Table 6 showcases the co-occurrence matrix built for the example in consideration. After the construction of matrix, the distance metrics is applied to find the semantic similarity between pair of words. Such similarity score is used to categorize and classify the documents. For example, the similarity between ''store'' and ''grocery'' vectors is strong, as compared to ''store'' and ''ran'' vectors. That is, the probability of ''store'' and ''grocery'' co-occurring together in a document is higher than ''store'' and ''ran''. Therefore, if the vectors are plotted in n-dimensional space, the vectors closed to each other exhibit a meaningful cluster.

TABLE 6. Hyperspace analogue to language example.

	He	ran	towards	the	grocery	store
He	0	0	0	0	0	0
ran	3	0	0	0	0	0
towards	2	3	0	0	0	0
the	1	2	3	0	0	0
grocery	0	1	2	3	0	0
store	0	0	1	2	3	0

In HAL, as each word vector is represented based on the context words in which it appears, the meaning of the word is indirectly determined by the context in which it appears. However, the problem with HAL approach is that frequently occurring words will end up contributing disproportionately to the similarity measure. That is, frequent words in the contexts might have very little to convey about the semantic relatedness of the given words but will end up having large effect on the similarity score of those words. For example, words ''grocery'' and ''sports'' are not strongly related. However, if they end up sharing similar words in their contexts, then their vectors will be projected closer to each other, implying high similarity score between them.

IV. DIMENSIONALITY REDUCTION TECHNIQUES

In the discrete space techniques, the corpus was represented numerically using either one-hot, bag-of-words, n-gram, frequency-based, TF-IDF, or co-occurrence technique. These approaches involved the construction of vectors and matrixes, which were simple to implement. However, they suffered from the dimensionality curse and data sparsity, requiring more time to train the models. Additionally, although tf-idf does help in identifying words that help discriminate the documents, it does not capture the inter or intra-document statistical structure represented by topics. To overcome these limitations, dimensionality reduction techniques were applied to capture and present the same information but with fewer dimensions. There are mainly two types of DRTs: feature selection and feature transformation.

A. FEATURE SELECTION

In feature selection, only a subset of dimensions is selected as features from the original set of dimensions. In feature selection, there are primarily four methods: Document Frequency (DF), Term-Frequency Variance (TFV), Mean TF-IDF (TI), and Information Gain (IG). In all these methods, the terms are first sorted on some score, and then a subset of terms are shortlisted based on specific thresholding criteria.

1) DOCUMENT FREQUENCY

Document Frequency for a term t is defined as the number of documents in which the term t appears at least once. Once the terms are sorted based on their DF, top k dimensions satisfying certain set of threshold values are selected as features.

The terms below the threshold are considered non-influential or non-informative for category prediction.

2) TERM-FREQUENCY VARIANCE

In this approach, the variance of term frequency is computed using the following formula, which captures or represents the quality of the terms [31]. The terms are then ranked based on their variance value, and top-k terms are selected as features.

$$TfV_i = \sum_{j}^{n} tf_j^2 - \frac{1}{N}\left[\sum_{j}^{N} tf_j\right]^2 \tag{5}$$

3) MEAN TF-IDF

In this technique, once the TF-IDF values for each term are computed, the mean value of TF-IDF (represented as TI) for each term over all the documents is calculated as a measure that represents the importance or quality of that term [29]. Then, the terms are ranked on the basis of their TI values, and top-k terms are selected which meet the threshold criteria.

$$TI_i = \frac{1}{N}\sum_{j=1}^{N} tf_j idf_i \tag{6}$$

In equation 5 and 6, tf_j is the term frequency of term i in document j, 'idf' is the inverse document frequency of term i, and N is the total number of documents.

4) INFORMATION GAIN

Information Gain (IG) is a goodness-criteria for terms to check how the presence or absence of a term affects the information obtained for category prediction [30]. Information gain helps determine which attribute from the given features is most helpful in discriminating between the classes to be learned. As stated in [30], Information Gain of term 't' is defined or calculated as:

$$\begin{aligned} IG(t) = &-\sum_{i=1}^{m} P(c_i)\, logP(c_i) \\ &+ P(t)\sum_{i=1}^{m} P(c_i|t)\, logP(c_i|t) \\ &+ P(\bar{t})\sum_{i=1}^{m} P\left(c_i|\bar{t}\right) logP\left(c_i|\bar{t}\right) \end{aligned} \tag{7}$$

In equation 7, m is the number of classes or categories in the target space. Based on their IG value, the features having information gain below a certain predefined threshold are removed. IG values help numerically compute a given feature's importance to the classification task.

B. FEATURE TRANSFORMATION

In feature transformation, the original high-dimensional space is projected onto a lower dimensional space. Each dimension from the lower space is either a linear or non-linear combination of the dimensions from the original space. Well-known linear transformation methods include latent semantic indexing, latent Dirichlet allocation, random indexing or projection, and independent component analysis. Reference [29] covers a comprehensive list of DRT methods, especially those used for text clustering applications.

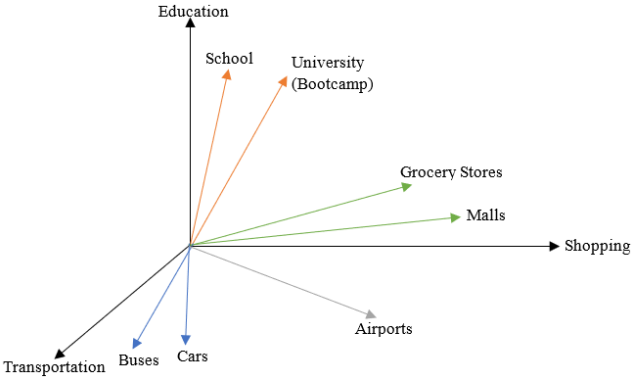


FIGURE 2. Latent semantic indexing example.

1) LATENT SEMANTIC INDEXING (LSI)

As stated by J. R. Firth, "You shall know a word by the company it keeps." The semantic similarity between two natural language expressions (or individual words) is proportional to the similarity between the contexts in which words or expressions are used. The algorithmic way is to count the co-occurrences of words and construct a topic vector using these groups of co-occurring words. LSI proposed in 1990 by Deerwester [32], manages to derive these topic vectors using Singular Value Decomposition (SVD) technique. If there are words that co-occur a lot, then it implies that they share something in common and can therefore be clubbed together under one topic vector. As a result, the documents frequently sharing co-occurring terms will have similar representations or projections in this latent semantic space.

In LSI, the task is to identify a group of words that represent a particular topic and then use the weighted (TF-IDF) vectors of those individual words to compute the vector of the topic. The words that strongly represent that topic are given more weight, whereas those that don't represent or identify that topic are assigned lesser or negative weights. Therefore, the dimensions or features in LSI represent a weighted linear combination of original tf-idf frequencies. Because of this transformation, LSI manages to compress the number of column vectors initially equal to the vocab size to merely a few columns representing topics. Furthermore, clubbing words into topics or concepts significantly reduces the number of features or dimensions in LSI. These topics are further used to compare (classify, cluster) the documents in the corpus. For example, on a corpus related to the "community" subject, LSI might deduce topics such as education, shopping, transportation, etc. Then, as shown in Figure 2, the documents are more precisely clustered based on their distance from these dimensions or topic vectors.

LSI technique involves linear transformation (rotation and stretching) of the given TF-IDF vectors and lining up them

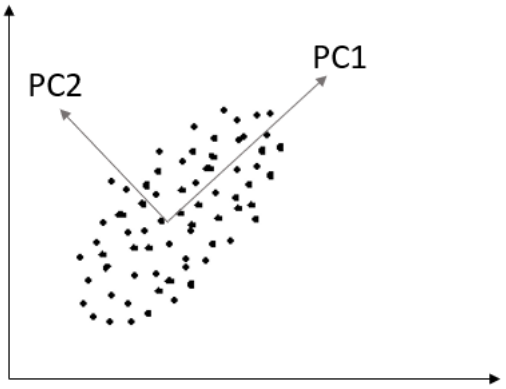


FIGURE 3. Princinpal component example.

to generate the highest spread. To achieve this, LSI uses SVD on the original matrix to identify a linear subspace that captures most of the variance in the space of tf-idf features. The ideal topic vectors generate the highest spread or variance among the documents because they have a better selection of words. Therefore, they make it easier to classify, cluster, and find similarities among the documents. LSI sorts these topic vectors and retains only those limited topics sufficient to classify the documents without compromising on the error.

Using SVD, LSI seeks to reduce the dimensions of the data by finding orthogonal or non-statistical related linear combinations or Principal Components (PC) of the original variables having the largest variance. For example, as shown in Figure 3, PC1 is the first principal component and goes in the direction with the largest variance. PC2 is the second principal component that is orthogonal to the first and goes in the direction with the second largest variance.

LSI works better if the input consists of smaller documents since the cooccurrence values and word meaning are well represented or captured in smaller sentences compared to longer articles. Moreover, as LSI detects high-order semantic structure between terms and documents (in the form of topics), it can resolve one of the main challenges encountered in NLP, synonyms or semantically related words. It, therefore, acts as one of the best tools for automatic indexing. As demonstrated in [33], LSI helps improves access to textual information in information retrieval. However, at the conceptual level, the representation obtained by LSI cannot solve the problem of polysemy. This problem was addressed by PLSI, as PLSI associates a latent context variable with each word occurrence, which explicitly accounts for polysemy.

2) PROBABILISTIC LATENT SEMANTIC INDEXING (PLSI)

Hofmann proposed PLSI in 1999 in [34] that was based on the likelihood principle, which helped minimize the word perplexity and solve the polysemy problem. As stated in [35], PLSI defines a proper generative data model to perform a probabilistic mixture decomposition, which results in a more principled approach with a solid foundation in statistical inference than LSI. PLSI uses a latent variable model termed an aspect model, which associates each observation with an unobserved class variable. The generative model can be defined or represented using the following three steps:

1) select a document d_i with probability $P(d_i)$
2) pick a latent class z_k with probability $P(z_k|d_i)$
3) generate a word w_j with probability $P(w_j|z_k)$

The aspect model is a statistical mixture model based on two assumptions: First is similar to the bag-of-words approach, where observation pairs(document d, word w) are assumed to be generated independently. Secondly, the aspect model introduces a conditional assumption, where document d and word w are independent and conditioned on the state of the latent-variable z. The aspect model is a latent variable model for co-occurrence data which associates an unobserved class variable $z_k \in \{z_1, \ldots, z_K\}$ with each observation. Here, observation $P(d, w)$ is the probability of occurrence of a word w in a particular document d.

When translated into a joint probability model, the data generation process results in the expression shown in equation 8.

$$P(d_i, w_j) = P(d_i)P(w_j|d_i), \text{ where}$$
$$P(w_j|d_i) = \sum_{k=1}^{K} P(w_j|z_k)P(z_k|d_i) \tag{8}$$

In equation 8,

- $P(d_i)$ - represents probability of observing word occurrence in particular document d_i
- $P(z_k|d_i)$ - represents document specific probability distribution over the latent variable (z) space.
- $P(w_j|z_k)$ - represents the conditional probability of a word w_j conditioned on the unobserved class variable z_k

Based on the equation 8, PLSI models each word in a document as a sample from a mixture model, where the mixture components are multinomial random variables that can be viewed as representations of ''topics.'' Thus each word is generated from a single topic, and different words in a document may be generated from various topics. Each document is represented as a list of mixing proportions for these mixture components and thereby reduced to a probability distribution on a fixed set of topics. The factor representation obtained by PLSI allows dealing with polysemous words and explicitly distinguishing between different meanings and different types of word usage.

[35] demonstrates how PLSI showcases substantial improvement over LSI regarding perplexity results for different types of text and linguistic data collections. Additionally, compared to LSI, where L2-norm or Frobenius-norm is used to determine the optimal approximation or decomposition, PLSI instead aims at an explicit maximization of the model's predictive power by using the likelihood function of multinomial sampling. Probabilistic Latent Semantic Analysis was thus considered a promising novel unsupervised learning method with a wide range of applications in computational linguistics, information retrieval, text learning, and information filtering.

3) LATENT DIRICHLET ALLOCATION (LDA)

LDA was proposed in [36]. It is a generative probabilistic model of a corpus. LDA models allow documents to exhibit multiple topics to different degrees. The basic idea is that documents are represented as random mixtures over one or more latent topics. Each topic is characterized by using a distribution of words or frequency of terms. The algorithm starts with the assumption that the weights or probabilities of the topics in a document and the weights of words in a topic follow the Dirichlet probability distribution. To find out the benefits offered by LDA, let us compare it with other latent variable models.

