ISSN 2007-9737

Word Embeddings: A Comprehensive Survey

Alexandr Pak[1], Atabay Ziyaden[1], Timur Saparov[1], Iskander Akhmetov[2], Alexander Gelbukh[3,*]

[1] Institute of Informational and Computational Technologies, Big Data Mining Lab, Almaty,, Kazakhstan

[2] Kazakh-British Technical University, Almaty, Kazakhstan

[3] Instituto Politécnico Nacional, Centro de Investigación en Computación, Mexico City, Mexico

aa.pak83@gmail.com, i.akhmetov@kbtu.kz, gelbukh@cic.ipn.mx

Abstract. This article is a systematic review of available studies in the area of word embeddings with an emphasis on classical matrix factorization techniques and contemporary neural word embedding algorithms such as Word2Vec, GloVe, and Bert. The efficiency and effectiveness of these methods for mapping semantic and lexical relationships are evaluated in greater detail providing analysis of the topology of these techniques. In addition, this approach demonstrates a model accuracy of 77%, which is 3% below the best human performance. At the same time the study has also shown the weaknesses of some models such as BERT, which lead to unrealistic high accuracy due to spurious correlations in the datasets. We see that there are three bottlenecks for the subsequent development of NLP algorithms: assimilation of inductive bias, common sense embedding, and generalization problem. The outcomes from this research help in enhancing the strength and applicability of word embeddings in natural language processing tasks.

Keywords. Language models, distributive semantics, word embeddings, natural language processing, deep learning.

1 Introduction

The idea of dealing with words the same way we deal with numbers comes from ancient times, with the attempts of medieval Judaic, Islamic, and Greek scholars to discover secret knowledge from Holy scriptures such as the Torah, Quran, and Gospel. The method that was used back then is called the gematria calculation [18], which assigns a numeric value to each letter of the Hebrew alphabet.

This method was used to analyze the Torah, where words with the same numerical values could be interpreted as being related in meaning. This echoes the modern concept of embeddings, in which similar words are “close” to each other in vector spaces. There is an analogous methods for the Arabic (abjadiya) which is also associated letters of the alphabet with numerical values.

In Islamic culture, this system was used for mystical interpretations of the Quran. Like modern embedding models, abjad allowed symbols to be transformed into numerical sequences while preserving semantic relationships. Moreover, Jifr was a mystical method in the Islamic world that was used to create codes and divination based on the numerical values of letters.

This numerical manipulation of texts can also be interpreted as an ancient form of mathematical analysis of language. Next is an isopsephy which in ancient Greek culture used the numerical values of Greek letters to find hidden meanings in texts.

Pythagoras, a prominent philosopher and mathematician, posited that numbers form the foundation of all things, including language. His ideas about the numerical nature of words can be seen as a precursor to modern embeddings.

Thus, words were turned into a sequence of numbers, which then were manipulated arithmetically. Interestingly, such methods allowed for preserving semantics and logical constructs linking the words, so we can consider these early research as ancient word embeddings.

This historical perspective closely aligns with the development of the distributional hypothesis in natural language processing. The core idea of this hypothesis states that “the meaning of a word can be known from the company it keeps” [13]. It has provided the pathway for many modern algorithms.

Over the decades, the concept of distributional semantics has inspired the development of various methods for representing words numerically. The initial methods included simpler approaches like Bag-of-Words and TF-IDF (Term Frequency-Inverse Document Frequency).

These approaches were based on word frequency, but they often overlooked contextual information, which result in a loss of semantic depth. Word representation experienced a revolution with the introduction of techniques like Word2Vec [24] and GloVe [27].

These approaches provided more complex distributed representations that successfully maintained semantic links in high-dimensional environments. Word2Vec uses architectures that are trained to predict a word from its context or vice versa, such as Continuous Bag of Words (CBOW) and Skip-Gram.

Words with similar semantic content could be represented as dense vectors (embeddings) in the vector space thanks to these models. Representing words as vectors in high-dimensional space has proven to be one of the most important parts of natural language processing (NLP).

They enable various machine learning models working with numbers to parallelize the processing loop, preserving some semantic and syntactic information. Nowadays, natural language processing (NLP) has gone through a significant transformation.

It initially started from the elaboration of task-specific representations and architectures. NLP advanced towards the adoption of task-agnostic models and pre-training methods.

This upgrade has led to major attainments in various challenging NLP tasks, like reading comprehension, question answering, and logical inference. A substantial growth in this field is the rise of Large Language Models (LLMs).

Famous and convenient GPT-4 is the illustration of LLMs, which build on the foundational work of models like GPT-3. These LLMs leverage larger datasets and more complex architectures, which provides enhanced performance across a wide range of tasks [7].

Moreover, there is growing interest in multimodal embeddings. These embeddings combine text with other modalities, like images and video. The features of these embeddings enable models to better understand context and perform tasks which require a comprehensive understanding of both textual and visual information.

For instance, using context-specific prompts has been shown to improve performance in various datasets, such as specifying the type of image or object being described. As a result, it expands their real-world applicability [30].

The paper is organized as follows. In section 2, we describe the issue of word vectorization as the search for the context-dependent optimum aggregate function.

In section 3, the fundamental concepts behind context-independent techniques like as LDA, Word2vec, Glove, and FastText are presented. In section 4, we describe context-dependent techniques for both short and long context embeddings.

Short-context embeddings include models such as ELMo, BERT, and ELECTRA, and long-context embeddings include variants of several transformer designs (Longformer, Linformer, Reformer, Sparse Transformer, BigBird, and Synthesizer) or methods such as GPT-3.

In section 5, we explain the data used to run several models. In the next part, we present an experiment undertaken to compare the effectiveness of all embedding approaches. In section 7, we then discuss the findings.

2 Problem Statement

There is a text corpus C represented as the sequence S of tokens $(t_1, t_2, ..., t_N)$. Each token of the sequence S corresponds to dense feature vector $\mathbf{h}_{t_i}$ [25]. In an old fashioned manner, the main structure that preserves semantic features is a matrix of words co-occurrences $\mathbf{W} \in R^{|V|\times|V|}$, where $|V|$ is the dictionary size. This structure can be obtained directly from the sequence S or from document-term matrix $\mathbf{D} \in R^{|C|\times|V|}$, where $|C|$ is the number of texts in the corpus C.

In real-world problems, the values of the corpus size $|C|$ or the dictionary size $|V|$ are great enough to produce an extra complexity during the calculation. It is convenient to reduce the dimensions of the matrices $\mathbf{W}$ in such a manner that there is a new embeddings matrix $\mathbf{E} \in R^{|V|\times d}, d \ll |V|$, where d is the dimension size of word embeddings. In particular, each row $\mathbf{e}_i$ of $\mathbf{E}$ relates to the semantic representation of word i in the dictionary V. There are the models that developed this approach with local context Word2vec [24] and Glove with global context [27].

Then again, techniques that learn contextual embeddings maps every token t_i to vector representation $\mathbf{h}_{ti}$ that takes into account every element of the current input sequence $S(j)$, where j is the position in the original sequence S, limited with window width w. Strictly speaking, the embedding vector of contextual representation can be written as $\mathbf{h}_{ti} = f(\mathbf{e}_{t1}, \mathbf{e}_{t2}, ..., \mathbf{e}_{tN})$. The vector $\mathbf{h}_{ti}$ can be considered as the part of optimization problem of Language Model (LM), that formulated as follows:

$$p\left(t_1, t_2, \ldots, t_N\right) = \prod_{i=1}^{N} p\left(t_i | t_1, t_2, \ldots, t_{i-1}\right),$$

where $p(t_i|t_{i-1})$ is conditional probability density function. The language model is the text generative method, which is the maximum likelihood of a sequence of multiple n words. These context-dependent representations are best suited for capturing semantics at sequence level than algorithms of non-contextual terms. For f, there are multiple model architectures, which we are considering here.

3 Context-independent Embeddings

3.1 Latent Dirichlet Allocation (LDA)

Blei et al. [5] propose a technique that models topics of documents. The approach assumes that each document contains a mixture of topics characterized by the word usage statistics and uses LDA to identify sets of words for each topic and calculates probabilities of attribution to topics for each document. LDA is based on the probabilistic model shown in (1):

$$p(d, w) = \sum_{t \in T} p(d) p(w \mid t) p(t \mid d), \tag{1}$$

where T is the set of topics, $p(d, w)$ is the probability of appearance of a document-word pair, $p(d)$ is the prior distribution on the documents set, $p(w|t)$ is the conditional probability of a word w given a topic t, $p(t|d)$ is the conditional probability of a topic t given a document d. With the additional assumptions:

– Document vectors $\theta_d = (p(t|d) : t \in T)$ are derived by the same probabilistic distribution on normalized vectors of size $|T|$; this distribution can be taken from parametric distribution family of Dirichle: $\mathrm{Dir}(\theta, \alpha), \alpha \in R^{|T|}$.

– Topic vectors $\phi_t = (p(w|t) : w \in W)$ are derived by the same probabilistic distribution on normalized vectors of size $|W|$; this distribution can be taken from parametric distribution family of Dirichle: $\mathrm{Dir}(\phi, \beta), \beta \in R^{|W|}$.

LDA parameters are typically determined using methods like Gibbs sampling, variational Bayesian inference, or Expectation-Propagation.

The main problem is to correctly choose the number of topics we seek, which can be solved by running the algorithm on several topic numbers and calculating the quality metric for each version.