In a unigram, latent variable model, the words of every document are drawn independently from a single multinomial distribution, represented in equation 9.

$$P(w) = \prod_{n=1}^{N} P(w_n) \tag{9}$$

In a mixture of unigram models, each document is generated by choosing a topic z and then generating N words independently from the conditional multinomial p(w|z). Here, the word distributions can be viewed as representations of topics, assuming each document exhibits precisely one topic. The probability of a document in this model is shown in equation 10.

$$P(w) = \sum_{z} P(z) \prod_{n=1}^{N} P(w_n|z) \tag{10}$$

In PLSI, document d and word w are conditionally independent given an unobserved topic z.

$$P(d_i, w_j) = P(d_i) \sum_{k=1}^{K} P(w_j|z_k)P(z_k|d_i) \tag{11}$$

The PLSI model attempts to relax the simplifying assumption made in the mixture of unigrams model that each document is generated from only one topic. In a sense, it does capture the possibility that a document may contain multiple topics since $p(z \mid d)$ serves as the mixture weights of the topics for a particular document d. However, here the model learns the topic mixtures $p(z \mid d)$ only for those documents on which it is trained. For this reason, PLSI is not a well-defined generative model of documents, as there is no natural way to use it to assign a probability to a previously unseen document. Additionally, the number of parameters in PLSI grows linearly with the number of training documents. That is, a corpus with V vocabulary size and M documents with k topics will result in $kV + kM$ parameters.

LDA overcomes these problems by treating the topic mixture weights as a k parameter hidden random variable instead of individual parameters associated with the training set. As a result, it generalizes easily with new or unseen documents. Also, the parameters don't grow linearly with the corpus size. Regardless of the number of documents, the k topic LDA model parameters will always be $(k + kV)$.

Unlike LSI, LDA uses a nonlinear statistical algorithm to group words. As a result, it is more precise in allocating the words to topics than the linear approach in LSI. However, because of the nonlinearity, LDA generally takes much longer to train than linear approaches like LSI. Its usage is, therefore, limited and is helpful for some single-document problems such as document summarization.

4) INDEPENDENT COMPONENT ANALYSIS (ICA)

ICA is very closely related to the method called blind source separation [37]. To define ICA, the following statistical "latent variables" model can be used [38], where linear mixtures of 'n' independent components are observed:

$$x_j = a_{j1}s_1 + a_{j2}s_2 + \ldots + a_{jn}s_n \tag{12}$$

The model can also be written as:

$$x = \sum_{i=1}^{n} a_i s_i \tag{13}$$

The ICA model in equation 13 is generative as it describes how the observed data x are generated by process of mixing the independent components s_i. Here it is assumed that each mixture variable x and the independent component s are random variables. These independent components are called latent variables because they are not perceived directly. As the mixing matrix (A) is unknown, using the observed random vectors (x), ICA estimates both A and s_i by making two assumptions. First, ICA assumes that components s_i are statistically independent and have nongaussian distributions. Statistically independent means the information on the value of one component s_1 doesn't give any information on the value of another component s_2.

ICA is a statistical technique where random data are linearly transformed into components that tend to be maximally independent. Unlike LSI, Independent Component Analysis (ICA) seeks linear combinations or projections of dimensions that are not necessarily orthogonal to each other. Instead, they are as statistically independent as possible, which is a stronger condition than statistical uncorrelatedness. This is because statistical independence involves all the higher-order statistics, whereas LSI is limited to only to second-order statistics. While LSI or Principal Component Analysis (PCA) seeks uncorrelated or orthogonal variables, ICA looks for independent variables. Therefore, in most of the ICA applications, PCA is used as a preprocessing step, where ICA applies additional transformations to the PCA-transformed data matrix. In addition to feature extraction, applications of ICA can be found in many different areas, such as audio processing, biomedical signal processing, image processing, telecommunications, and econometrics.

5) RANDOM INDEXING (RI)

The idea of random mapping rest on the Johnson-Lindenstrauss lemma (Johnson & Lindenstrauss, 1984), which states that if points are projected in a vector space

into a randomly selected subspace of sufficiently high dimensionality, the distances between the points are approximately preserved. As computation on very high dimension space becomes difficult, using RI, the high-dimensional model can be projected to lower dimensionality without compromising on the accuracy of the results. In high-dimensional space, a much larger number of almost orthogonal than orthogonal directions exist. Thus, the dimensionality of a given matrix X can be reduced by multiplying it with (or projecting it through) a random matrix R, resulting in a new matrix X'. In other words, if a given matrix X needs to be projected from m dimensional to a lower p dimensional space, then as shown in equation 14, it can be multiplied by random matrix R to generate its equivalent lower dimensional representation

$$X^{RP}_{pxn} = R_{pxm} X_{mxn} \tag{14}$$

In equation 14, X_{mxn} is the original training samples in features space, R_{pxm} is a random matrix, and X^{RP}_{pxn} is the training samples randomly projected into a reduced dimension feature space. This is possible because, in high dimensional space, random vectors may be sufficiently close to orthogonal, that is $R^T R \approx I$, where I is the identity matrix if the random vectors in matrix R are orthogonal so that $R^T R = I, then X = X'$. However, if the random vectors are nearly orthogonal, then $X \approx X'$ in terms of the similarity of their rows.

Given points $x_1, x_2, \ldots x_n \in R^d$, where d is large, Random Indexing constructs a projection or mapping, π:

$$\pi : R^d \rightarrow R^k, \tag{15}$$

where, $k < d$ such that all distances are ''nearly preserved''. That is,

$$\|x_i - x_j\|_{R^d} \approx \|\pi(x_i) - \pi(x_j)\|_{R^k} \tag{16}$$

where, π is a random projection.

In other words, Random indexing proves or implements the following Johnson-Lindenstrauss theorem.

Let $0 < \epsilon < 1, n \in N$. Then, for any set of n points $x_1, x_2, \ldots x_n \in R^d$, there exists a map $\pi : R^d \rightarrow R^k$, such that for all $i \neq j$:

$$(1-\epsilon)\|x_i - x_j\|^2_{R^d} \leq \|\pi(x_i) - \pi(x_j)\|^2_{R^k} \leq (1+\epsilon)\|x_i - x_j\|^2_{R^d} \tag{17}$$

That is, if the distance in the original dimension is represented by $\|x_i - x_j\|^2_{R^d}$, then it should be equal to the distance in the new lower dimensional space represented by $\|\pi(x_i) - \pi(x_j)\|^2_{R^k}$, where $(k < d)$ and a deviation of ϵ can be tolerated.

RI can help speed up latent semantic indexing (LSI) if RI first reduces the dimensionality of the data and the burdensome LSI is only computed in the new low-dimensional space. As discussed in [39], unlike other dimensionality reduction techniques, RI is scalable and incremental; that is, it doesn'' need an initial sampling of the entire data. Most other reduction techniques require access to the entire data before similarity computations can be performed. For example, in SVD, constructing co-occurrence matrix and then transforming it has to be done from scratch every time new data is encountered. Whereas RI is incremental, it can find the similarity after going through just a few examples. Therefore, the RI technique is computationally less expensive than SVD.

Additionally, once set, the dimensionality in RI doesn'' change, whereas in other methods, it is subject to change as new data is encountered. Unlike LSI, where first the construction of co-occurrence is needed, followed by a separate dimension reduction phase, RI is incremental and does not require a different dimension reduction phase. Furthermore, in RI, the new dimensions are generated randomly. They are random linear combinations of the original terms with no ordering of importance. These new dimensions are only approximate orthogonal. It takes a fraction of the time that PCA requires, and yet the results generated by RI are comparable to PCA.

V. CONCEPTUAL EMBEDDING

In the traditional BoW approach, the text categorization was based on word occurrences in the given corpus or training set. However, humans classify documents by considering common sense, background knowledge, and experience. Explicit Semantic Analysis (ESA) tries to incorporate this information and enrich the document representation using concepts readily available from Wikipedia. ESA was proposed by Gabrilovich in 2007 [40]. Here, the text is represented as a weighted mixture of a predetermined set of natural concepts, for example, from Wikipedia.

It relied on the information or concepts from Wikipedia to represent the meaning of the text in semantic space. Concepts from Wikipedia helped in considering or incorporating the domain-specific world knowledge, to make the semantic representation more fine-grained. Such derived features leverage information that cannot be deduced from the given corpus alone. This helped ESA represent the meaning of the text in terms of Wikipedia-based concepts and to compute the degree of semantic relatedness between individual words or fragments of long texts as represented in equation 18, a semantic interpreter is used by ESA to map the given text into a weighted sequence of Wikipedia concepts ordered by relevance.

$$Text_i = W_1 * Concept_1 + \cdots + W_n * Concept_n \tag{18}$$

As the text gets represented into weighted vectors of concepts, these vectors are called interpretation vectors. The meaning of a text can thus be interpreted or represented as its affinity or closeness with a list of Wikipedia concepts. To find the semantic relatedness among texts, their respective interpretation vectors can be compared using some distance metrics such as cosine.

The inverted index is maintained to map each word to a list of concepts in which it appears. The semantic interpreter then iterates over the words of the given text and retrieves and merges their corresponding entries of concepts from the inverted index into a weighted vector. Entries of this vector reflect the relevance of the corresponding concepts to text T.

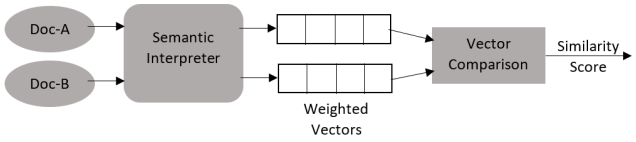


FIGURE 4. Explicit semantic analysis components.

When concepts in each vector are sorted in the decreasing order of their score, the top k concepts are the most relevant ones for the input text.

This approach differs from the count or frequency-based approaches because it doesn'' rely on the frequency or presence of words; instead, the vectors in ESA rely on the concepts encountered in Wikipedia. It also differs from LSI since the concepts from ESA are more grounded in human cognition rather than the ''latent concepts or topics'' in LSI that were nothing but a linear combination of the tf-idf features. Here, the feature generation process was accomplished by contextual analysis of document text, where the documents were mapped onto the Wikipedia or ontology concepts to generate features. Because of a contextualized way of generating the features, ESA helps solve NLP problems such as synonyms and polysemy.

VI. DENSITY DISTRIBUTION EMBEDDING

Instead of projecting words in discrete or continuous vector space, this approach represents words using density-based distributed embeddings, where the mean and covariance matrix are learned from the data. Here, each word is mapped to a Gaussian distribution over a latent embedding space, such that the properties of the distributions and the relationships between the distributions capture its linguistic properties. This allowed the words to be represented as densities over a latent space, directly representing notions of probability mass, and uncertainty and enabled a richer geometry in the embedded space.

This work of Gaussian Embeddings presented in [53], is more expressive than the typical point embeddings because, in this approach, the distribution for one word can encompass the distributions of sets of related words, enabling the representation of concepts such as entailment. However, being unimodal, it is similar to the deterministic approach and, therefore, cannot capture multiple meanings of a word. As Gaussian distribution is limited to one mode, such learned uncertainties will get diffused if the words have multiple meanings (polysemy). In such a scenario, the mean of the Gaussian will get pulled in many different directions, leading to a biased distribution that might center around one semantics. In contrast, the others get excluded or not well represented. Moreover, the mean vector for such a word can get pulled between several clusters, where the mass of the distribution might center on a region or space far from one of the meanings. For example, as shown in the left side of Figure 5, for the word 'wing' with multiple meanings (branch or part), the variance of the learned representation

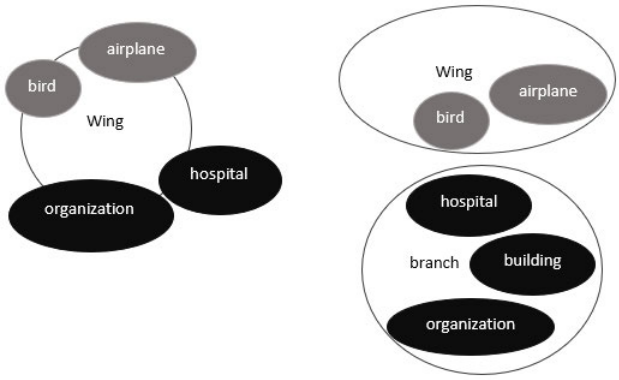


FIGURE 5. Gaussian embeddings example.

becomes unnecessarily large to assign some probability to all the meanings.

To overcome this limitation, instead of using single Gaussian distribution, [54] proposes an approach where each word can be represented or expressed using multimodal distribution to capture multiple distinct meanings. That is, each word is modeled using a mixture of Gaussians. For example, the word 'bank' can have various meanings based on its context. Capturing and learning such multiple meanings of the word offers flexibility and is crucial for applications dealing with predictive tasks.

The multimodal approach from [54] mixes a fixed number of Gaussians and presents a training method to learn the parameters of such a mixture. Here, each word is represented as a Gaussian mixture with K components, and each Gaussian component's mean vector of the word w represents one of the distinct meanings of that word. Moreover, each such component is represented by an ellipsoid, whose center is specified by the mean vector, and the contour surface (represented by covariance matrix) reflects subtleties in meaning and uncertainty. For example, as shown in the right side of Figure 5, one component of the word 'wing' is closer to 'bird' and 'airplane,' whereas the other component is closer to 'hospital' and 'organization'. It also demonstrates how the more general word 'branch' distribution encapsulates words such as 'hospital,' 'building,' and 'organization.'

VII. NEURAL NETWORK BASED EMBEDDINGS

Word embeddings are an essential concept and a popular technique for various tasks in Natural language processing (NLP). They are dense vector representations of words that capture semantic and relational information [106]. In other words, they represent words as real-valued numbers in a high-dimensional space that machine learning methods can easily use. The critical idea in word embedding techniques such as Word2Vec [44] and GloVe [51] is that each word in a language can be represented as real-valued numerical vectors. According to Zhang et al., [107]. Usually, word embeddings are computed at the word level from a large corpus of unlabeled text. In real-world applications and different NLP tasks, the success of word embeddings depends on NLP models

trained on large-scale corpora (i.e., billions of words like the Wikipedia dataset) that can connect the semantic and syntactic relationship, i.e., words and phrases [106].

The embedding approaches discussed so far were more rule-based or statistically grounded. In the neural-network-based method, the models are used to automatically deduce the features, which implicitly represent the syntactic and semantic aspects of the language. So using neural models, the problem of feature engineering is automatically alleviated. Most of these neural network models used language modeling (LM) or Machine Translation (MT) tasks to generate the embeddings. The distributed representations derived from neural networks are real-valued vectors, which capture the semantic aspect of the words. The computational complexity of such a model is determined by the number of parameters that need to be trained.

Neural network-based techniques can broadly be classified into feature-based and fine-tuning-based embeddings. In the feature-based approach, the network is pre-trained to provide language representations, such as word, phrase, sentence embeddings, etc. In a fine-tuning-based approach, the model is pre-trained on a language modeling objective in a self-supervised manner and then finetuned on the downstream tasks (such as classification, NER, QA etc.) with supervised data. Feature-based techniques can either generate static or dynamic embeddings. Static embeddings are non-contextual, as the embeddings remain the same or are static regardless of the context. To learn such word embeddings, shallow networks are used. Whereas, in dynamic embeddings, the embeddings of the same word changes based on the context, hence addressing the polysemy aspect of the words. Below we cover prominent embeddings in each category that were generated using neural network methods.

A. STATIC WORD EMBEDDINGS

1) Word2Vec

Reference [44] proposed two methods to derive embeddings using Feed Forward neural network. In the Continuous Bag of Words (CBOW) approach, given the context, it predicts the target word. In the Skip-gram approach, given the word, it predicts the context. Both the architectures were simple, as no hidden layers were used, and therefore the computational complexity of learning embeddings was much lower. However, most of the complexity in neural networks (such as Feed Forward,and Recurrent) is caused by non-linear hidden layers. So the authors of [44] explored much simpler models which heavily relied on the softmax normalization efficiency.

The proposed neural network model was first used to learn word vectors, and then the N-gram NNLM was trained using these distributed representations of words. To generate the embeddings with 1000 dimensions, the model was trained on the Google news dataset having 6 billion training words. Word analogy (consisting of semantic and syntactic subtasks), word similarity, and sentence completion tasks were used to measure the accuracy and quality of word vectors. The proposed architectures were much more straightforward than feed-forward or recurrent networks, resulting in lower computational complexity.

2) GloVe

Unlike Word2Vec, which only focused on local context, GloVe proposed in [51] took into account local context window and global matrix factorization methods. In the GloVe model, the global corpus statistics are captured directly by the model. When GloVe was tested on the same dataset from [44] consisting of 19,544 questions, it obtained an accuracy of 81.9% on the semantic task and 69.3% on the syntactic task. In addition to the word analogy task, GloVe was also tested on word-similarity and NER tasks, where it outperformed SVD and CBOW.

3) FastText

In continuous space representation models, a distinct vector represents each word. Such representation, however, ignores the internal structure or morphology of words. This may not be suitable for morphologically rich languages having large vocabularies with many words that occur rarely. FastText [52] overcomes this limitation by focusing on character-level information of words and learning representations of subwords. It proposes a skip-gram model in which the words are represented as bags of character n-grams. So instead of assigning the vectors to each word, here, the vectors are allocated to each character n-gram. The vector of a word is represented as the sum of the n-gram vectors of its characters. In short, FastText is an extension to the continuous skip-gram model introduced by Mikolov, which considers subword information that was missing in the earlier skip-gram model.

This model is comparatively faster and allows to compute vectors for words that rarely occur or not at all in the training corpus. The model was successfully tested and evaluated across nine languages exhibiting different morphologies. The experiments were conducted on nine different languages (Arabic, Czech, German, English, Spanish, French, Italian, Romanian, and Russian). For the Word-analogy task, it was observed that subword information significantly improved the accuracy of syntactic tasks. The proposed approach could produce good embeddings or word vectors with tiny training datasets. The experiments also demonstrated how taking long n-grams (for generating subwords) helps in improving semantic accuracy. Additionally, using the sub-word vectors helped enhance the perplexity of the Language Modeling task, especially for morphologically rich languages such as Czech and Russian. Table 7 shows the results of these static embeddings on word-related tasks.

B. DYNAMIC EMBEDDINGS

Instead of random initialization, transferring information using word vectors (such as Word2Vec, and GloVe) has improved performance on various tasks. However, such representation of words is independent of the context. Therefore

TABLE 7. Static word embedding performance.

Embeddings	Training Dataset	Semantic task	Similarity task	Syntactic task	Sentence Completion task
CBOW	Google News	57.3	-	68.9	-
Skip-Gram	Google News	66.1	-	65.1	48
Glove	CoNLL-2003	81.9	83.6	69.3	-
FastText	Wikipedia (nine languages)	77.8	71	74.9	-

they must refrain from disambiguating the word senses based on the surrounding context. As a result, an ambiguous word can refer to multiple, potentially unrelated meanings depending on its context. The ability to derive different representations of the same word based on the context in which it appeared would further improve transfer learning in NLP tasks where contextual representations are essential, such as name entity recognition, word sense disambiguation, and co-reference resolution.

Contextualized word embeddings overcome this limitation by computing dynamic embeddings for words that adapt or change based on context. Advanced and more profound models, such as CoVe, ELMo, GPT, and BERT, focus on generating contextual embeddings. These embeddings are context-sensitive, as they change as per the context. They represent the word semantics depending on its context. A normally pre-trained Language Modeling (LM) task is used to derive contextual embeddings. Pre-trained LMs learn contextualized text representations by predicting words based on their context using large amounts of text data. The LM is trained on a general-domain corpus to capture general features of the language in different layers. Below we discuss prominent contextual embeddings derived using different neural network architectures.

1) Context2Vec (C2V)

Reference [48] proposed an unsupervised model that uses bidirectional LSTM for learning generic (task-independent) representation of wide sentential context around a target word. It can represent variable-length contexts with a fixed-size vector. It borrows the CBOW architecture but replaces the simple neural model with a much more powerful parametric model of bidirectional LSTM. One LSTM is fed input from left to right, and another is fed from right to left. The parameters of both networks are independent, including two separate sets of left-to-right and right-to-left word embeddings. The output of both LSTMs are concatenated. Unlike CBOW's limited context window size, here the entire sentence is considered to generate sentential context. This representation helps capture the relevant information when it is distant/remote from the target word. As a result, the target words associated with similar sentential contexts have similar embeddings. As demonstrated in [48], the embeddings successfully preserve the part-of-speech and tense information because of sentential context. The experiments were conducted on three tasks - Sentence Completion, Lexical Substitution Task (LST), and Word Sense Disambiguation (WSD). The context2vec surpassed AWE (Average-of-Word-Embeddings) across all benchmarks.

2) CoVe

CoVe computes contextualized representations using a neural machine translation encoder. Reference [47] proposed context-Vectors (CoVe) derived from deep LSTM encoder of an attentional seq-to-seq model trained for machine translation task. It trained an LSTM encoder one Machine Translation (MT) task and then transferred the encoder to other tasks in NLP. It showed that attentional encoders trained on NMT task transfer well to other NLP tasks. Furthermore, it demonstrated how the MT data holds potential comparable to CNN's ImageNet as a cornerstone for reusable models. That is, the pairing of MT-LSTM is similar to the ImageNet-CNN pairing of computer vision. Finally, the results illustrated how appending CoVe to the word vectors of the downstream model improved the performance of the downstream task over that of baseline models using pre-trained word vectors alone.

Fixed-length representations derived from the NMT encoder do better on semantic similarity tasks than those obtained from monolingual encoders (e.g., language modeling). The experiments also revealed how the quantity of training data used to train the MT-LSTM was positively correlated with the performance of the downstream tasks. GloVe embeddings of English words are fed to the attentional seq-to-sql bidirectional LSTM model. After the MT-LSTM is trained, the output of the encoder is treated as CoVe. That is, the GloVe word vectors were used to generate context vectors (CoVe). For tasks such as classification and question answering, the GloVe vector of a word is concatenated with its corresponding CoVe vector. The results indicate how (CoVe + GloVe) together achieve higher validation performance on all the classification and question-answering tasks than the models that use GloVe alone. Additionally, appending character n-gram embeddings can boost performance even further. That is, the information provided by CoVe is complementary to both the word-level information provided by GloVe and the character-level information provided by character n-gram embeddings.

3) UNIVERSAL LANGUAGE MODEL FINE-TUNING (ULMFiT)

ULMFiT was proposed in [57], using Language Modeling (LM) task for pretraining. Language modeling is the ideal source task as it captures many aspects of a language, such as hierarchical relations, long-term dependencies, and semantic and syntactic facets. Additionally, unlike Machine Translation, the data for LM is available in abundance. However, ELMo and CoVe train the primary/test task model from scratch and treat the pre-trained embeddings as fixed parameters, limiting their usefulness. ULMFiT paper proposed a fine-tuning transfer technique to avoid task-specific

modifications and training of downstream tasks from scratch, as training from scratch requires large datasets and days to converge.

As stated in ULMFiT, since different layers capture different types of information, they need to be fine-tuned to different extents. ULMFiT proposed a novel fine-tuning approach called discriminative fine-tuning, where each layer can be tuned with varying learning rates to address this issue. Additionally, to adapt the parameters to task-specific features, instead of using the same learning rate, ULMFiT proposes Slanted Triangular Learning Rates (STLR), which first linearly increase the learning rate and then linearly decay it. Furthermore, the authors also propose a gradual unfreezing technique. Rather than fine-tuning all layers at once (which risks catastrophic forgetting), the paper presents gradually unfreezing the model starting from the last layer as this contains the least general knowledge. It first unfreezes the last layer and fine-tunes all unfrozen layers for one epoch. It then unfreezes the next lower frozen layer and repeats until all layers are fine-tuned and until the convergence at the last iteration. All these techniques (discriminative fine-tuning, slanted triangular learning rates, and gradual unfreezing) complement each other and enable the ULMFiT method to perform well across diverse datasets. Based on the results, ULMFiT outperformed state-of-the-art methods on six classification tasks, reducing the error by 18-24% on most datasets.

4) EMBEDDINGS FROM LANGUAGE MODELS (ELMo)

To capture deeper and context-dependent aspects of word meanings, ELMo was proposed [49], which uses a bidirectional LSTM model to derive the word embeddings. ELMo embeddings differ from traditional word type embeddings in that each token or word representation is a function of the entire input sentence. ELMo embeddings were derived from a bidirectional LSTM that was trained on a large text corpus with a Language Model objective. They have learned functions of the internal states of a deep bidirectional Language Model (biLM) that combines representations of forward and backward language models. As they are a function of all internal layers of the biLM, they gave improved performance over using the output of just the top LSTM layer.