The quality of each topic number variant is measured by the topic words coherence [33] as their semantic closeness, which is the euclidean distance between words in a vector space.

3.2 Neural Network Language Model

In [4], authors proposed the model named Neural Network Language Model (NNLM), which produces vector representations of a word during the optimization of a neural network, which interpolates a prediction function of the next word. Like the conventional language model, at NNLM, there is a context window with the width $N-1$ of previous words.

However, there is a difference between LM and NNLM, which consists in finding such function that approximates LM with fewer parameters rather than accurately predicting with the help of words counting. In other words, the main idea of NNLM is replacing large multi-dimensional tensors (e.g., statistical language models) by bounded and low-dimension representations that calculated with the help of distributed word interrelations across all texts of large corpus.

Another advantage of NNLM is the replacement of the computational difficulty from the processing stage to preprocessing. Of course, the cost of this additional step requires more computations, but the hardware requirements scale linearly, not exponentially, with the number of conditioning variables, how it was in statistical LM. All models presented in this section exploits this idea.

3.3 Latent Semantic Analysis (LSA)

Probabilistic latent semantic analysis (PLSA) is a statistical technique for the analysis of two-mode and co-occurrence data proposed by Hofmann and Thomas [17]. It has applications in information retrieval and filtering, topic modeling. Due to latent correlations between terms and topics, one can use PLSA for embedding generation.

The main mechanics behind the scene is a Singular Value Decomposition of co-occurrence tables, also there is a mixture decomposition derived from a latent class model.

In order to avoid overfitting, the method uses a widely applicable generalization of maximum likelihood model fitting by tempered EM. Let a set of documents $D=\{d_i\}_{i=1\dots n}$ be given, where each document is a sequence of tokens $d_i=(w_1,\dots,w_{n_i})$ from the dictionary W. Formally, the task is formulated by (2):

$$\begin{aligned} p(d,w) &= \sum_{t\in T} p(t)p(w\mid t)p(d\mid t), \\ &= \sum_{t\in T} p(d)p(w\mid t)p(t\mid d), \\ &= \sum_{t\in T} p(w)p(t\mid w)p(d\mid t), \end{aligned} \tag{2}$$

where T is a set of topics, $p(t)$ is the unknown prior distribution of topics in the entire collection; $p(d)$ is the prior distribution on the set of documents; and $p(w)$ is the prior distribution on the word set. It is worth noting that the approximation of $p(w\mid t)$ gives the required word embedding with preserved semantic properties. In other words, co-occurrence matrix word-topic has semantic knowledge about word meanings. The main shortcomings of PLSA are:

1. The number of parameters grows linearly with the number of documents in the collection, which can lead to model over-fitting;
2. When adding a new document d to the collection, the distribution $p(t\mid d)$ cannot be calculated using the same formulas as for other documents without rebuilding the entire model.

3.4 Word2Vec

It is important to note that relationships can be approximated in both directions, from the central word to its context (skip-gram) and from the context to its central word (CBOW, Continuous Bag of Words). In the last decade, the first of the most popular algorithms is word2vec [25]. The central idea of Word2Vec is to utilize cosine distance properties between word vectors, transforming the point cloud so that words with similar meanings are close in vector space. And vice-versa words with different meanings should be far enough, in terms of cosine distance. Formally, the model of word2vec in terms of skip-gram is following:

$$\sum_{j=1}^{T} \sum_{-h\le k\le h, k\ne 0} \ln \frac{\vec{w}_{j+k}^{T}\vec{w}_j}{\sum_{i=1}^{|V|}\exp\left(\vec{w}_i^{T}\vec{w}_j\right)}, \tag{3}$$

where T - the amount of tokens in text corpus, j - current token, h - the size of context window, k - the index of the context word of current token, $|V|$ - the size of word dictionary.

3.5 Global Vectors for Word Representation

A variation of the metric approach to word embeddings is GloVe, or Global Vectors for Word Representation. In contrast to the previously mentioned algorithm it uses the approximation of the global context with the help of co-occurrences matrix [27]. The definition of GLoVE model is following:

$$\sum_{i,j=0}^{|V|} f\left(X_{ij}\right)\left(\vec{w}_i^T \vec{w}_j + b_i + b_j - \log\left(X_{ij}\right)\right), \qquad (4)$$

where X_{ij} is a frequency value of the co-occurrence matrix; f is a normalization function. Both of the algorithms mentioned above have one common drawback. Namely, if there is a case during the production stage, that a high-level system tries to process words that don't belong to its vocabulary, then all pipeline models of the system have to be trained once again considering new words. The FastText algorithm solves the problem outside the vocabulary word (Out-of-Vocabulary, OOV) by taking into account the alphabetic bigrams and trigrams that make up the encoded-word in the resulting vector [6].

It was shown that the algorithms of this approach have a drawback, namely, instability of reproducibility of the results [35], due to the random initialization of weights, the random order of training examples. When expanding the size of the dictionary, the entire correlation structure of the word cloud changes, resulting in a machine model, the training below in the pipeline processing also require retraining.

3.6 FastText

The models of Word2Vec with negative sampling and FastText looks pretty the same, but unlike Word2Vec, Fasttext considers words as a combination of N-grams. For the word “text,” it could be [“te,”, “tex,” “ext”, “xt”], depending on the parameters of the maximum and minimum n-gram. The word-vector is composed as the sum of the vectors of all the n-grams of this word. The objective function of the FastText model can be written as follows:

$$\sum_{j=0}^{T}\left[\sum_{c\in C_j}\log\left(1+\exp\left(-s(w_j,w_c)\right)\right) + \sum_{n\in N_j}\log\left(1+\exp\left(s(w_j,w_n)\right)\right)\right], \qquad (5)$$

where w_j is the central word, C_j is the set of words in the context of the central word, N_j is the set of negative sampling words. The proximity function can be defined as follows (4):

$$s(w,c) = \sum_{g\in G_w} \vec{z}_g^T \vec{w}_c. \qquad (6)$$

As a result, this method allows you to get better representations for words, not only based on the co-occurrence of words, but also the co-occurrence of their syllables and n-grams. Notably greatly improved are vectors for rare words, but at the same time, their parts are found in other, more frequent words.

Another significant advantage is the solution to the problem of out-of-vocabulary words due to the consideration of subword information. In other words, FastText allows us to work with words that were not available in the training set since the vectors of these words can be composed of the sum of its n-grams. On the other hand, we get increased learning time and more memory requirements [6, 22, 21].

4 Context-dependent Embeddings

Pre-trained bidirectional language models (biLMs) form the backbone of contextualized word embeddings [29]. Originally, they supposed to solve two issues:

1. Syntax and semantics characteristics of word.

2. The changing of these characteristics across variety of linguistic contexts (e.g., homonymy and polysemy). Subword information and contextual.

They have been used to improve performance for many NLP tasks. Formally bidirectional language model defined as the probability of next token with given set of previous tokens:

$$p\left(w_1, w_2, \ldots, w_N\right) = \prod_{k=1}^{N} p\left(w_k | w_{k+1}, w_{k+2}, \ldots, t_N\right). \tag{7}$$

The embedding models based on the such LM take into account only the past information. The need to incorporate the information of future context is useful for the improvement of model performance. Hence, the design variation of BLM consists in consideration the context of current token in both directions:

$$\sum_{k=1}^{N} \Big(\log p(w_k \mid w_1, w_2, \ldots, w_{k-1}; \overrightarrow{\theta}) + \log p(w_k \mid w_{k+1}, w_{k+2}, \ldots, w_N; \overleftarrow{\theta})\Big), \tag{8}$$

where $\overrightarrow{\theta}$ and $\overleftarrow{\theta}$ are the forward and backward parameters of LM, respectively. For instance, the task of translation between Russian and Kazakh languages requires to see the relations of words inside the text. These relations are syntax structure, coreference links.

The text summarization task also exploits the context features to determine key-words and most relevant sentences. Another industry application of contextual embeddings algorithms is semantic sentiment analysis if there are ironic or sarcastic statements, then the model can detect them only with the help of context features.

The question answering systems exploit the idea that the lexical distribution of the question correlates with the lexical distribution of the answer. In other words, these systems need to understand the contexts of answers and questions to find the correct answer. In this paper there are considered contextual embedding models based on the variety of LM, namely Bi-directional Language Model (BLM), Masked Language Model (MLM), Permutation Language Model (PML) and extension of MLM with ideas of Generative Adversarial Network(GAN). All these models extend the basic model LM in an original way to achieve specific properties about which we explain further in the paper. Based on the length of the context and the data processed by the described methods, embeddings may be split into two types: long and short-context embeddings.

The methodology to processing such information differs across various techniques. When processing a lengthy context, the algorithm's complexity, processing time, and amount of resources utilized are frequently problematic. In order to circumvent this issue, specialized algorithms intended to operate in such circumstances have been developed.

4.1 Short-context Embeddings

4.1.1 Embeddings from Language Models (ELMo)

The popular recurrent neural network architectures, such as the Long Short Term Memory (LSTM) [16], the Generalized Recurrent Unit (GRU) [9], have opened a new stage in the development of Natural Language Processing (NLP), namely their combination in the form architecture Encoder-Decoder, please see Fig. 1.