It derived the embeddings using a bidirectional LSTM network which was trained with a coupled Language Model (LM) objective. They are computed on top of two-layer biLMs with character convolutions as a linear function of the internal network states. The ELMo learned representations are, therefore, deeper than traditional word vectors since in ELMo the embeddings are a function of all of the internal layers of the bidirectional language model. In ELMo, the higher-level LSTM states capture the semantic and context-dependent aspects, while lower-level LSTM states focus on syntactic aspects. In ELMo, they extract context-sensitive features from a left-to-right and a right-to-left language model. Therefore, the contextual representation of each token is the concatenation of the left-to-right and right-to-left representations.

The embeddings captured the complex characteristics of word usage, such as syntax and semantics, and how their usage varies across linguistic contexts. That is, it successfully modeled polysemy. The paper demonstrated how these representations improved state-of-the-art performance across six challenges. While training, as bidirectional LSTM, is used, ELMo embeddings capture the complex characteristics of word use (e.g., syntax and semantics) and how their uses vary across linguistic contexts (polysemy). Reference [49] demonstrates how the ELMo embeddings significantly improve state-of-the-art across six challenging NLP problems, including question answering, textual entailment, and sentiment analysis. For tasks where direct comparisons were possible, ELMo outperformed CoVe, which computed contextualized representations using a neural machine translation encoder.

5) GENERATIVE PRE-TRAINING (GPT)

Using unsupervised (pre-)training to boost performance on discriminative tasks has long been an important goal of Machine Learning research. Reference [56] proposed GPT, where they explored a semi-supervised approach for language understanding tasks using a combination of unsupervised pre-training and supervised fine-tuning. GPT setup does not require the domain of target tasks and the unlabeled corpus (used for pre-training) to be the same. Language Modeling objective was used on the unlabeled data to learn the embeddings, which were subsequently adapted to the target tasks (using the corresponding supervised objective). The dataset used in GPT for pre-training the model contains long stretches of contiguous text, which allows the generative model to learn to condition on long-range information. GPT achieved new state-of-the-art results in 9 out of 12 datasets.

The paper also analyzed the impact of several layers transferred from the pre-trained model on the accuracy of the downstream tasks (RACE and MultiNLI). It was observed that each transformer [113] layer provided more benefit as it helped increase the accuracy. This indicated that each layer in the pre-trained model contained proper functionality for solving target tasks. To better understand why language model pre-training of transformers is effective, the authors conducted a zero-short learning experiment, and it appeared that attentional memory of the transformer [113] assisted in (better) transfer compared to LSTMs. By pre-training on a diverse corpus with long stretches of contiguous text, the GPT model acquires significant world knowledge and the ability to process long-range dependencies, which are then successfully transferred to solving discriminative tasks such as question answering, semantic similarity assessment, entailment determination, and text classification, improving state of the art on 9 of the 12 datasets.

6) BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS (BERT)

Pre-trained word embeddings offer benefits over learning the embeddings from scratch. For the pre-training purpose, the left-to-right LM objective has been used. However, BERT

TABLE 8. Contextual embeddings - performance on GLUE tasks.

	Co-LA	SS-T2	MR-PC	ST-SB	QQP	QN-LI	RTE	MN-LI m	MN-LI mm
ELMo	36	90.4	84.9	73.3	64.8	79.8	56.8	76.4	76.1
GPT	45.4	91.3	82.3	80	70.3	87.4	56.0	82.1	81.4
BERT-base	52.1	93.5	88.9	85.8	71.2	90.5	66.4	84.6	83.4
BERT-large	60.5	94.9	89.3	86.5	72.1	92.7	70.1	86.7	85.9
UNILM	61.1	94.5	90	87.7	71.7	92.7	70.9	87	85.9

employs a bidirectional Transformer encoder to fuse both the left and right context to predict the masked words. Reference [55] pre-trained deep bidirectional Transformer encoder representations from the unlabeled text by joint conditioning on the left as well as the right context in all the layers. There are primarily two steps in the BERT framework - pre-training and fine-tuning. The model is trained on unlabeled data over different pre-training tasks during pre-training. For fine-tuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters.

As pointed out by the authors in [55], if a unidirectional approach is used for pre-training, then the representations might be detrimental to sentence-level and token-level downstream tasks. For example, it is crucial to incorporate context from both directions in the question-answering task. Unlike GPT, which uses left-to-right architecture, where every token can only attend to previous tokens in the self-attention layer, in BERT context, from left and right is taken into account.BERT uses Masked Language Model (MLM) for pre-training and is, therefore, successful in alleviating the unidirectionality constraint. In MLM, some of the tokens are randomly masked, and the objective is to predict the vocab-id of the masked tokens based on the context in which it appeared. MLM helps enable the representation to fuse the left and the right context. Additionally, BERT also uses Next Sentence Prediction (NSP) task for pre-training.

Compared to ELMo where the shallow concatenation of independently learned embeddings is done, BERT uses MLM to enable deep bidirectional representations. Many important downstream tasks, such as Question Answering (QA) and Natural Language Inference (NLI) are based on understanding the relationship between two sentences, which is not directly captured by language modeling. Therefore, to train a model that understands sentence relationships, it pre-trains for a binarized next-sentence prediction task. Table 8 results show how BERT outperformed GPT and ELMo on all the GLUE tasks. Also, BERT-large did significantly better than BERT-base. Paper [55] demonstrated the importance of bidirectional pre-training for representations.

7) UNIFIED PRE-TRAINED LANGUAGE MODEL (UNILM)

Although BERT significantly improves the performance of a wide range of natural language understanding tasks, its bidirectionality makes it challenging to apply to natural language generation tasks. Reference [58] proposes UNILM that can be applied to both natural language understanding (NLU) and natural language generation (NLG) tasks. UNILM is a multi-layer Transformer network, jointly pre-trained on large amounts of text, optimized for three unsupervised language modeling objectives. The unified LM is jointly pre-trained by multiple language modeling objectives, sharing the same parameters. The shared parameters (or pre-trained unified LM) are fine-tuned and evaluated on various datasets, including language understanding and generation tasks. Like BERT, the pre-trained UNILM can be fine-tuned (with additional task-specific layers if necessary) to adapt to various downstream tasks.

But unlike BERT, which is used mainly for NLU tasks, UNILM can be configured to aggregate context for different types of language models and thus can be used for both NLU and NLG tasks. One of the advantages of UNILM is that the parameter sharing makes the learned text representations more general because they are jointly optimized for different language modeling objectives where context is utilized in different ways, mitigating overfitting to any single LM task. Furthermore, in addition to its application to NLU tasks, the use of UNILM as a sequence-to-sequence LM makes it a natural choice for NLG. UNILM uses the masking technique to control how much context the token should attend to when computing its contextualized representation. After pre-training (learning the contextualized representation), they are fine-tuned using task-specific data of downstream tasks. Each input token's vector representation is computed by summing the corresponding token embedding, position embedding, and segment embedding. As the Bidirectional LM pretraining task encodes contextual information from both directions, it generates better contextual text representations than its unidirectional counterpart. The results of these dynamic contextual embeddings on GLUE tasks are highlighted in Table 8.

C. FINE-TUNING-BASED EMBEDDINGS

Most research on word embedding implementations usually focuses on general-domain text generation. However, as the authors in [108] demonstrate, such general-domain applications do not work optimally when used in the domain-specific analysis of very large corpora, for example, in the biomedical domain. Below, we discuss variants of word embeddings used in the NLP domain, where the Fine-tuned embeddings capture or adapt to the change in context and the downstream tasks.

Unlike previous approaches where the embeddings were not updated once fed to the system, these embeddings get further fine-tuned when used on downstream tasks. Pre-trained LMs learn contextualized text representations by predicting words based on their context using large amounts of text data and can be fine-tuned to adapt to downstream tasks. After the

embeddings (learned representations) are transferred, they are fine-tuned on the target task data to learn task-specific features and improve the downstream task performance.

Although generic-domain data and tasks are used for pre-training, the data for the target tasks will be from different distributions. So the pre-trained (LM) model is fine-tuned on data of the target task to adapt to the peculiarities of the target data. Also, the fine-tuning approach converges comparatively faster than training the model from scratch. In a fine-tuning-based system, the embeddings generated from pretraining are fed as input to the model, which is then refined or adjusted to improve the downstream task performance.

Based on the type of downstream task, fine-tuning-based embeddings can be broadly classified as:

1) Cross-Lingual Embeddings
2) Knowledge-Enriched Embeddings
3) Domain-Specific Embeddings
4) Multi-Modal Embeddings
5) Language-Specific Embeddings

1) CROSS-LINGUAL EMBEDDINGS

Much research has focused on pre-training models on large unlabeled corpora in the English language, then fine-tuning it on specific target tasks with smaller amounts of supervised data in English. This approach is monolingual, as the language used for pre-training and fine-tuning is the same. However, there has been a recent surge in the cross-lingual method, where the source (pre-training) and target (fine-tuning) languages are different. This multilingual approach results in embeddings spanning across multiple languages. Such encoders trained in multi-languages can encode sentences from other languages into the same embedding (shared semantic) space. These cross-lingual models can be divided into cross-lingual understanding (XLU) models and cross-lingual generation (XLG) models. Below we discuss prevalent models in each of these categories.

a: M-NMT

Multilingual NMT (M-NMT) [61] proposed by Google uses a standard NMT system. Because parameters get shared by all the language pairs, the model generalized well beyond language boundaries. Additionally, the quality of the low-resource language pairs was improved significantly because of such generalization. The paper explored many-to-one, one-to-many, and many-to-many experiments, where cardinality indicates several languages involved in the setup. After pre-training on 12 language pairs, the model could perform translation between language pairs not seen during training, thus demonstrating zero-shot translation and true transfer learning.

b: M-BERT

Unlike BERT, which was trained solely on English Corpus, M-BERT is trained on the Wikipedia pages of 104 languages with a shared word-piece vocabulary. Here, neither cross-lingual objective nor any parallel (aligned) data was used for training. Furthermore, it never encouraged translation equivalent pairs to have similar representations. Yet, M-BERT performed cross-lingual generalization surprisingly well, where task-specific annotations from the source language are used to fine-tune and evaluate the target language model. The paper [59] highlights how the transfer is possible even to languages in different scripts and that transfer works best between typologically similar languages. They experimented to see the effect of vocabulary overlap between source and target language on the accuracy, and it was found out that m-BERT demonstrated promising results on zero-shot cross-lingual model transfer. However, the paper also showed how these representations exhibit systematic deficiencies affecting specific language pairs.

Additionally, as demonstrated in [60], the lexical overlap between languages plays a minor role in the cross-lingual success of M-BERT. In contrast, the depth of the network was an integral part of it. That is, M-BERT enabled a much deeper representational capacity than simple vocabulary memorization and generalized well. It also showed how performance improves with similarity, delivering that it is easier for M-BERT to map linguistic structures when they are more similar.

c: XLM

XML technique proposed in [62] introduced new Translation Language Modeling (TLM) objective for improving cross-lingual pre-training. TLM objective is an extension of MLM, where parallel sentences from source and target languages are concatenated instead of considering monolingual text streams. It randomly masks words in both the source and target sentences. For example, for English to French translation, the model can either attend to surrounding English words or the French translation words to predict a word masked in an English sentence. This encourages the model to align the English and French representations and is helpful when the model cannot infer the masked English words. It can leverage the French context to predict the word. It also demonstrated how cross-lingual language models improve the perplexity of low-resource languages. For example, XLM trained in Nepali can be benefited or enriched with high-resource languages, such as Hindi, especially when they share a significant fraction of vocabulary.

d: UNICODER

XLM used only a single cross-lingual task (TLM) during pre-training. Unicoder proposed in [63] used three cross-lingual tasks for pre-training purposes. These include - MLM, TLM, cross-lingual word recovery, cross-lingual paraphrase classification, and cross-lingual masked language model. MLM is monolingual, whereas the other four task were trained using a Machine Translation dataset. The cross-lingual word recovery task learns the underlying word alignment relation between two languages by leveraging the attention

matrix between the bilingual sentence pairs. Cross-lingual paraphrase classification classifies whether two sentences from different languages have the same meaning. The cross-lingual masked language model (CMLM) is similar to MLM, except the documents fed as input are written in multiple languages. These tasks help Unicoder make a better language-independent encoder and learn the mappings among different languages from several perspectives. The paper demonstrated how fine-tuning multiple languages together could bring further improvement in the result.

e: XLM-R

Unlike NMT, where the source language is translated in the target language, XLG, based on the documents in the source language, has summary or response generation in the target language. Also, different from language understanding, language generation aims to generate natural language sentences conditioned on some inputs from other languages. XLM-RoBERTa (XLM-R) was trained using Transformer based MLM on clean CommonCrawl data in 100 languages, with more than two terabytes of filtered clean data. XLM-R outperforms multilingual BERT (mBERT) on various cross-lingual benchmarks, including XNLI, MLQA, and NER. Reference [64] demonstrated how XLM-R provides strong gains over previous multilingual models like mBERT and XLM on classification, sequence labeling, and question answering and performs exceptionally well on low-resource languages. XLM-R illustrated how pre-training multilingual language models at a large scale could improve cross-lingual transfer tasks performance.

f: MASS

MASS [65] uses a transformer as a sequence-to-sequence model and pre-trains it on the WMT monolingual corpus of four different languages. It pre-trains the encoder and decoder jointly using masked Sequence to Sequence task, where the decoder predicts the masked sentence fragment when the remaining part of the sentence is given as input to the encoder. During pre-training, as the words are masked on the encoder side, it forces the encoder to understand the meaning of unmasked tokens. Furthermore, in addition to the previous tokens on the decoder side, MASS forces the decoder to rely more on unmasked tokens from the encoder to predict the next token.To verify its effectivness, it is fine-tuned on three language generation tasks, including NMT, text summarization, and conversational response generation. For unsupervised NMT tasks, MASS achieved state-of-the-art BLUE scores on three language pairs.

g: XNLG

XNLG [66] uses monolingual for MLM and cross-lingual setting for cross-MLM (XMLM) objectives during pre-training. MLM aims to predict randomly masked words according to the contextual information from the left and right sides. In XMLM, given a parallel corpus, the pair of bilingual sentences are concatenated to a whole sequence and used as the input of MLM. These pre-training objectives encourage the model to utilize the alignment of bilingual sentences to learn to encode cross-lingual texts into a shared embedding space. After the pre-training step, it fine-tunes the pre-trained model on downstream NLG tasks using monolingual data. The sequence-to-sequence model trained in a single language is then evaluated when it accepts multi-lingual input to produce multilingual output.