In paper [28], authors obtained context-sensitive vector representations of words, which partially solved the problems of polysemy coding and homonymy. The key advantages that have allowed the development of NLP systems to a new level are the inclusion of word order in the text and bidirectional language models:

$$p\left(w_1, w_2, \ldots, w_N\right) = \prod_{k=1}^{N} p\left(w_k \mid w_1, w_2, \ldots, w_{k-1}\right). \tag{9}$$

$$p\left(w_1, w_2, \ldots, w_N\right) = \prod_{k=1}^{N} p\left(w_k \mid w_{k+1}, w_{k+2}, \ldots, t_N\right). \tag{10}$$

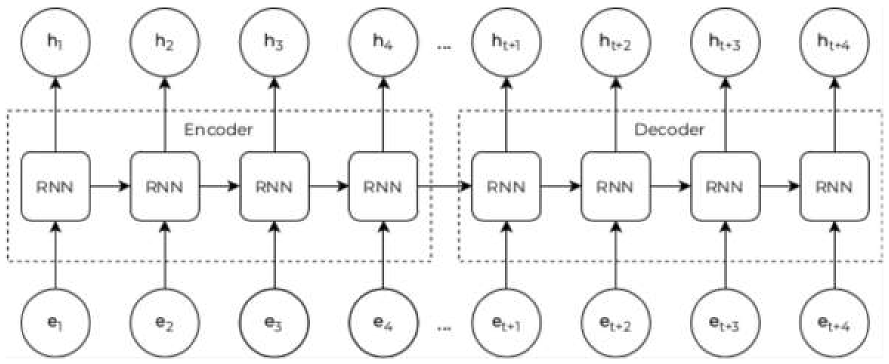

Fig. 1. The neural network architecture of RNN encoder-decoder

$$\begin{aligned} R_k &= \left\{ x_k^{LM}, \vec{h}_{k,j}^{LM}, \vec{h}_{k,j} \mid j = 1, 2, \ldots, L \right\} \\ &= \left\{ h_{k,j}^{LM} \mid j = 0, \ldots, L \right\}. \end{aligned} \quad (11)$$

Recursive architectures carried additional challenges such as slow learning speed, explosion, and fading gradients. Models based on the Transformer [37] neural network architecture, such as BERT [11] and XLNet [40], solved the above problems and are the last word in the field of word vectoring.

4.1.2 BERT

The architecture of Bidirectional Encoder Representations from Transformers (BERT) exploits the idea Encoder-Decoder approach. The Transformer model was proposed by Vaswani et al. [37]. The basic element of the Transformer model is a Dot-Product Attention unit, which is included in a Multi-Headed Attention unit.

In return, each layer in the Encoder and Decoder consists of Multi-Headed Attention units, they compute vector representations between a given token and all other tokens in the sequence S, after that position-wise feed-forward network calculate the output. BERT is an acronym for Bidirectional Encoder Representations from Transformers.

Input data is corrupter by replacing certain words with unique tokens like “[MASK]” for a training model to reconstruct the original sentence. Like the GPT, the BERT architecture is based on the Transformer model. Basically it’s a multi-layer bidirectional Transformer encoder [11]. BERT base consists of 12 layers with 12 self-attention heads each and hidden size=768. BERT large consists of 24 layers with 16 self-attention heads each and hidden size=1024.

Both models are pre-trained using two unsupervised tasks. The first training task is Masked LM. 15 percents of words in each sentence are replaced by “[MASK]” token. The second task on which this model was trained is NSP or Next Sentence Prediction. In the training process model learns to predict second sentence in the pair of sentences given as input.

Half of input pairs are subsequent sentences, and other half is random sentences from a corpus paired together. In the beginning of a sentence “[CLS]” token is inserted and “[SEP]” token at the end for model be able to distinguish between sentences.

Due to the fact that Bert learned not only to predict the next word in a sentence, but learned to predict a masked word on the basis of the previous and next words on it, he got high results and pushed GPT from the GLUE dataset leaderboard.

4.1.3 XLNet

The algorithm allows you to simulate bidirectional contexts by maximizing the expected probability for all permutations up to the factorization order and overcomes the limitations of BERT due to its autoregressive architecture. Also, XLNet brings together ideas from Transformer-XL, the most advanced autoregressive model.

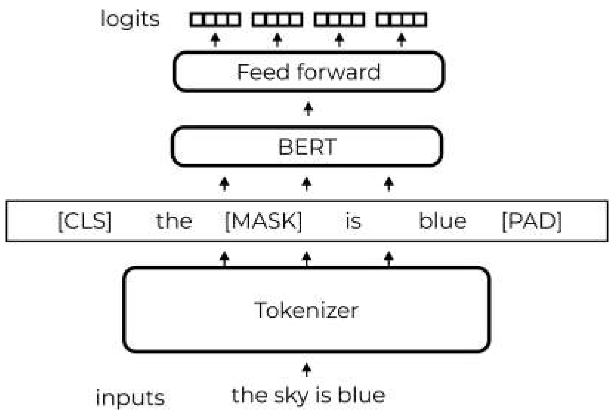

Fig. 2. The scheme of pre-training and fine-tuning BERT. The figure from [11]

Experimentally, XLNet surpasses BERT in 20 tasks, often by a wide margin, and achieves the most excellent results in 18 tasks, including answering questions, output in natural language, analysis of tonality, and ranking of documents. The original idea that distinguishes XLNet from BERT is the Permutation Language Model (PLM).

For a sequence x of length T, there is $T!$ various options for autoregressive factorization. It is intuitively clear that if the model parameters are common to all factorization options, then the model will learn to take into account bi-directional context information. Formally, let Z_T be the set of all possible permutations of length T, then index the sequence $[1,\ 2,\ \ldots,\ T]$, the PLM model can be written in the following form:

$$\max_{\theta} E_{z\sim Z_T}\left[\sum_{t=1}^{T}\log p_{\theta}\left(x_{z_t}\mid x_{z<t}\right)\right], \qquad (12)$$

where z_t is the t-element of the sequence, and $z_{t<}$ is the $t-1$ of the first elements of the sequence.

Essentially, for a text sequence x, we select the factorization order z at a time and expand the probability $p_\Theta(x)$ by the factorization order. Since all factorization orders share the same model parameter Θ during training, each possible element $x_i \neq x_t$ in the sequence is taken into account in the calculation of x_t; therefore, it can cover a bidirectional context.

Moreover, since this objective function corresponds to the approach of autoregressive models, it naturally avoids the assumption of independence and inconsistency between the pre-workout and fine-tuning settings that were mentioned earlier [40].

The proposed objective function rearranges only the factorization order, not the sequence order. In other words, the original word order preserved with positional coding corresponding to the original sequence, and the corresponding attention mask in Transformers are used to obtain a permutation of the factorization order.

Please note that this choice is necessary, as the model only encounters text sequences with a natural order during training. Let us demonstrate an example of the prediction of the token x_3 for the same input sequence x, but with different factorization orders in Figure 3.

4.1.4 ELECTRA

There is another original idea in contrast to the previous models that are based on masking and permutations. [10] proposed a new pre-training approach called "Replaced token detection." Instead of masking, they replace some tokens by synthetically generated and then learns a model to distinguish real tokens from generated replacements.

The main idea of the approach is to train two neural networks the first one is a generator and second one is discriminator, each of them consisting of a Transformer network that maps input tokens X into contextualized vector representations h. The first one is a generator G; second is discriminator D. Probability for generating a token x_t at given position t:

$$p_G(x_t|x) = \frac{\exp(e(x_t)^T h_G(x)_t)}{\sum_{x'}\exp(e(x')^T h_G(x)_t)}, \qquad (13)$$

where **e** stands for token embeddings. And prediction of discriminator whether token x_t is real:

$$D(x,t) = \text{sigmoid}\left(w^T h_D(x)_t\right). \qquad (14)$$

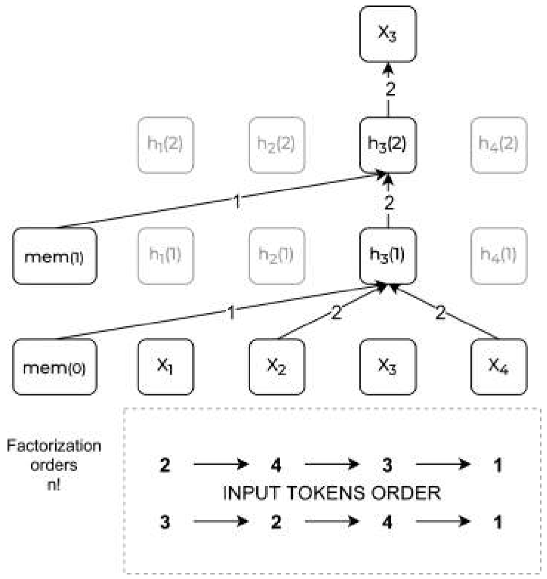

Fig. 3. Illustration of the PLM model, several permutation steps for predicting x_3 for the same input sequence x, but with a different factorization order. Figure taken from [40]

4.2 Long-context Embeddings

Since the introduction of Transformer architecture [37], there was a rapid development of transformer-based models. BERT being the most successful one [11]. However, the classical transformer is not without flaws, and there were many attempts to refine and improve it over the years.

Contemporary researches state the problem of Transformer optimization in terms of time and memory expenses. The fact of scientists and engineers have to train large models in long periods of time with the help of costly computational devices, e.g., sizes up to 64 layers with a width of 500 million parameters in each layer.

This resource limitation is especially evident in tasks with longer sequences. So the main research question is "does the Transformer really need such a volume of resources, or is it the inefficiency of the model?" Briefly, the main bottleneck of Transformer algorithm is contextual mapping matrix $P = Q \times K^T$, that computes as a part of dot-product-attention function:

$$\text{Att}(Q, K, V) = \text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_{\text{hidden}}}}\right) V, \quad (15)$$

where $\frac{1}{\sqrt{d_{\text{hidden}}}}$ is scaling factor; $Q \in R^{n \times d_{\text{hidden}}}$ - query matrix; $K \in R^{n \times d_{\text{hidden}}}$ - key matrix; $V \in R^{n \times d_{\text{output}}}$ - value matrix; n is a sequence length; d_{hidden} is a size of internal states, d_{output} is a size of resulting state. So, the P matrix has quadratic complexity against the input sequence length. So, at this section there is a discussion about various optimization approaches of Transformer.

4.2.1 Reformer

RevNets proposed by Gomez et al. [14], the idea of which is that the activation of each next layer can be obtained from the activation of the previous one, using only the model parameters. This allows not to save the activation of each layer, i.e. allows you to get rid of N in the complexity of the algorithm.

In their model Reformer, Nikita Kitaev, Łukasz Kaiser and Anselm Levskaya [23] proposed another way to complexity reduction for transformers. Reversible layers allow you to store only one copy of the activation for the entire model instead of N times.

Handling activation in chunks allows you to reduce memory in fully connected layers. An approximate calculation of the attention mechanism based on locality-sensitive hashing reduces the complexity from O (L^2) in the corresponding layers to O $(L \log L)$ and also allows you to work with long sequences.

Locally Sensetive Hashing (LSH). Main constraint of the attention mechanism is the dot product of QK^T, also known as a context mapping matrix, which is dependent on the length of the input sequence in a quadratic fashion.

A context mapping matrix may also be calculated using vector-matrix dot-product instead of matrix-matrix dot-product. This is less effective, but it reduces memory requirements by a factor of length. Deduced from Q and K, the LSH attention computation starts with two tensors $Q = K$ and V.

Considering that just $\text{softmax}(QK^T)$ is of importance, not QK^T. That softmax is impacted by the biggest elements, but only in the keys nearest to q_i. Only the 32 or 64 nearest keys are evaluated.

To swiftly locate neighbors, local-sensitive hashing is utilized, in which nearby vectors have

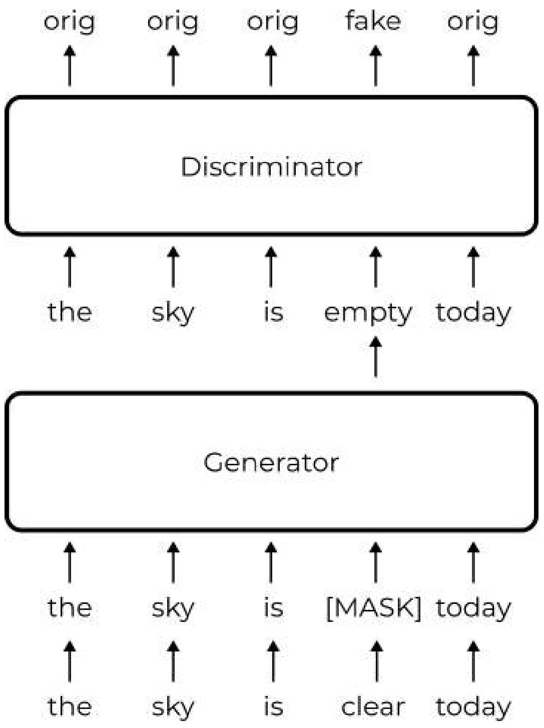

Fig. 4. The neural network architecture of ELECTRA

a high possibility of having similar hashes. P_i is the set to which the query at position i pertains, and z is the normalizing term in the $\mathrm{softmax}$. They skip scaling by $\sqrt{d_k}$ as well:

$$o_i = \sum_{j \in P_i} \exp(q_i \cdot k_j - m(j, P_i) - z(i, P_i))v_j, \quad (16)$$

where $m(j,\ P_i) = \begin{cases} \infty & \text{if } j \notin Pi, \\ 0 & \text{otherwise.} \end{cases}$

4.2.2 Sparse Transformer

Sparse Transformer was created [8] to tackle the quadratic memory growth that can be attributed to the Transformers' design. The model's factorized attention mechanism takes advantage of an algorithmic innovation to extract patterns from twenty to thirty times longer sequences than was previously possible.

When modeling the density of lengthy sequences, the model produced performance comparable to or superior to that of conventional Transformers, although requiring a large reduction in the number of operations.

Sparse Transformer also demonstrated the exploitation of long-term context and the generation of globally coherent samples. On the basis of some ideas described above, OpenAI created their new model called GPT-1.

4.2.3 OpenAI GPT

In 2020, OpenAI, a group devoted to "discovering and implementing the route to safe artificial intelligence," announced the launch of GPT-3, the most advanced natural language processing technology (Generative Pre-trained Transformer). It is characterized as a superintelligent system that learns and adapts from the immense sea of digital text to independently produce fresh, intelligent, and creative material.

It has been hailed as a stunning artificial intelligence text generator capable of imitating human writing with exceptional fluency. GPT models have revolutionized the landscape of Natural language processing (NLP) with their potent capacities to accomplish diverse NLP tasks.

The outcomes include faster reaction times and increased precision. These language models need extremely few or even no examples to comprehend the job and execute with even more accuracy and inventiveness than models that are highly trained on a vast number of examples.

OpenAI's GPT-3 is the third in a series of natural language processing (NLP) tools. Before its release, the model underwent years of research and development to achieve the current stage of innovation in the area of AI text production.

GPT-1. was launched in 2018 by OpenAI. Trained on an enormous BooksCorpus dataset, this generative language model was able to learn large range dependencies and acquire vast knowledge on a diverse corpus of contiguous text and long stretches [31].

In terms of its architecture GPT-1 applies the 12-layer decoder of the transformer architecture with a self-attention mechanism for training.

As a result of its pre-training, one of the significant achievements of GPT-1 was its ability

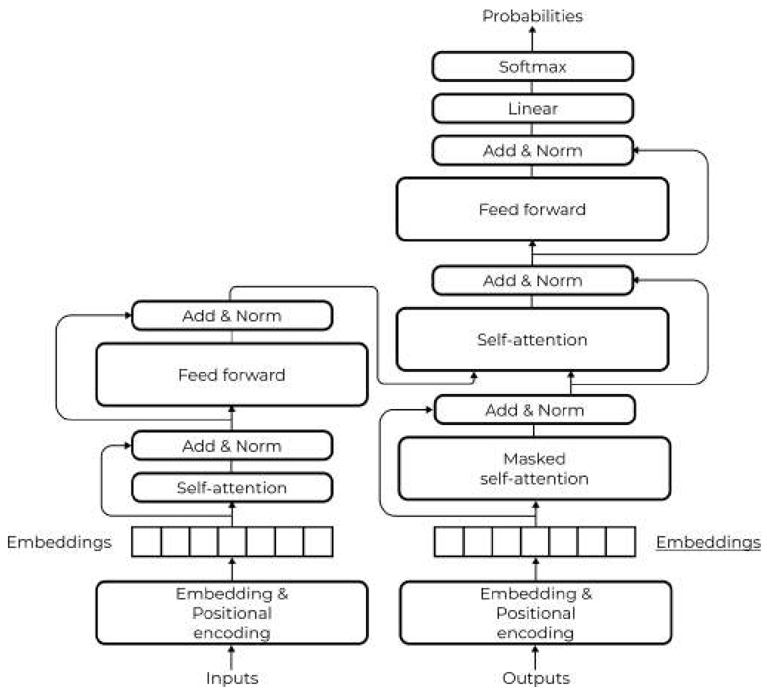

Fig. 5. The neural network architecture of transformer encoder-decoder

to carry out zero-shot performance on various tasks. This ability proved that generative language modeling can be exploited with an effective pretraining concept to generalize the model.

With Transfer learning as its base GPT became a powerful facilitator to perform natural language processing tasks with very little fine-tuning. It generated pathways for other models which could further enhance its potential in generative pre-training with larger datasets and parameters.

GPT-2. Later in 2019, OpenAI created a Generative pre-trained Transformer 2 (GPT-2) [32] by using a bigger dataset and including extra parameters to create a more robust language model. Similar to GPT-1, GPT-2 utilizes the transformer model's decoder.

With 1.5 billion parameters, GPT-2 is 10 times bigger than GPT-1 (117 million parameters), and it contains 10 times as many parameters and 10 times as much data.

It is trained on a varied dataset, making it effective at handling numerous language problems such as translation, summarization, etc., utilizing just raw text as input and little or no training samples. GPT-2 beat its predecessors by greatly boosting the accuracy of recognizing long-range relationships and predicting words across many downstream datasets.

GPT-3. GPT-3 is an earlier open version of the Generative Pre-training Model, paving the way for the more advanced closed GPT-4 and GPT-4o [39]. OpenAI has created a huge language prediction and generation model capable of creating extended sequences of the original text.

GPT-3 became the groundbreaking AI language software for OpenAI. Simply put, un some tasks, such as text generation, abstractive summarization, etc, it came close to human-level performance.

It includes around 175 billion parameters and is 100 times bigger than the GPT-2 database. It is trained using a 500-billion-word data set (referred to as “Common Crawl”) obtained from the large internet and content repository.