In the first pre-training stage, it initializes the encoder and decoder parameters using the 15-language pre-trained XLM model. In the second stage, monolingual data from Wikipedia is used for the Denoising Auto-Encoding (DAE) objective and parallel data from MultiUN for the Cross-Lingual Auto-Encoding (XAE) objective. Results have demonstrated how XNLG outperforms MT-based methods on cross-lingual question generation and abstractive summarization. Such cross-lingual transfer especially helps improve the performance of low-resource languages, as it successfully leverages information from rich resource language.

The architectures and languages used during pre-training of the above cross-lingual embeddings are given in Table 9.

TABLE 9. Cross-lingual models.

Model	Architecture	Languages used for pre-training
M-NMT	Seq-2-Seq (GNMT)	12 language Pairs
m-BERT	BERT (Encoder)	Wikipedia pages of 104 languages
XLM	Transformer (Encoder, Decoder)	Wikipedias of the 15 XNLI languages
Unicoder	Transformer (Encoder, Decoder)	15 languages
XLM-R	RoBERTa	CommonCrawl data in 100 languages
MASS	Transformer (Encoder, Decoder)	WMT monolingual corpus of 4 languages
XNLG	10-layer encoder and a 6-layer decoder	15-languages from XLM, Wikipedia (monolingual), and MultiUN (parallel corpus)

2) KNOWLEDGE-ENRICHED EMBEDDINGS

Contextual word representations do not contain any information about real-world entities and, therefore, cannot remember or retrieve factual knowledge about those entities. Therefore, most LMs need to incorporate knowledge information in their pre-training tasks, which can provide rich structured knowledge facts for better language understanding. Knowledge-Enriched Embeddings (KEE) are word embeddings that include additional information or knowledge from external sources, such as ontologies or knowledge graphs. Knowledge information helps enrich the embeddings and achieve better results on knowledge-driven applications, such as entity typing and relation classification and extraction. Knowledge information is captured structurally using a knowledge graph in triplet (h, r, t) format, which is used to

describe a relational fact, where h is a head entity, t is the tail entity, and r is the relation type. Knowledge embedding (KE), generated using knowledge graphs (KGs), effectively embeds the entities and their relationship information. Below we describe popular approaches (techniques) that integrate Knowledge information during the pre-training phase.

a: ERNIE

ERNIE pre-trains a language representation model on both large-scale textual corpora and KGs and simultaneously takes full advantage of lexical, syntactic, and knowledge information. ERNIE proposes a new pre-training task to inject knowledge into language representation by informative entities. Some token-entity alignments are randomly masked, and the system is then required to predict all corresponding entities based on aligned tokens. This procedure was termed a denoising entity auto-encoder (dEA).

In addition to Textual Encoder (T-Encoder) that captures lexical and syntactic information, ERNIE also uses a Knowledgeable Encoder (K-Encoder) for aggregating token and entity embeddings. K-Encoder consists of stacked aggregators designed to encode both tokens and entities and then fuse their heterogeneous features. ERNIE uses the knowledge embeddings trained on Wikidata by TransE as the input embeddings for entities. ERNIE effectively reduces the noisy label challenge by injecting the information obtained from KGs. As a such informative entity, embeddings help ERNIE predict the labels more precisely. The experimental results demonstrated that ERNIE significantly improves knowledge-driven tasks such as Entity Typing and Relation Classification.

b: KnowBert

KnowBert [83] proposes a general method to embed multiple knowledge bases (KBs) into large-scale models and thereby enhance their representations with structured, human-curated knowledge. It presents a mechanism to incorporate multiple KBs into a large pre-trained model with Knowledge Attention and Recontextualization (KAR) method. In KAR, the entity spans in the raw text are first identified. Then the entity linker retrieves their embeddings from KB to construct knowledge-enhanced entity span representation. These representations are then recontextualized with word-to-entity span attention, which allows long-range interaction between entity spans in the context. Because of incorporating Knowledge base information, KnowBERT shows improved perplexity and is better at recalling entity facts. Downstream evaluations demonstrate the better performance of KnowBERT over relationship extraction, entity typing, and word sense disambiguation tasks.

c: SentiLARE

To enhance the language understanding and benefit the downstream tasks in sentiment analysis task, SentiLARE [86] incorporates word-level linguistic knowledge (such as PoS tags and sentiment polarity) into pre-trained models. It proposed a new label-aware MLM pre-training task with two subtasks, early fusion, and late supervision, to construct and integrate knowledge-aware language representation. Given the sentence-level sentiment label, the early-fusion subtask predicts the word's PoS tag and sentiment polarity at masked positions. In the late supervision subtask, it simultaneously predicts masked words, sentence-level sentiment labels, and linguistic knowledge (PoS tags and sentiment polarity). These sub-tasks establish a connection between sentence-level representation and word-level linguistic knowledge, benefiting the downstream SA tasks. Experiments demonstrated how SentiLARE obtained new SOTA performance over various sentiment analysis tasks.

d: K-BERT

K-BERT [84] highlights how too much knowledge incorporation can result in Knowledge Noise, which may divert the sentence from its correct meaning. To overcome the Knowledge Noise (KN) issue, K-BERT introduces a soft position and visible matrix that limits the impact of knowledge. The knowledge layer transforms the original sentence into a knowledge-rich sentence tree for a given input sentence by injecting relevant triples from the Knowledge Graph. The visible matrix is then used to control the visible area of each token, which prevents the change in the original meaning of the input text caused due to too much knowledge injection. Results demonstrate K-BERT significantly outperforms BERT, especially in domain-specific tasks (finance, law, medicine), which indicates how K-BERT can be a better choice for solving knowledge-driven problems that require experts.

e: KEPLER

Unlike other approaches, KEPLER [85] uses entity descriptions to encode and map the text and entities to a unified semantic space. It doesn't use a separate KE model to generate entity embeddings, which requires an entity linker, resulting in additional inference overhead. Additionally, instead of integrating fixed Entity embeddings to provide external knowledge, it encodes the entities from their corresponding descriptions. KEPLER uses entity descriptions to bridge the gap between KE and PLM and align the semantic space of the text to the symbol space of entities in KGs. KEPLER can produce embeddings for unseen entities based on their descriptions, while conventional KE methods can only learn representations for the seen entities during training. KEPLER integrates factual knowledge from external KGs into language representations by jointly training KE and MLM objectives while preserving the strong abilities of PLMs for syntactic and semantic understanding. Because of the factual knowledge and language representation alignment into the same semantic space, KEPLER greatly improves NLP and KE applications, such as relation extraction, entity typing. Such informative entities in KGs have enhanced language representation with external knowledge.

TABLE 10. Knowledge-enriched embedding models.

Model	Architecture	Pre-training dataset	Downstream task	Knowledge Graph
ERNIE [107]	BERT	English Wikipedia (4500M subwords, 140M entities)	RCLS, ET	Wikidata (5M entities and 24M fact triples).
KnowBERT [83]	BERT	Wikipedia and Books Corpus	RE, ET, WSD	Wikipedia, WordNet
K-BERT [84]	BERT	WikiZh (1M entries, 12M sentences, 1.2G) and WebtextZh (4.1M entries, 3.7G)	CLS, QA, NER	CN-DBpedia (5.17M triples), HowNet (52.5K triples), MedicalKG (13.8K triples)
KEPLER [85]	BERT	Wikipedia (2,500M words) and BookCorpus (800M words)	RCLS, ET	Wikidata5M, WordNet
SentiLARE [86]	BERT	Yelp Dataset Challenge 2019 (6.7M reviews, 5 labels)	Sentence & Aspect level SA	word-level linguistic knowledge (PoS tags, sentiment polarity)

Note: CLS - Classification , RCLS - Relation Classification, ET - Entity Typing, RE - Relation Extraction, WSD - Word Sense Disambiguation, QA - Question Answering, NER - Name Entity Recognition

Additionally, the authors in [109] provide the knowledge-enriched word embeddings (KEWE), which uses knowledge graphs to encode the knowledge of reading difficulty into words evaluated on both English and Chinese datasets. In the biomedical domain, the authors in [110] empirically show how using an external semantic knowledge base combined with local contextual information improved the quality of word embeddings used to generate biomedical concepts. Finally, in [111], the authors proposed a Knowledge-enriched answer generator (KEAG) that exploits an external symbolic knowledge base to generate answers. The architecture, pre-training dataset, downstream tasks, and knowledge graph used for each discussed KEE model are listed in Table 10. Overall, KEs can be helpful in a variety of NLP tasks (e.g., entity recognition, relation extraction, and semantic labeling) by integrating both external knowledge sources with local contextual information (semantics) to achieve a much more accurate and robust word embeddings models thus significantly enhancing the quality of word representations.

3) DOMAIN-SPECIFIC EMBEDDINGS

The embeddings generated using large corpora (such as Wikipedia) are more generic and might need to be revised for specific domain tasks. Such pre-trained generic embeddings often yield unsatisfactory results as word distribution shifts from the general to the biomedical domain.

Recently, in research where specialty or domain-specific corpora is used for pre-training, the language model to generate more tailored embeddings has been conducted in a few areas. These are word embeddings trained on very large specific domain corpus or topics such as finance, healthcare, reviews [122], sentiments [123], emotions [124] etc. For example, authors in [114] and [115] empirically demonstrate how word embeddings tailored to a particular domain, such as biomedical, oil/gas and social media (Covid-19 Tweets) can be leveraged to improve performance in NLP tasks. Furthermore, in [116], the authors trained a shallow neural network on a large corpus on Radiology. They empirically demonstrated that domain-specific embeddings performed significantly better on the multi-label classification task.

BioBERT [101] is initialized with the weights from the BERT model. It is pre-trained using biomedical corpora and is fine-tuned and evaluated on three popular biomedical text mining tasks. The results demonstrated how BioBERT significantly outperformed BERT on biomedical NER, RE, and QA.SciBERT [102] is a BERT-based model pre-trained using scientific publications, mostly from computer science and biomedical domains. SciBERT outperforms BERT on the downstream scientific NLP tasks, such as sequence labeling, sequence tagging, text classification, and dependency parsing, as it uses domain-specific scientific text for pre-training.

ClinicalBERT [103] is pre-trained on clinical text corpora. Because of the specialty corpora, the embeddings from ClinicalBERT demonstrated greater cohesion around medical and clinical-operations-related words. ClinicalBERT showed how using domain-specific corpora resulted in better embeddings, yielding performance improvements on three downstream clinical NLP tasks compared to nonspecific or generic embeddings. PatentBERT [104] pre-trains the BERT model using the United States Patent and Trademark Office (USPTO) database, a specific domain corpus of patents. After fine-tuning it on the patent classification task, it outperformed DeepPatent's performance.

To capture compositional sentiment semantics, SentiBERT [105] incorporates contextualized representation with the help of a recursive constituency parse tree structure. Furthermore, SentiBERT demonstrates how contextualized representation can be combined with the syntactic tree structure to capture the semantic compositionality better. The architecture, pre-training dataset, downstream tasks, along with pre-training data size used for these domain-specific models are highlighted in Table 11.

4) MULTI MODAL EMBEDDINGS

The emergence of multimodal embedding models has demonstrated the potential to integrate different types of information and features from audio, visual, and text modalities to create much more robust and enhanced word embeddings via deep neural networks. Below we cover different variants of multimodal embedding techniques, where large neural networks such as [55], [112], and [113] have shown significant improvements when used in the multimodal setting.

a: SPEECH-BASED EMBEDDINGS

Embeddings for audio words are challenging, as the same word token can have several audio signal realizations in

TABLE 11. Domain-specific models.

Domain & Model	Architecture	Pretraining-dataset	Down-stream task	Pretraining Data-Size
Biomedical, BioBERT [101]	BERT	PubMed abstracts and PMC full-text articles	Biomedical NER, RE, QA	4.5B Tokens
Scientific, SciBERT [102]	BERT	1.14M papers from Semantic Scholar	ST, SL, TCLS, RCLS, NER, DeP	3.17B Tokens
Clinical, Clinical-BERT [103]	BERT	MIMIC-III v1.4 database	MedNLI, NER	2M Notes
Patent, Patent-BERT [104]	BERT	USPTO dataset	Patent Classification	3M patents
Sentiment, SentiB-ERT [105]	BERT and RoBERTa	SST dataset	Sentiment and Emotion Classification	

Note: ST - Sequence Tagging, SL - Sequence Labeling, TCLS - Text Classification, DeP – Dependency Parsing, NLI - natural language inference

different utterances. Additionally, the boundaries for audio words in utterances are not available. Finally, the context of the audio word is in audio form, which can sometimes be noisy, confusing, and difficult to handle. As a result, learning semantics from an audio context is hard. To deal with these challenges, [68] proposes SpeechBERT, which learns audio-and-text jointly for the end-to-end Spoken Question Answering (SQA) task. SpeechBERT is pre-trained using both text and audio datasets on the MLM task.

It uses Text-BERT to derive the semantic information of the word from the text data. It then uses an RNN sequence-to-sequence autoencoder to capture and reconstruct the phonetic structure information of the audio words. The autoencoder model learns to retain the phonetic structure of the audio word by reconstructing the original speech feature 'x' as much as possible. At the same time, it also learns to fit into the embedding distribution of the Text BERT, which carries plenty of semantic information for the word tokens into the same embedding space. This enables the model to learn a joint embedding space, integrating semantic and phonetic information extracted from text and audio datasets. The SpeechBERT model was trained on the Spoken SQuAD dataset, in which the text for all audio paragraphs and questions is from the original SQuAD dataset. It achieved the state-of-the-art result on the Spoken SQuAD dataset.

b: IMAGE-BASED EMBEDDINGS

In Image-based embeddings, joint contextualized representations of vision and language are derived. The goal is to bridge vision and language by aligning input text elements with regions in the input image. The self-attention mechanism learns the association between image and text.

In VisualBERT [75], two language model objectives are used for pre-training, where the image and text inputs are jointly processed. The joint processing enables rich interaction among words and objects (regions), enabling the model to deduce intricate or complicated associations between text and image. The pre-training is done on image-captioned data. In the first objective, part of the text is masked so that the model can learn to predict it using the remaining text and the visual context. In the second objective, the model learns to determine if the text and image match or are associated with each other or not. In addition, the objectives help capture the semantics in the image using the associated text.

In ViLBERT [76], the text and visual data are fed into separate streams for language and vision processing, which interact and communicate using co-attentional layers. Such structure makes the interaction between the modalities possible at varying depths of representation. In the multi-head attention, the key-value pairs of each modality is passed as input to the other modality. Therefore, the attention block produces features for each modality conditioned on the other.

B2T2 [77] also considers the visual context in addition to the textual context and demonstrates how a correct integration between them improves the visual question-answering task. The paper evaluates two different architectures. In late fusion architecture, the bounding boxes around objects are ignored, whereas bounding boxes are considered in early fusion architecture. Using bounding boxes was the critical factor in improving the model's accuracy.

LXMERT [78] architecture consists of three encoders - image, text, and cross-modality encoder. The cross-modality is responsible for learning vision-language interaction and aligning them. Pre-training uses masked object prediction (feature regression and label classification), masked language modeling, cross-modality matching, and image question-answering tasks. Unlike traditional MLM, multimodality-based masking helps infer masked features using visual and textual elements. This help learns the cross as well as intra-modality relationships. For images, instead of using the feature map output of CNN, the features of detected objects are used as the embeddings of the image.

VL-BERT [81] has a single stream for the input where each element or token is either a region-of-interest (RoI) from the input image or a word from the input sentence. MLM with visual clues and Masked RoI classification with Linguistic clues tasks are used for pre-training. In the first pre-training task, visual clues are incorporated to capture the dependencies between textual and visual content. The second pre-training task is to predict the label of the masked RoI based on the other clues. These pre-training tasks help aggregate and align visual-linguistic clues according to the element's contents, categories, and positions.

Unicoder-VL [79] uses a cross-modal pre-training framework to capture relationships between visual and linguistic contents and learn their joint representations. Three tasks are used for pre-training - MLM, Masked Object Classification (MOC), and Visual-linguistic Matching (VLM). The first two tasks help in the joint learning of context-aware embeddings that are based on visual and linguistic contents. The third

TABLE 12. Speech- and image-based MultiModal models.

Model	Streams	Pre-training Framework	Pre-training-dataset	Down-stream task
SpeechBERT	Single stream	Share	Spoken SQuAD dataset	SQA
VisualBERT	Single stream	Share	COCO image caption dataset (3.3 M images)	VQA, VCR, NLVR, RPG
ViLBERT	Separate stream	Cross	Conceptual Captions dataset	VQA, VCR, GRE, CBIR
B2T2	Single stream	Late and Early fusion	Conceptual Captions dataset	VCR
LXMERT	Separate stream	Cross	MS COCO, Visual Genome, VQA v2.0, GQA balanced version, VG-QA	VQA, $NLVR^2$
Unicoder-VL	Single stream	Cross	Conceptual Captions dataset, SBU Captions dataset	CBIR, VCR
VL-BERT	Single Stream	Share	Conceptual Captions dataset	VQA, VCR, GRE

Note: SQA - Spoken Question Answering, VQA -Visual Question Answering, VCR -Visual Commonsense Reasoning, NLVR - Natural Language for Visual Reasoning, RPG - Region to Phrase Grounding, GRE - Grounding Referring Expression, CBIR - Caption Based Image Retrieval, NLVR2 - Natural Language for Visual Reasoning for Real

task predicts whether text and image describe each other or not. Overall, these image embedding techniques ground or associate elements from language text to the image regions from the image without explicit supervision. The streams, downstream tasks, pre-training framework, and datasets used for the speech and image multimodal models are enumerated in Table 12.

c: VIDEO-BASED EMBEDDINGS

In Videobert [69], videos are represented using visual words or tokens. To generate these visual words, the clips of 30-frame (1.5 seconds) are created using non-overlapping windows over the input video. A pre-trainned video ConvNet is used to extract the features of each clip. The video-only pre-training task forces the model to learn and forecast videos. The text-video objective helps the model learn the alignment and correspondence between the two domains.

Unlike VideoBERT, CBT [70] trains each modality separately and then uses a cross-modal transformer to maximize the mutual information between modalities. The textual and video sequences may not align since the person might speak about things at time t that are not visible in the frame or clip at time t. The mutual information between modalities is maximized at the sequence level instead of the frame level to address this issue.

VideoTranslate (VideoAsMT) [71] proposes an approach to formulate video understanding as a machine translation task. It presents a generative solution where, instead of optimizing the model to determine whether a given pair

TABLE 13. Video-based MultiModal models.

Model	Streams	Pre-training Framework	Pre-training dataset	Downstream task
VideoBERT	Single	Share	Cooking videos from YouTube	VCLS, VC
CBT	Separate	Joint	HowTo100M, Kinetics videos	AA, VC, AS, AR
Video-Translate	Single	Encoder-Decoder	HowTo	VQA, VC, TVR, VCLS
UniVL	Separate	Joint	HowTo100M	VC, AS, ASL, MSA, TVR
HERO		Joint	HowTo100M, and large-scale TV datasets	TVR, VQA, VC, VLI

Note: VCLS - Video Classification, VC - Video Captioning, AA - Action Anticipation, AS - Action Segmentation, AR - Action Recognition, TVR - Text based Video Retrieval, ASL - Action Step Localization, MSA - Multimodal Sentiment Analysis, VLI - Video and Language Inference

of text and video matches, it trains a language machine to generate spoken text associated with the given video clip. It uses encoder-decoder architecture initialized with T5 [121] weights to generate text from multimodal representations. Unlike encoder-only based architectures, this approach enables learning of autoregressive decoder, which is trained to generate text from the encoder's multimodal embeddings.

UniVL [72] is a flexible model for both video language understanding and generation tasks. Five objectives, including video-text joint, conditioned masked language model (CMLM), conditioned masked frame model (CMFM), video-text alignment, and language reconstruction, are used for pre-training. In addition to individual text and video encoders, UniVL also employs cross encoders to allow interaction between text and video modality features. Finally, a decoder is used to reconstruct the input text.

HERO [79] encodes multimodal inputs in a hierarchical structure. HERO is a hierarchical model that first absorbs visual and textual local context at the frame level, which is then transferred to a global video-level temporal context. For the pre-training purpose, it uses MLM, Masked Frame Model (MFM), Video-Subtitle Matching (VSM), and Frame Order Modeling (FOM) tasks. In FOM, the model predicts the right order of the shuffled video frames. It uses a cross-modal transformer to fuse subtitle text with video frames and a temporal transformer to generate contextual embeddings of the frame based on the surrounding frames as a global context. The streams, downstream tasks, pre-training framework, and datasets used for the speech and image multimodal models are enumerated in Table 13.

5) LANGUAGE-SPECIFIC EMBEDDINGS

Language-specific embeddings are word embeddings trained on large corpora in specific languages to capture the relationship between words and phrases in that language. These

TABLE 14. Language-based models.

Language & Model	Architecture & Pre-training Strategy	Pre-training dataset	Downstream task	Data Size
French, Camem-BERT [88]	RoBERTa, Monolin-gual	OSCAR corpus (French data)	PoS, DeP, NER, NLI	
French, FlauBERT [89]	BERT, Monolin-gual	WMT19, OPUS collection, Wikimedia (French data)	FLUE Bench-mark: TC, PP, NLI, DeP, PoS, WSD	
Finnish, Fin-BERT [90]	BERT, Monolin-gual	News (Yle, STT), Suomi24, internet crawl	PoS, NER, DeP,	234M Sen-tences, 3.3B Tokens
Dutch, BERTje [91]	BERT, Monolin-gual	Books, TwNC, SoNaR-500, Web news, Wikipedia	PoS, NER, SA, SRL	12GB, 2.4B Tokens
Dutch, Rob-BERT [92]	RoBERTa, Monolin-gual	OSCAR corpus	SA, PoS, NER	39GB, 6.6B Tokens, 126M Sentences
Arabic, AraBERT [93]	BERT, Monolin-gual	Arabic Corpus, OSIAN, Arabic news websites	SA, NER, QA	24GB, 70M Sentences
German, Got-tBERT [99]	RoBERTa, Monolin-gual	German portion of the OSCAR data set	NER, TC, 145GB, 21.5B Tokens	
Spanish, es-BERT [95]	BERT, Monolin-gual	Wikipedia, OPUS Project GLUES (Spanish GLUE) benchmark		3B Tokens
Chinese, ERNIE [94]	BERT, Monolin-gual	Chinese Wikepe-dia, Baidu Baike, Baidu news and Baidu Tieba	NLI, SS, NER, SA, QA	
Portuguese, BERTim-bau [97]	BERT, Monolin-gual	brWaC [46] cor-pus	STS, TE, NER	17.5 GB, 2.68B Tokens
Indonesian, In-doBERT [98]	BERT, Monolin-gual	Indonesian, Kompas news articles, Tempo, Liputan6, Indonesian Web Corpus	IndoLEM (PoS, NER, DeP, SA, TS, next tweet prediction, tweet ordering)	220M Tokens
Russian, Ru-BERT [96]	BERT, Multilin-gual	BERT – 100 lan-guages	RC, PAWS, QA	
11 Indian lan-guages, In-dicBERT [87]	ALBERT, Multilin-gual	crawling news articles, magazines and blogposts, OSCAR corpus	IndicGLUE benchmark	8.8B Tokens
Marathi Maha-BERT, Maha-RoBerta, Maha-ALBERT [100]	(BERT, RoBERTa, AL-BERT), Multilin-gual	mBERT-104 languages, XLM RoBERTa - 100 languages	SA, TC, NER	24.8M Sentences, 289M Tokens

language-specific models can enhance the accuracy and performance of NLP tasks such as text classification, sentiment analysis, and named entity recognition. Typically, the pretraining of attention-based models happens in an unsupervised way on large corpora, followed by fine-tuning on a small labeled dataset. In multilingual models, the large corpora consist of data from multiple languages. However, compared to multilingual models, recent studies have shown that monolingual models perform better on NLU tasks.