It can also do basic arithmetic problems, including producing code snippets and executing

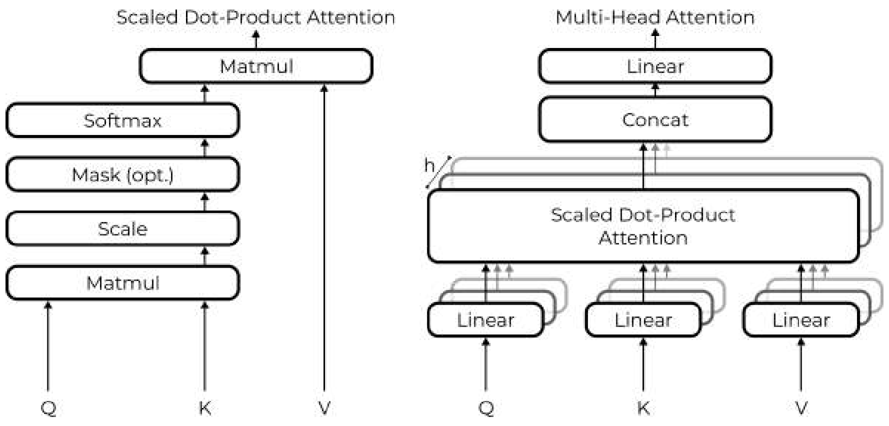

Fig. 6. On the left side there is a neural network architecture of dot-product attention. On the right side there is multi-head dot product attention

intelligent activities, which is a remarkable and unique skill. The outcomes include a quicker reaction time and higher degree of precision, enabling NLP models to aid businesses by successfully and continuously maintaining best practices and decreasing human mistakes.

Due to its complexity and massive scale, several academics and developers have referred to it as the ultimate black box approach to artificial intelligence.

This makes it very costly and unpleasant to execute inference, and its billion-parameter size makes it resource-intensive and difficult to apply to practical applications in its present form. GPT-3 was designed to make language processing more potent and quicker than its predecessors, without requiring any further tweaking.

The majority of earlier language processing models (such as BERT) need extensive fine-tuning using thousands of examples to educate the model how to execute downstream tasks. The GPT-3 eliminates the need for fine-tuning.

The size of the three GPT models differs amongst them. The first Transformer Model had over 110 million parameters. GPT-1 accepted the size, whereas GPT-2 increased the number of parameters to 1.5 billion.

With the addition of 175 billion parameters to GPT-3, it became the biggest neural network. GPT-3 was designed to be more robust than GPT-2 in that it can address more specialized issues. It was recognized that GPT-2 performed poorly in specialized fields like as music and narrative.

GPT-3 can do more complex tasks like as question answering, essay writing, text summary, language translation, and computer code generation. Data contamination is a unique problem that GPT-3 must address.

Due to the internet origin of their training dataset, it is probable that part of the training data may overlap with the testing data.

Although this subject was addressed in GPT-2, it is especially pertinent to GPT-3 175B due to the fact that the datasets and models used for GPT-3 175B are about two orders of magnitude greater than those used for GPT-2, hence increasing the likelihood of contamination and memorization.

To study the effect of data contamination, the OpenAI team generates a “clean” version of the testing dataset for each downstream job, removing any possibly leaky instances, loosely described as “examples having a 13-gram overlap with anything in the training set”.

They next analyze GPT-3 on these “cleaned” test datasets and compare the results to those of the “uncleaned” original datasets.

Table 1. Comparison of embedding methods

Embedding Method	Model Type	Context Sensitivity	Architecture	Training Method	Use Cases	Strengths	Limitations
Word2Vec	Static	None	Feed-forward Neural Network	Predictive (Skip-gram, CBOW)	Text classification, document retrieval	Efficient, good for semantic relationships	No context sensitivity, struggles with polysemy
GloVe	Static	None	Matrix Factorization	Global statistical info	Text classification, keyword search	Captures global word co-occurrences	Static representations, same issues as Word2Vec
FastText	Static	None	Feed-forward Neural Network	Predictive	Text classification, morphological analysis	Handles subword information well	Still lacks contextual understanding
ELMo	Contextual	Yes	LSTM (Bidirectional)	Contextual embeddings	Sentiment analysis, question answering	Captures context, handles polysemy	Computationally intensive
BERT	Contextual	Yes	Transformer (Bidirectional)	Masked language modeling	Question answering, dialogue systems	Excellent at understanding context	High computational cost, input length limits
GPT-3	Contextual	Yes	Transformer (Unidirectional)	Generative pre-training	Text generation, creative writing	Strong text generation capabilities	High resource demand, interpretability issues
ChatGPT	Contextual	Yes	Transformer (Fine-tuned)	Fine-tuning on dialogues	Conversational agents, chatbots	Maintains context in conversations	Bias issues, high latency

4.2.4 Linformer

Experiments with traditional Bidirectional Transformers demonstrate that a context mapping matrix has a low rank [38]. The structure of the trials was predicated on the observation of ensuing unique values of P across multiple layers and different heads, averaged across 10k words, after the learning process.

Long-tail spectrum distribution is uniform throughout each head and layer. Consequently, the matrix P may be estimated with little performance loss by using a more straightforward representation.

In addition, we should note that the upper levels have a context mapping matrix P with a lower rank than the lower layers. Consequently, such a characteristic gives rise to a number of dimension-reduction-related optimization strategies, such attention kernalization and attention low-dimensional projections.

Consequently, using the Eckart–Young–Mirsky Theorem [12], one may approximate self-attention using a low-rank representation P_{low} with time and space complexity reduced to $O(nk)$. In this way, the resultant model is as follows:

$$\begin{aligned}\overline{\text{head}_i} &= \text{Attention}\left(QW_i^Q, E_iKW_i^K, F_iVW_i^V\right)\\ &= \underbrace{\operatorname{softmax}\left(\frac{QW_i^Q\left(E_iKW_i^K\right)^T}{\sqrt{d_k}}\right)}_{\bar{P}:n\times k}\cdot\underbrace{F_iVW_i^V}_{k\times d},\end{aligned}\quad (17)$$

where $E_i, F_i \in R^{n \times k}$key linear projection matrices and query matrices coincide. Thus, a relatively tiny projected dimension k, such that $k \ll n$, should be used to significantly reduce memory and space usage.

4.2.5 Longformer

The strategy that was presented by the authors of the paper "Longformer: The Long-Document Transformer" [2] was based on the concept of shrinking the attention window in such a manner that it would fit particular regions, hence reducing the amount of memory needed and the amount of time needed to execute calculations.

How were these objectives attained? They employed three types of attention. The sliding attention window is the first of them. Instead of taking the whole $n \times n$ window, restrict it to the main diagonal of the attention matrix with a width of w, where each token attends to $1/2w$ tokens on each side, as shown in the graphic below.

Thus, the complexity of computing becomes $O(n \times w)$. Based on this technology, the authors devised a technique with a far larger receptive field by expanding the windows even further. This Dilated sliding window introduces the d parameter, which defines the distance between windows.

However, we know that in multi-head attention approaches, various heads have varied attention scores, therefore by varying the dilation window, the authors were able to accomplish the outcome that some heads focus more on local context and others on long context.

The authors devised a second method dubbed "global attention" that utilizes unique tokens to further restrict the attention window. This strategy is based on increasing emphasis symmetrically at certain spots, based on the positioning of specific tokens in text. However, the number of these global attention windows is still quite modest compared to the total size of a $n \times n$ window, therefore the complexity remains $O(n)$.

4.2.6 BigBird

The BigBird method, a descendant to the Longformer algorithm explained above, employs similar attention window principles. By combining global attention and window attention with so-called random attention, Zaheer et al. hypothesized that they may get superior outcomes in Natural Questions Long Answer (LA), TriviaQA, and WikiHop activities [41].

A computation approach that boosts the GPU/performance TPU's is a further technological advancement. The attention matrix is divided into blocks of size 2 $\times$ 2, which accelerates retrieval owing to sparsity, while "rolling" is the transformation of a sparse matrix into a smaller nonsparse counterpart for quicker computing.

4.2.7 Synthesizer

Synthesizer model has no the query-key-values block in the self-attention module, instead of it the context matrix mapping is directly synthesized from input x [36]. For simplicity, in the current paper the explanation assumes the per head and per layer computation. The first step of Synthesizer model is an projection parametrized function $F_{h,\ell} : R^d \to R^l$:

$$B_{i,h,\ell} = F_{h,\ell}\left(X_{i,h,\ell}\right), \tag{18}$$

where an input $X_{h,\ell} \in R^{N \times d}$, i is a number of token. Intuitively, this can be interpreted as learning a token-wise projection to the sequence length N. Essentially, with this model, each token predicts weights for each token in the input sequence. In such a manner, the prediction function can be presented as follows:

$$F_{h,\ell}\left(X_{i,h,\ell}\right) = W_{2,h,\ell}\left(\sigma_R\left(W_{1,h,\ell}\left(X_{i,h,\ell}\right)\right)\right), \tag{19}$$

where σ_R is ReLU activation function, $W_{1,h,\ell} \in R^{d \times d}$, $W_{2,h,\ell} \in R^{d \times l}$ is two simple layers of neural network:

$$Y_{h,\ell} = \mathrm{softmax}\left(B_{h,\ell}\right) \cdot G_{h,\ell}\left(X_{h,\ell}\right), \tag{20}$$

where $G_{h,\ell}(.)$ is additional parameterized function of X that is similar to value matrix $V_{h,l}$ in the standard Transformer model.

This approach eliminates the dot product attention $Y = \mathrm{softmax}(Q_{h,l}K^T_{h,l})V_{h,l}$, altogether by replacing query matrix Q in standard Transformers with the synthesizing function $F_{h,\ell}$.