Such monolingual models demonstrate that pretraining on fewer monolingual datasets is enough to achieve equivalent or better results with larger multilingual datasets. When evaluating the performance of English NLP models, GLUE and SuperGLUE benchmark frameworks are used prominently. Similarly, other languages are gaining momentum and have proposed their multi-task evaluation benchmarks, equivalent to GLUE for English. For instance, FLUE, GLUES, and IndicGLUE are the benchmarks for French, Spanish, and 11 major Indian languages. A Chinese version of GLUE is also developed to evaluate performance for Chinese NLP tasks.

Table 14 list the latest monolingual models and their datasets. These models are based on BERT and therefore use the same tasks (MLM and NSP) for pre-training. Most of these languages have also composed their own pre-training and test dataset and GLEU benchmarks. It was also observed that transfer learning from a multilingual to a monolingual model does help with lower-resourced languages and reduces the training time. However, [90] demonstrated that even for a lower-resourced language such as Finnish, model FinBERT trained from scratch outperformed multilingual BERT at a range of tasks, advancing state-of-the-art in many NLP tasks.

The architecture, downstream tasks, pre-training datasets and their size used for different language models are highlighted in Table 14.

VIII. CONCLUSION AND FUTURE WORK

Given the recent advances in computing power and the advent of very large neural networks and pre-trained language models, for example, GPT, Transformers, and BERT, rapid progress have been made in downstream NLP tasks such as text classification, sentiment analysis and machine translation.

A growing interest in developing new language models for NLP tasks has significantly improved the performance and accuracy of word embeddings. Furthermore, the use of different types of word embeddings, such as domain-specific and Knowledge-enriched embeddings, has demonstrated its usefulness in capturing the semantic and syntactic relationships between words in large corpora by utilizing local contextual information and external knowledge sources.

While word embeddings are widely used for generating word representations, they still have limitations such as varying performance on different tasks [108], Bias [117], [118],

interpretability [119], and privacy issues [120]. Future research directions in NLP may likely lead to new and improved techniques for word embeddings, further contributing to the progress in the NLP field.

REFERENCES

[1] J. Hutchins, "The history of machine translation in a nutshell," *Retrieved December*, vol. 20, no. 2009, p. 1, 2005.
[2] G. Gazdar, E. Klein, G. K. Pullum, and I. A. Sag, "Generalized phrase structure grammar," *New Gener. Comput.*, vol. 4, pp. 217–219, Jun. 1986.
[3] R. A. Sharman, F. Jelinek, and R. L. Mercer, "Generating a grammar for statistical training," in *Speech and Natural Language*. Valley, PA, USA, Jun. 1990.
[4] P. R. Asveld, "Fuzzy context-free languages—Part 2: Recognition and parsing algorithms," *Theor. Comput. Sci.*, vol. 347, nos. 1–2, pp. 191–213, 2005.
[5] F. Wang, "A fuzzy grammar and possibility theory-based natural language user interface for spatial queries," *Fuzzy sets Syst.*, vol. 113, no. 1, pp. 147–159, 2000.
[6] R. Patil and Z. Chen, "STRUCT: Incorporating contextual information for English query search on relational databases," in *Proc. 3rd Int. Workshop Keyword Search Structured Data*, 2012, pp. 11–22.
[7] R. Patil, Z. Chen, and Y. Shi, "Database keyword search: A perspective from optimization," in *Proc. Int. Conf. Web Intell. Intell. Agent Technol.*, vol. 3, 2012, pp. 30–33.
[8] R. Patil, S. Boit, and N. Bowman, "SQL ChatBot—Using context free grammar," in *Proc. IEEE Int. IoT, Electron. Mechatronics Conf. (IEMTRONICS)*, Jun. 2012, pp. 1–7.
[9] Z. S. Harris, "Distributional structure," *Word*, vol. 10, nos. 2–3, pp. 146–162, 1954.
[10] R. Patil and A. Shrestha, "Feature-set for sentiment analysis," in *Proc. SoutheastCon*, 2019, pp. 1–5.
[11] C. Hutto and E. Gilbert, "VADER: A parsimonious rule-based model for sentiment analysis of social media text," in *Proc. Int. AAAI Conf. Web Social Media*, vol. 8, no. 1, 2014, pp. 216–225.
[12] F. A. Nielsen, "A new ANEW: Evaluation of a word list for sentiment analysis in microblogs," 2011, *arXiv:1103.2903*.
[13] S. Mohammad and P. Turney, "Emotions evoked by common words and phrases: Using mechanical Turk to create an emotion lexicon," in *Proc. NAACL HLT Workshop Comput. Approaches Anal. Gener. Emotion Text*, 2010, pp. 26–34.
[14] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 168–177.
[15] T. Wilson, J. Wiebe, and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis," in *Proc. Hum. Lang. Technol. Conf. Conf. Empirical Methods Natural Lang. Process.*, pp. 347–354, 2005.
[16] P. J. Stone, D. C. Dunphy, and M. S. Smith, *The General Inquirer: A Computer Approach to Content Analysis*. Cambridge, MA, USA: MIT Press, 1966.
[17] S. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-35, no. 3, pp. 400–401, Mar. 1987.
[18] P. F. Brown, J. Cocke, S. A. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, and P. S. Roossin, "A statistical approach to machine translation," *Comput. Linguistics*, vol. 16, no. 2, pp. 79–85, 1990.
[19] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," *Comput. Speech Lang.*, vol. 13, no. 4, pp. 359–394, Oct. 1999.
[20] G. Salton and M. E. Lesk, "Computer evaluation of indexing and text processing," *J. ACM*, vol. 15, no. 1, pp. 8–36, 1968.
[21] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *J. Document.*, vol. 60, no. 5, pp. 493–502, Oct. 2004.
[22] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, 1975.
[23] L. Havrlant and V. Kreinovich, "A simple probabilistic explanation of term frequency-inverse document frequency (tf-idf) heuristic (and variations motivated by this explanation)," *Int. J. Gen. Syst.*, vol. 46, no. 1, pp. 27–36, Mar. 2017.
[24] C. Boulis and M. Ostendorf, "Text classification by augmenting the bag-of-words representation with redundancy-compensated bigrams," in *Proc. Int. Workshop Feature Selection Data Mining*, 2005, pp. 9–16.
[25] F. R. López, H. Jiménez-Salazar, and D. Pinto, "A competitive term selection method for information retrieval," in *Proc. Int. Conf. Intell. Text Process. Comput. Linguistics*, 2007, pp. 468–475.
[26] D. Hiemstra, "A probabilistic justification for using tf-idf term weighting in information retrieval," *Int. J. Digit. Libraries*, vol. 3, no. 2, pp. 131–139, Aug. 2000.
[27] K. Lund and C. Burgess, "Producing high-dimensional semantic spaces from lexical co-occurrence," *Behav. Res. Methods, Instrum., Comput.*, vol. 28, no. 2, pp. 203–208, Jun. 1996.
[28] C. Burgess, K. Livesay, and K. Lund, "Explorations in context space: Words, sentences, discourse," *Discourse Process.*, vol. 25, nos. 2–3, pp. 211–257, Jan. 1998.
[29] B. Tang, M. Shepherd, E. Milios, and M. I. Heywood, "Comparing and combining dimension reduction techniques for efficient text clustering," in *Proc. SIAM Int. Workshop Feature Selection Data Mining*, 2005, pp. 17–26.
[30] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Proc. 14th Int. Conf. Mach. Learn.*, 1997, pp. 412–420.
[31] J. Kogan, C. Nicholas, and V. Volkovich, "Data mining–text mining with information-theoretic clustering," *Comput. Sci. Eng.*, vol. 5, no. 6, pp. 52–59, Nov. 2003.
[32] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. Amer. Soc. Inf. Sci.*, vol. 41, no. 6, pp. 391–407, 1990.
[33] S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman, "Using latent semantic analysis to improve access to textual information," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst. (CHI)*, 1988, pp. 281–285.
[34] T. Hofmann, "Probabilistic latent semantic indexing," in *Proc. 22nd Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 1999, pp. 50–57.
[35] T. Hofmann, "Unsupervised learning by probabilistic latent semantic analysis," *Mach. Learn.*, vol. 42, no. 1, pp. 177–196, 2001.
[36] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Jan. 2003.
[37] C. Jutten and J. Herault, "Blind separation of sources, Part I: An adaptive algorithm based on neuromimetic architecture," *Signal Process.*, vol. 24, no. 1, pp. 1–10, Jul. 1991.
[38] A. Hyvärinen and E. Oja, "Independent component analysis: Algorithms and applications," *Neural Netw.*, vol. 13, nos. 4–5, pp. 411–430, Jun. 2000.
[39] M. Sahlgren, "An introduction to random indexing," in *Proc. Methods Appl. Semantic Indexing Workshop 7Th Int. Conf. Terminol. Knowl. Eng.*, 2005, pp. 1–9.
[40] E. Gabrilovich and S. Markovitch, "Computing semantic relatedness using Wikipedia-based explicit semantic analysis," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, vol. 7, 2007, pp. 1606–1611.
[41] Y. Li and T. Yang, "Word embedding for understanding natural language: A survey," in *Guide to Big Data Applications*, vol. 26. Cham, Switzerland: Springer, 2018, pp. 83–104.
[42] A. Paccanaro and G. E. Hinton, "Learning distributed representations of concepts using linear relational embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 13, no. 2, pp. 232–244, Mar. 2001.
[43] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 13, 2000, pp. 1–7.
[44] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.
[45] T. Mikolov, I. C. K. Sutskever, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 1–9.
[46] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1188–1196.
[47] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in translation: Contextualized word vectors," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–12.
[48] O. Melamud, J. Goldberger, and I. Dagan, "Context2vec: Learning generic context embedding with bidirectional LSTM," in *Proc. 20th SIGNLL Conf. Comput. Natural Lang. Learn.*, 2016, pp. 51–61.
[49] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, 2018, pp. 2227–2237.

[50] E. Huang, R. Socher, C. Manning, and A. Ng, "Improving word representations via global context and multiple word prototypes," in *Proc. 50th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2012, pp. 873–882.

[51] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.

[52] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 135–146, Dec. 2016.

[53] L. Vilnis and A. McCallum, "Word representations via Gaussian embedding," 2014, *arXiv:1412.6623*.

[54] B. Athiwaratkun and A. G. Wilson, "Multimodal word distributions," 2017, *arXiv:1704.08424*.

[55] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[56] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever. (2018). *Improving Language Understanding By Generative Pre-Training*. [Online]. Available: https://s3-us-west-2.amazonaws.com/openaiassets/researchcovers/languageunsupervised/languageunderstandingpaper.pdf

[57] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," 2018, *arXiv:1801.06146*.

[58] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H. W. Hon, "Unified language model pre-training for natural language understanding and generation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–13.

[59] T. Pires, E. Schlinger, and D. Garrette, "How multilingual is multilingual BERT?" in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 4996–5001.

[60] K. Karthikeyan, Z. Wang, S. Mayhew, and D. Roth, "Cross-lingual ability of multilingual BERT: An empirical study," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–9.

[61] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean, "Google's multilingual neural machine translation system: Enabling zero-shot translation," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 339–351, Oct. 2017.

[62] A. Conneau and G. Lample, "Cross-lingual language model pretraining," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–10.

[63] H. Huang, Y. Liang, N. Duan, M. Gong, L. Shou, D. Jiang, and M. Zhou, "Unicoder: A universal language encoder by pre-training with multiple cross-lingual tasks," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 2485–2494.

[64] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 8440–8451.

[65] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "MASS: Masked sequence to sequence pre-training for language generation," 2019, *arXiv:1905.02450*.

[66] Z. Chi, L. Dong, F. Wei, W. Wang, X. L. Mao, and H. Huang, "Cross-lingual natural language generation via pre-training," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 5, 2020, pp. 7570–7577.