5 Data

In our experiments, we utilized a variety of datasets to evaluate the performance of different word embedding models. Table 1 lists the datasets used for each model, along with a brief description of their contents: The use of these datasets not only allows for the assessment of the performance of various models but also helps identify their strengths and weaknesses across different contexts and tasks.

6 Evaluation

In our evaluation, we aimed to compare the efficiency of various word embedding methods by employing several performance metrics tailored to different tasks. We include these types of dataset:

1. Massive Multitask Language Understanding (MMLU), which comprises a diverse array of tasks spanning multiple domains, including mathematics, science, and social studies. MMLU serves as a critical benchmark for assessing the reasoning capabilities of language models across varying levels of difficulty.
2. The Stanford Question Answering Dataset (SQuAD) was also utilized; this reading comprehension dataset requires models to answer questions based on a set of Wikipedia articles, making it instrumental in evaluating their abilities to understand and extract relevant information from text.
3. Additionally, we employed GLUE (General Language Understanding Evaluation) and SuperGLUE, which consist of a collection of tasks designed to evaluate model performance on various language understanding challenges, including sentiment analysis and textual entailment.
4. Other domain-specific datasets, such as the TREC AP corpus and the Brown corpus, were also included to evaluate performance in specific contexts.

In order to ensure comparability, the input data and the training conditions were standardized in the experimental setups for each model. This also included preparation of the text resources from the chosen datasets, such as cleaning and tokenization. In the case of multilingual datasets, certain rules regarding tokenization were followed in order to suit other languages.

Each word embedding model was fitted to data trained on the respective corpus using standard settings. In the case of BERT, GPT-2, and T5 models, we applied their pre-trained versions and adjusted them on the specific datasets to achieve better outcomes.

Hyperparameters, including but not limited to learning rate, batch size, and the number of training epochs, were thoroughly tuned for each model so as to improve performance. Various metrics were employed to assess model performance.

1. Accuracy is the ratio of the total number of correct predictions made out of the total population of predictions by the model, and this has been reported in most text classification tasks of the system performance.
2. The F1 score, which is the mean of recall and precision, is very applicable to problems with skewed classes because it helps to strike a chord between the positives retrieved and instances that are required.

By employing these metrics, we can effectively compare the efficiency of different word embedding methods. This approach enables us to evaluate not only how accurately the models predict or classify data but also how well they generalize across various tasks and datasets.

Furthermore, modern benchmarks like MMLU, XNLI, and SQuAD enable a robust evaluation framework by incorporating diverse tasks and languages, ensuring that the models are assessed under realistic conditions.

Analyzing performance across these datasets and metrics provides insights into the strengths and weaknesses of each embedding method, ultimately contributing to ongoing advancements in natural language processing.

7 Discussion

In this section, we reflect on the findings regarding matrix factorization and neural network methods for generating embeddings, particularly in the context of natural language processing and recommendation systems.

7.1 Matrix Factorization Methods vs. Neural Network Methods

Matrix factorization is a method that breaks a given Matrix into parts in such a way that a lower-dimensional representation of it is produced. A well-known instance of this is seen in collaborative filtering, where the user-item matrix is broken into user and item latent feature matrices A and B correspondingly.

Thus, within this class, there are numerous known factorization techniques, which include Singular Value Decomposition (SVD) as it shrinks a high dimensional data but retains important information; Non-negative Matrix Factorization (NMF) as it is suitable for applications where it makes sense to constraint vendors' factors to positive values; and Probabilistic Matrix Factorization (PMF) which means addition of a probabilistic approach to the process of the factorization for the purpose of improving its robustness.

Matrix factorization offers numerous benefits to users. They are easy to apply and understand which allows for their use in many situations. Also, they can be performed on very large and sparse interaction patterns very well. Further, they seem to have the best performance when dealing with the cold start problem especially when the new item/user has very few or no data at all.

This is achieved by tapping into the hidden factors from the simple product metrics present in the implemented system. That being said, it is not without weaknesses that matrix factorization has come. Most of them will be based around data which leads to a problem of linearity which does not explain more creative and complex data. Without a deeper description of temporal or contextual information, there is the possibility that these procedures are not going to be entirely effective in any problem where these aspects are critical. It is likely to perform poorly in problems where it is necessary to appreciate an even higher level of meaning than carried different perspectives. On the other hand, there are differences, especially in the use of thin portions of these matrices because convolution neural networks and neural network structures such as fast hidden units enable horizontal problems to be addressed more effectively. They are also made up of several layers used for feature extraction and classification analysis ready via softwares in CNN.

Some of the widely applied varieties within neural network methodologies are feedforward neural networks having adjacencies with neurons exchanging information in one neural direction; recurrent neural networks, which process quasi-periodic spectral components and design utilizing recurrent neural connections spanning over a length of time; classifiers which act as a series of connected operations in CNN; and finally, transformer models. Which has been an incredible transformation in advancing natural language processing – BERT and GPT – by means of capturing long-range dependencies. Neural network techniques serve the purpose very well. They are good at grasping intricate, non-linear patterns within the data making it easier to improve the performance on different fronts.

Besides, some state-of-the-art models such as BERT and GPT are equipped with strong contextual embeddings that contribute to a better understanding of meaning and relations between data points. These models also do well in that they are highly changeable, allowing them to be altered

Table 2. Datasets and models used in experiments

Model	Dataset
Latent Dirichlet Allocation	16,000 documents from a subset of the TREC AP corpus
Neural Network Language Model	Brown corpus, Associated Press (AP) News from 1995 and 1996
Latent Semantic Analysis	MED, CRAN, CACM, CISI
Word2Vec	Google News corpus
GloVe	2010 Wikipedia dump, 2014 Wikipedia dump
FastText 2016	Wikipedia data in nine languages: Arabic, Czech, German, English, Spanish, French, Italian, Romanian, and Russian
ELMo	One billion word benchmark for measuring progress in statistical language modeling (Chelba 2014)
BERT	BooksCorpus and English Wikipedia
XLNet	BooksCorpus and English Wikipedia, Giga5 (16GB text), ClueWeb 2012-B, Common Crawl
ELECTRA	ClueWeb, CommonCrawl, and Gigaword
Reformer	enwik8 and imagenet64
Sparse Transformer	CIFAR-10, Enwik8, ImageNet 64x64
GPT-1	BooksCorpus dataset
GPT-2	Common Crawl, WebText
GPT-3	Common Crawl (filtered) 410 billion, WebText2 19 billion, Books1 12 billion, Books2 55 billion, Wikipedia
Linformer	BookCorpus, English Wikipedia
Longformer	text8 and enwik8
BigBird	Books, CC-News, Stories, and Wikipedia
Synthesizer	SuperGLUE
RoBERTa	SQuAD, GLUE
XLM-R	XNLI
DeBERTa	MMLU, SQuAD
T5 (Text-to-Text Transfer Transformer)	MMLU, GLUE, SuperGLUE, SQuAD

according to the requirements and application making them suitable for any type of given problem.

On the other hand, it is not all roses. There are cons to the application for neural networks due to several reasons. These models are computationally expensive and calls for huge resources in terms of memory size and processing capability. This may put a disadvantage to some businesses, particularly.

Also, most applications of the machine learning Neural Networks typically require rich training data which could be difficult to obtain in some applications. Finally, a key component, which is training process for such models is usually longer as compared to matrix factorization methods hence being unable to meet deadlines on certain tasks.

Projection of User-Item invocations onto k-dimensional spaces is also a general process that is more suitable for simpler tasks and situations where understandability, and ubiquitous and geographically expansive use is the norm, as opposed to capturing the different hidden layers of a network.

The essence of Neural Network Methods is on the other hand more robust and target based as it is well adapted for complex, high-dimensional data and intense contextual relationships but incurs higher resource outlay and structural intricacies.

It is mostly focused on the needs of a specific task, including but not limited to the volume of information such as data, performance and computational infrastructure level, and the aspect of the problems or solution under consideration.

7.2 Differences between Embedding Methods

In the course of the development of natural language processing (NLP), historically famous models like Word2Vec and GloVe have attracted interest with their understanding specialized in the word embeddings. However, as the evolution of the field entails, new models are developed such as BERT, GPT-3, and ChatGPT, which constitute solutions for significantly advanced problems.

Models of content-dependent and independent models were discussed in the previous sections. But no comparative studies were given. Types of context-independent models (Word2Vec and GloVe) are useful in conducting basic functions including text classification, document retrieval, and tasks where context is not important. They are less time-consuming and can be trained with smaller datasets.

On the other hand, context-dependent models such as BERT and ELMo are more efficient for advanced operations e.g., understanding the relationship between concepts, answering questions, translating. They are most useful in text manipulation due to the high level of precision, however, they require more resources and a larger sample data set for training.

As described earlier, not all embeddings are created equal as they have intricate differences in terms of their design, training procedures, use, and effects. Various embeddings are employed because different tasks underscore different strategies in text manipulation and understanding of the context. We will also focus on the differences between different types of embeddings, their benefits and drawbacks, and their applicability in various tasks.

7.2.1 Traditional Models: Word2Vec and GloVE

The model developed by [24], known as Word2Vec, works by constructing vectors such that the relationships between words are predicted. Word embeddings can be created using the Skip-gram and Continuous Bag of Words model implementations. The ultimate result of this guideline is that it considers the distance or being among the words in terms of meaning but does not consider their abstract meaning in context, making use of the same word sense in multiple situations (polysemy) problematic. On the contrary, static efficiency is known as the inefficiency of traditional methods in handling such cases due to the surrounding words or, simply put, the context. This is the same case for GloVe. This fails on the notions of global statistics which comes from the complete corpus leading to static representation and insensitivity of context. These embeddings can come in handy for use in such situations, those being:

– When a quick solution is needed and deep text analysis is not required;
– When the data volume is limited, and as a result, the context does not matter;
– For tasks like text classification, keyword-based search, and creation of recommendation systems.

7.2.2 Limitations of Traditional Models

The adaptivity of embeddings from the Word2Vec and GloVe models is limited by the fact that these vectors are constant regardless of the given semantic unit. This is unsuitable for fine linguistic uses. The two technologies are economizing of computational resources allowing for hardly more burdensome operations at an immense proportion whilst the effectiveness of these technologies falls back very quickly when large data sets and advanced NLP applications are used. This is one of the reasons why they are not good is real-world applications. The following are the most significant they did not include the following limitations:

– No Context Sensitivity: Due to the nature of such embeddings, they provide a fixed vector to every word such that a word's meaning cannot be distinguished in different contexts. For example, the word 'bank' is the same whether it is used about a financial institution or a riverbank;

– Dealing with Ambiguity: If there are two meanings to one word, for instance the verb "lead" and the metal "lead" the situation I the same and this is one of the causes to error in fierce contextualized work patterns such as machine translation and answering questions;

– Sparse Data Headache: The most difficult part in working with these models is helping them cope with training statistics that never catches up with some target vocabulary items. This follows that some rare words or expressions will be hard to learn even when using these models;

– Limited Reach: They are typically meant to capture co-occurrences of words within a small contiguous region— hence they cannot report word relations which span beyond the window;

– Fixed Representations: Embedded representations do not admit of variability that would take into account changing conditions within the context such as a word's difficulty;

– Data Requirement: These methods call for huge amounts of well-prepared data so that the provided embeddings will be correct, and it is impossible to represent anything informative with a limited amount of data.

7.2.3 Newer Models: BERT, GPT-3, and ChatGPT

Nevertheless, BERT (Bidirectional Encoder Representations from Transformers) makes use of a transformer architecture to generate embeddings for a given term considering both left-to-right and right-to-left context.

Consequently, BERT is good at various NLP benchmarks because it can cope with the ambiguity and the context in which it is available. Different from it, GPT-3 (Generative Pre-trained Transformer 3) employs a one-way transformer model trained on an enormous source text.

The collaborative writing abilities such as few-shot text generation and coherence text enable to aid in the creation of text or stories.

The modifying of GPT-3 to be a chatbot is referred to as ChatGPT, and this improvement assists in context preservation and engagement dissonance. We present a criterion based on when it is preferable to use content-specific approach for these reasons:

– When There is need of a deep understanding of the context controlling a certain word: These types of embeddinings are best suited for tasks that are dependent on the context of words. This includes tasks like question-answering, sentiment analysis, and natural language understanding;

– Polysemy and Ambiguity are Both addressed by content Dependent models: One of the examples: The word "bank" in one case is a financial institution and in another case it is a side of a river, in such cases, contextual embeddings differentiate these two meanings according to the context in which the word is used;

– Cardinals in Dialogue Agents and Conversational AI: For such systems as chatbots or virtual agents, continuing and comprehending the context across utterances are pivotal, with the model used GPT-3 and its customization into ChatGPT being best at producing contextually appropriate responses;

– Tasks with Entailment: in cases where there exist words or phrases that are semantically related, but are way too distant in a sentence or even in a document, embedding models work better than non-embedding models;

– Text Summoning and Synthesis: Content-aware models generate relevant and context-coherent text by understanding the organization of the input text in the cases of machine translation, text summarization or even fiction writing;

– Chores that Offer a Challenge to Transferrable Mechanisms Such As Machine Translation or Restatement: In contrast, when changing languages or redirecting meaning to certain sentences, contextually-dependent models demonstrate a better understanding of the sentence or paragraph construction and its purpose, hence expressing translations or modifications more accurately;

– Wide-Ranging and Varied Tasks: Moreover, such models are capable of carrying out Intelligent NLP designs for multiple and wide-ranging tasks like language translation when dealing with big varying data because they are capable of self-adjusting their knowledge instantly.

7.2.4 Limitations of Newer Models

Even several outstanding developments such as the BERT, GPT-3, and ChatGPT models seem to have their own obstacles. These new designs can undertake at greater length various problems and perform well in several tasks than the traditional designs; there comes a cost, however, because of their capacity and they have some setbacks. let us look at some of the draw backs and issues that these modern models have as they may be needed.

Here is a short version for dependent representations of the content such as BERT, GPT-3, and ChatGPT:

– Appropriate for tasks that require more than basic understanding; it is more necessary with tasks like sentiment analysis or answering questions;

– Mainly linguistic Use in a broader sense gets intensive sedated erasing within given context;

– Best for pertinent usage cases, i.e. chatbots where one needs to preserve the context through multiple conversational turns;

– Used for requests with words that are geographically far from one another;

– Outputs are generated ensuring coherence and text relevancy;

– Outstanding in translation and rephrasing by perceiving the meaning of a sentence;

– Able to address the issue of different contexts correctly and efficiently in different data worlds.

7.2.5 Comparative Analysis

The advancements in context handling are significant; while traditional models provide static representations, BERT and GPT-3 adapt embeddings based on context, leading to improved performance. In terms of scalability, newer models leverage large datasets and sophisticated architectures, allowing them to scale effectively in complex applications.

Furthermore, BERT and GPT-3 demonstrate superior efficiency in real-world applications, showcasing their ability to understand and generate human-like text, thus addressing the challenges faced by traditional models. Table 2 provides a comparative overview of various word embedding methods, summarizing key aspects of each model.

7.3 Special Types

The landscape of Natural Language Processing (NLP) is continuously evolving, and with it comes an explosion of specialized embedding techniques designed to better suit individual needs or use cases. This section takes an in-depth look at some of the special types of embeddings that help NLP models to learn better and perform well.

7.3.1 Multimodal Embeddings

This offers multimodal embeddings (that is, information from textual and visual or auditory modalities) for models to be able to more extensively capture context..) These embeddings are exceptionally effective in captioning a picture, understanding the content of a video and cross-modal retrieval.

For example, models such as CLIP (Contrastive Language-Image Pretraining) use textual and visual information together to increase the accuracy in tasks where fine-grained knowledge of both modalities is needed.

7.3.2 Domain-specific Embeddings

So domain-specific embeddings literally mean they are embeddings which are specifically trained on a certain field of text such as the biomedical, legal or financial texts.

The additional embeddings are useful for capturing customized lingo and context that broader, or generalized perhaps the word in am looking for, embeddings might miss. BioBERT, on the other hand, is BioMedically-oriented BERT and training it on biomedical literature helps the model to better understand and process domain-relevant information.

7.3.3 Contextualized Embeddings

The main goal of contextualized embeddings was to account for context (that adds a higher level) — that means words change representations based on surrounding text. ELMo, BERT, RoBERTa which are contextualized embeddings help to solve challenges like the presence of homonymous or polysemous tokens. Such flexibility vastly enhances the effectiveness of NLP duties as it permits spacious semantic representations.

7.3.4 Graph-based Embeddings

Graph-based embeddings come from graph structures, showing connections between entities in a network. These embeddings are important in knowledge graphs, social networks, and recommendation system applications. Models such as Node2Vec and GraphSAGE generate embeddings by the graph topology, which results in learning much deeper relational data.

7.3.5 Temporal Embeddings

Time-aware entities are temporal embeddings to capture the changes in meaning or usage from a historical context. That is useful for trend analysis, event prediction, and time-relevant recommendations. Models that use temporal information, like Temporal Graph Networks, tend to be able to model well the language dynamics and how it walks over time.

7.3.6 Adversarially Trained Embeddings

Adversarially trained embeddings are adversarially robust and incrementally learned by an adversarial training approach. This is particularly important in applications like sentiment analysis and spam detection, where models must perform well even when attacked. The adversarial training techniques along with these embeddings, along improve the quality and reliability of the model as a whole.

7.3.7 Zero-shot and Few-shot Learning Embeddings

Zero-shot learning (and few-shot learning) embeddings are meant for cases in which models are required to generalize from only a handful of examples. These embeddings are very beneficial in classification and translation tasks with less or no labeled data. This is how GPT-3 models work, they are capable of performing a wide range of tasks with few examples, this way opening more possibilities for using NLP systems.

8 Bias in Word Embedding

This topic of bias in word embeddings is particularly relevant as we continue to move towards artificial intelligence and natural language understanding/processing. Societally-biased training data can create models that perpetuate, or worse, amplify societal biases.

For example, a word embedding that encodes gender stereotypes may lead to models associating specific genders with certain professions or characteristics which in turn can lead to discrimination and bias in hiring practices, exposure within media, etc.

While likewise, racial prejudices can be present in how communities are represented by data-driven applications. These biases have severe ethical ramifications, especially in applications that require fairness and equity.

The same type of scrutiny should be used with the datasets on which AI systems are trained as organizations increasingly deploy them. This has spawned a burgeoning set of communities demanding the ethical development of AI, calling on researchers and practitioners to think first about fairness, accountability and transparency in their work. To extend this discussion, we have found more recent research a useful reference as in [3] who write about the dangers of using large language models and the ethical obligations that their developers have.