[67] Y. Liu, J. Gu, N. Goyal, X. Li, S. Edunov, M. Ghazvininejad, M. Lewis, and L. Zettlemoyer, "Multilingual denoising pre-training for neural machine translation," *Trans. Assoc. Comput. Linguistics*, vol. 8, pp. 726–742, Dec. 2020.

[68] Y.-S. Chuang, C.-L. Liu, H.-Y. Lee, and L.-S. Lee, "SpeechBERT: An audio-and-text jointly learned language model for end-to-end spoken question answering," 2019, *arXiv:1910.11559*.

[69] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid, "VideoBERT: A joint model for video and language representation learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 7464–7473.

[70] C. Sun, F. Baradel, K. Murphy, and C. Schmid, "Learning video representations using contrastive bidirectional transformer," 2019, *arXiv:1906.05743*.

[71] B. Korbar, F. Petroni, R. Girdhar, and L. Torresani, "Video understanding as machine translation," 2020, *arXiv:2006.07203*.

[72] H. Luo, L. Ji, B. Shi, H. Huang, N. Duan, T. Li, J. Li, T. Bharti, and M. Zhou, "UniVL: A unified video and language pre-training model for multimodal understanding and generation," 2020, *arXiv:2002.06353*.

[73] L. Li, Y.-C. Chen, Y. Cheng, Z. Gan, L. Yu, and J. Liu, "HERO: Hierarchical encoder for video+language omni-representation pre-training," 2020, *arXiv:2005.00200*.

[74] L. Zhu and Y. Yang, "ActBERT: Learning global-local video-text representations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8746–8755.

[75] L. Harold Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang, "VisualBERT: A simple and performant baseline for vision and language," 2019, *arXiv:1908.03557*.

[76] J. Lu, D. Batra, D. Parikh, and S. Lee, "ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–10.

[77] C. Alberti, J. Ling, M. Collins, and D. Reitter, "Fusion of detected objects in text for visual question answering," 2019, *arXiv:1908.05054*.

[78] H. Tan and M. Bansal, "LXMERT: Learning cross-modality encoder representations from transformers," 2019, *arXiv:1908.07490*.

[79] G. Li, N. Duan, Y. Fang, M. Gong, and D. Jiang, "Unicoder-VL: A universal encoder for vision and language by cross-modal pre-training," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 7, 2020, pp. 11336–11344.

[80] L. Zhou, H. Palangi, L. Zhang, H. Hu, J. Corso, and J. Gao, "Unified vision-language pre-training for image captioning and VQA," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 7, 2020, pp. 13041–13049.

[81] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai, "VL-BERT: Pre-training of generic visual-linguistic representations," 2019, *arXiv:1908.08530*.

[82] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, "ERNIE: Enhanced language representation with informative entities," 2019, *arXiv:1905.07129*.

[83] M. E. Peters, M. Neumann, R. L. Logan IV, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith, "Knowledge enhanced contextual word representations," 2019, *arXiv:1909.04164*.

[84] W. Liu, P. Zhou, Z. Zhao, Z. Wang, Q. Ju, H. Deng, and P. Wang, "K-BERT: Enabling language representation with knowledge graph," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 2901–2908.

[85] X. Wang, T. Gao, Z. Zhu, Z. Liu, J. Li, and J. Tang, "KEPLER: A unified model for knowledge embedding and pre-trained language representation," *Trans. Assoc. Comput. Linguistics*, vol. 9, pp. 176–194, Feb. 2020.

[86] P. Ke, H. Ji, S. Liu, X. Zhu, and M. Huang, "SentiLARE: Sentiment-aware language representation learning with linguistic knowledge," 2019, *arXiv:1911.02493*.

[87] D. Kakwani, A. Kunchukuttan, S. Golla, N. C. Gokul, A. Bhattacharyya, M. M. Khapra, and P. Kumar, "IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages," in *Findings of the Association for Computational Linguistics: EMNLP*. 2020, pp. 4948–4961.

[88] L. Martin, B. Müller, P. J. O. Suárez, Y. Dupont, L. Romary, É. V. De La Clergerie, D. Seddah, and B. Sagot, "CamemBERT: A tasty French language model," 2019, *arXiv:1911.03894*.

[89] H. Le, L. Vial, J. Frej, V. Segonne, M. Coavoux, B. Lecouteux, A. Allauzen, B. Crabbé, L. Besacier, and D. Schwab, "FlauBERT: Unsupervised language model pre-training for French," 2019, *arXiv:1912.05372*.

[90] A. Virtanen, J. Kanerva, R. Ilo, J. Luoma, J. Luotolahti, T. Salakoski, F. Ginter, and S. Pyysalo, "Multilingual is not enough: BERT for Finnish," 2019, *arXiv:1912.07076*.

[91] W. de Vries, A. van Cranenburgh, A. Bisazza, T. Caselli, G. van Noord, and M. Nissim, "BERTje: A Dutch BERT model," 2019, *arXiv:1912.09582*.

[92] P. Delobelle, T. Winters, and B. Berendt, "RobBERT: A Dutch RoBERTa-based language model," 2020, *arXiv:2001.06286*.

[93] W. Antoun, F. Baly, and H. Hajj, "AraBERT: Transformer-based model for Arabic language understanding," 2020, *arXiv:2003.00104*.

[94] Y. Sun, S. Wang, Y. Li, S. Feng, X. Chen, H. Zhang, X. Tian, D. Zhu, H. Tian, and H. Wu, "ERNIE: Enhanced representation through knowledge integration," 2019, *arXiv:1904.09223*.

[95] J. Cañete, G. Chaperon, R. Fuentes, J. H. Ho, H. Kang, and J. Pérez, "Spanish pre-trained BERT model and evaluation data," in *Proc. ICLR*, 2020, pp. 1–10.

[96] Y. Kuratov and M. Arkhipov, "Adaptation of deep bidirectional multilingual transformers for Russian language," 2019, *arXiv:1905.07213*.

[97] F. Souza, R. Nogueira, and R. Lotufo, "BERTimbau: Pretrained BERT models for Brazilian Portuguese," in *Proc. 9th Brazilian Conf. Intell. Syst.*, Rio Grande, Brazil: Springer, Oct. 2020, pp. 403–417.

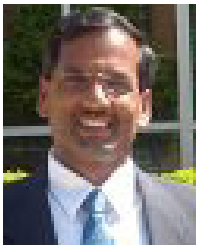
[98] F. Koto, A. Rahimi, J. H. Lau, and T. Baldwin, "IndoLEM and IndoBERT: A benchmark dataset and pre-trained language model for Indonesian NLP," 2020, *arXiv:2011.00677*.

[99] R. Scheible, F. Thomczyk, P. Tippmann, V. Jaravine, and M. Boeker, "GottBERT: A pure German language model," 2020, *arXiv:2012.02110*.

[100] R. Joshi, "L3Cube-mahacorpus and MahaBERT: Marathi monolingual corpus, Marathi BERT language models, and resources," 2022, *arXiv:2202.01159*.
[101] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "BioBERT: A pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, Feb. 2020.
[102] I. Beltagy, K. Lo, and A. Cohan, "SciBERT: A pretrained language model for scientific text," 2019, *arXiv:1903.10676*.
[103] E. Alsentzer, J. R. Murphy, W. Boag, W.-H. Weng, D. Jin, T. Naumann, and M. B. A. McDermott, "Publicly available clinical BERT embeddings," 2019, *arXiv:1904.03323*.
[104] J.-S. Lee and J. Hsiang, "Patent classification by fine-tuning BERT language model," *World Pat. Inf.*, vol. 61, Jun. 2020, Art. no. 101965.
[105] D. Yin, T. Meng, and K.-W. Chang, "SentiBERT: A transferable transformer-based architecture for compositional sentiment semantics," 2020, *arXiv:2005.04114*.
[106] J. Mao, J. Xu, K. Jing, and A. L. Yuille, "Training and evaluating multimodal word embeddings with large-scale web annotated images," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1–9.
[107] Y. Zhang, Q. Chen, Z. Yang, H. Lin, and Z. Lu, "BioWordVec, improving biomedical word embeddings with subword information and MeSH," *Sci. Data*, vol. 6, no. 1, p. 52, May 2019.
[108] B. Chiu, G. Crichton, A. Korhonen, and S. Pyysalo, "How to train good word embeddings for biomedical NLP," in *Proc. 15th Workshop Biomed. Natural Lang. Process.*, 2016, pp. 166–174.
[109] Z. Jiang, Q. Gu, Y. Yin, and D. Chen, "Enriching word embeddings with domain knowledge for readability assessment," in *Proc. 27th Int. Conf. Comput. Linguistics*, 2018, pp. 366–378.
[110] K. Jha, "Knowledge-base enriched word embeddings for biomedical domain," 2021, *arXiv:2103.00479*.
[111] B. Bi, C. Wu, M. Yan, W. Wang, J. Xia, and C. Li, "Incorporating external knowledge into machine reading for generative question answering," 2019, *arXiv:1909.02745*.
[112] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, and D. Amodei, "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901.
[113] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–11.
[114] F. Nooralahzadeh, L. Øvrelid, and J. T. Lønning, "Evaluation of domain-specific word embeddings using knowledge resources," in *Proc. 11th Int. Conf. Lang. Resour. Eval. (LREC)*, 2018, pp. 1–19.
[115] S. A. Aigbe and C. Eick, "Learning domain-specific word embeddings from COVID-19 tweets," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2021, pp. 4307–4312.
[116] T. L. Chen, M. Emerling, G. R. Chaudhari, Y. R. Chillakuru, Y. Seo, T. H. Vu, and J. H. Sohn, "Domain specific word embeddings for natural language processing in radiology," *J. Biomed. Informat.*, vol. 113, Jan. 2021, Art. no. 103665.
[117] A. C. Kozlowski, M. Taddy, and J. A. Evans, "The geometry of culture: Analyzing meaning through word embeddings," 2018, *arXiv:1803.09288*.
[118] T. Bolukbasi, K. W. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai, "Man is to computer programmer as woman is to homemaker? Debiasing word embeddings," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1–9.
[119] L. K. Senel, I. Utlu, V. Yucesoy, A. Koc, and T. Cukur, "Semantic structure and interpretability of word embeddings," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 26, no. 10, pp. 1769–1779, Oct. 2018.
[120] C. Culnane, B. I. P. Rubinstein, and V. Teague, "Health data in an open world," 2017, *arXiv:1712.05627*.
[121] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 1, pp. 5485–5551, 2020.
[122] M. U. Salur and I. Aydin, "A novel hybrid deep learning model for sentiment classification," *IEEE Access*, vol. 8, pp. 58080–58093, 2020.
[123] A. U. Rehman, A. K. Malik, B. Raza, and W. Ali, "A hybrid CNN-LSTM model for improving accuracy of movie reviews sentiment analysis," *Multimedia Tools Appl.*, vol. 78, no. 18, pp. 26597–26613, Sep. 2019.
[124] A. Khare, S. Parthasarathy, and S. Sundaram, "Multi-modal embeddings using multi-task learning for emotion recognition," 2020, *arXiv:2009.05019*.

RAJVARDHAN PATIL received the M.S. degree in computer science and the Ph.D. degree in information technology from the University of Nebraska Omaha, in 2012 and 2016, respectively. He has two years of industrial experience as a Data Engineer. Since 2020, he has been an Assistant Professor of CIS with Grand Valley State University, Allendale, MI, USA. His research interests include natural language processing, transfer learning, image processing, and applied AI.

SORIO BOIT received the Ph.D. degree in information systems from Dakota State University. He is currently an Assistant Professor with the Department of Computing, Grand Valley State University, Allendale, MI, USA. His research interests include transfer learning, deep learning, applied AI applications, NLP, and computer vision.



VENKAT GUDIVADA received the Ph.D. degree in computer science from the University of Louisiana at Lafayette, Lafayette, LA, USA. He is currently the Chairperson and a Professor with the Department Computer Science, East Carolina University. Prior to this, he was the Founding Chair and a Professor with the Department of Computer Science and Electrical Engineering (formerly Weisberg Division of Computer Science), Marshall University. He was also the Vice President of Wall Street companies in New York City for over six years, including the Bank of America Merrill Lynch and GoldenSource (formerly Financial Technologies International). His current research interests include data management, information retrieval, applied machine learning, and personalized learning.

JAGADEESH NANDIGAM received the Ph.D. degree in computer science from the University of Louisiana at Lafayette, in 1995. He is currently a Professor with the School of Computing, Grand Valley State University, Allendale, MI, USA. His research interests include software engineering, programming languages, software security, and natural language processing.