Additionally, [30] have explored the implications of multimodal models that utilize language and vision together, emphasizing the need for careful consideration of bias in both domains. By addressing these ethical concerns and incorporating a comprehensive analysis of bias in word embeddings, we can contribute to a broader understanding of the implications of AI technologies in society. This approach not only enhances the readership of our work but also aligns with the growing emphasis on fairness in AI research and development.

9 Future Challenges

Presented work is dedicated to the fundamental task of mapping human language concepts into a vector space, with the preservation of topological properties in terms of semantics and lexics. This paper discusses the features of various approaches that developed in neurocomputing science. The paper's key analysis and findings are described as follows.

This paper outlines unsettled tasks posed by the modern state of industry and technology, and proposes several related approaches as well. In details, despite all the mentioned successes in the field of natural language processing [11, 34, 40, 10], several authors [20, 1, 19] showed that most modern systems are "fragile" and "fictitious".

Some researchers suggest that the voiced problems should be solved by integrating "common sense" at various levels of word processing, including the level of word vectorization. Attention-based models also have such drawbacks.

The peak accuracy of 77 for BERT in the problem of understanding arguments reaches only three points below the average level of an unprepared person.

However, this result is entirely explained by false statistical patterns in the data set. The reason these models achieve random accuracy lies in the methodology for constructing the data set [26].

To study how BERT "makes a decision," examples were considered that are most easily classified through many runs of the algorithm.

The authors of [15] performed a similar analysis with the SemEval dataset, and as evidence of their results, it was found that BERT uses the presence of a hint word to confirm a sentence, for example, denial words such as "no" or "not."

Through point experiments, a method isolating such an effect was developed; indeed, the accuracy of the BERT model can be entirely dependent on random statistical laws [26]. The question of "fragility" and "deceit" of XLNet remains open; studies in the areas of machine translation, machine reading of a text, construction of argumentative models, tonality analysis, and others seem interesting.

In our opinion, there are three main areas of development of natural language processing algorithms, namely:

1. The use of the inductive bias approach for better control over the result, the use of linguistic structures in neural network architectures is one of the main trends of 2017.

   It should be noted that such architectures become more dependent on the manual work of researchers and developers. Nevertheless, the approach allows learning more complex behavior with fewer data.

2. The integration of common sense in the natural language processing model; indeed, most of the urgent tasks of text processing involve such qualities as abstraction, logic, comprehensive knowledge of the world.
3. Modeling data and distributions that do not belong to the training set, since most systems are oriented to a specific knowledge domain and do not have, in a broad sense, the quality of generalization.

Acknowledgements

This research has been/was/is funded by the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No. AP23489782).

References

1. **Belinkov, Y., Bisk, Y. (2017).** Synthetic and natural noise both break neural machine translation. International Conference on Learning Representations, pp. 1–13.
2. **Beltagy, I., Peters, M. E., Cohan, A. (2020).** Longformer: The long-document transformer. DOI: 10.48550/ARXIV.2004.05150.
3. **Bender, E. M., Gebru, T., McMillan-Major, A., Shmitchell, S. (2021).** On the dangers of stochastic parrots: Can language models be too big? Proceedings of the ACM Conference on Fairness, Accountability, and Transparency, pp. 610–623. DOI: 10.1145/3442188.3445922.
4. **Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C. (2003).** A neural probabilistic language model. The Journal of Machine Learning Research, Vol. 3, pp. 1137–1155.
5. **Blei, D. M., Ng, A. Y., Jordan, M. I. (2003).** Latent dirichlet allocation. The Journal of Machine Learning Research, Vol. 3, pp. 993–1022.
6. **Bojanowski, P., Grave, E., Joulin, A., Mikolov, T. (2017).** Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, Vol. 5, pp. 135–146. DOI: 10.1162/tacl_a_00051.
7. **Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., et al. (2020).** Language models are few-shot learners. Advances in Neural Information Processing Systems, Vol. 33, pp. 1877–1901.
8. **Cho, K., van-Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y. (2014).** Learning phrase representations using RNN encoder–decoder for statistical machine translation. Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1724–1734. DOI: 10.3115/v1/D14-1179.
9. **Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., Bengio, Y. (2014).** Learning phrase representations using RNN encoder-decoder for statistical machine translation. CoRR, Vol. abs/1406.1078.
10. **Clark, K., Luong, M. T., Le, Q. V., Manning, C. D. (2020).** Electra: Pre-training text encoders as discriminators rather than generators. Proceedings of the International Conference on Learning Representations. DOI: 10.48550/arXiv.2003.10555.
11. **Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2019).** BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.
12. **Eckart, C., Young, G. (1936).** The approximation of one matrix by another of lower rank. Psychometrika, Vol. 1, No. 3, pp. 211–218. DOI: 10.1007/bf02288367.

13. **Firth, J. R. (1957).** A synopsis of linguistic theory, 1930-55: Studies in linguistic analysis. Blackwell.

14. **Gomez, A. N., Ren, M., Urtasun, R., Grosse, R. B. (2017).** The reversible residual network: Backpropagation without storing activations. Advances in Neural Information Processing Systems, Curran Associates, Inc, Vol. 30, pp. 1–11.

15. **Habernal, I., Wachsmuth, H., Gurevych, I., Stein, B. (2018).** SemEval-2018 task 12: The argument reasoning comprehension task. Proceedings of the 12th International Workshop on Semantic Evaluation, Association for Computational Linguistics, pp. 763–772. DOI: 10.18653/v1/s18-1121.

16. **Hochreiter, S., Schmidhuber, J. (1997).** Long short-term memory. Neural Computation, Vol. 9, No. 8, pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

17. **Hofmann, T. (2013).** Probabilistic latent semantic analysis. DOI: 10.48550/ARXIV.1301.6705.

18. **Idel, M. (2020).** Gematria and prognostication. pp. 785–787. DOI: 10.1515/9783110499773-052.

19. **Iyyer, M., Wieting, J., Gimpel, K., Zettlemoyer, L. (2018).** Adversarial example generation with syntactically controlled paraphrase networks. Vol. 1, pp. 1875–1885. DOI: 10.18653/v1/N18-1170.

20. **Jia, R., Liang, P. (2017).** Adversarial examples for evaluating reading comprehension systems. Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, pp. 2021–2031. DOI: 10.18653/v1/D17-1215.

21. **Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., Mikolov, T. (2016).** Fasttext.zip: Compressing text classification models. DOI: 10.48550/ARXIV.1612.03651.

22. **Joulin, A., Grave, E., Bojanowski, P., Mikolov, T. (2017).** Bag of tricks for efficient text classification. Vol. 2, pp. 427–431.

23. **Kitaev, N., Kaiser, L., Levskaya, A. (2020).** Reformer: The efficient transformer. arXiv. DOI: 10.48550/ARXIV.2001.04451.

24. **Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013).** Efficient estimation of word representations in vector space. DOI: 10.48550/ARXIV.1301.3781.

25. **Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J. (2013).** Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems, Vol. 26, pp. 3111–3119.

26. **Niven, T., Kao, H. Y. (2019).** Probing neural network comprehension of natural language arguments. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 4658–4664. DOI: 10.18653/v1/P19-1459.

27. **Pennington, J., Socher, R., Manning, C. (2014).** Glove: Global vectors for word representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, pp. 1532–1543. DOI: 10.3115/v1/d14-1162.

28. **Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L. (2018).** Deep contextualized word representations. Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1. DOI: 10.18653/v1/n18-1202.

29. **Peters, M. E., Neumann, M., Zettlemoyer, L., Yih, W. T. (2018).** Dissecting contextual word embeddings: Architecture and representation. Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, pp. 1499–1509. DOI: 10.18653/v1/D18-1179.

30. **Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I. (2021).** Learning transferable visual models from natural language supervision. Proceedings of the 38th International Conference on Machine Learning, Vol. 139, pp. 1–16.

31. **Radford, A., Narasimhan, K. (2018).** Improving language understanding by generative pre-training. api.semanticscholar.org/CorpusID:49313245.

32. **Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. (2019).** Language models are unsupervised multitask learners. api.semanticscholar.org/CorpusID:160025533.

33. **Röder, M., Both, A., Hinneburg, A. (2015).** Exploring the space of topic coherence measures. Proceedings of the 8th ACM International Conference on Web Search and Data Mining, pp. 1499–1509. DOI: 10.1145/2684822.2685324.

34. **Sanh, V., Debut, L., Chaumond, J., Wolf, T. (2019).** DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. DOI: 10.48550/ARXIV.1910.01108.

35. **Sanzhar, A., Pak, A., Bulatovna, J. A. (2018).** The estimation of stability of semantic space generated by word embedding algorithms. International Joint Symposium on Artificial Intelligence and Natural Language Processing, pp. 1–5. DOI: 10.1109/iSAI-NLP.2018.8692814.

36. **Tay, Y., Bahri, D., Metzler, D., Juan, D. C., Zhao, Z., Zheng, C. (2021).** Synthesizer: Rethinking self-attention for transformer models. Proceedings of the 38th International Conference on Machine Learning, Vol. 139, pp. 10183–10192. DOI: 10.48550/arXiv.2005.00743.

37. **Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., Polosukhin, I. (2017).** Attention is all you need. Advances in Neural Information Processing Systems, pp. 5998–6008.

38. **Wang, S., Li, B. Z., Khabsa, M., Fang, H., Ma, H. (2020).** Linformer: Self-attention with linear complexity. DOI: 10.48550/ARXIV.2006.04768.

39. **Winata, G. I., Madotto, A., Lin, Z., Liu, R., Yosinski, J., Fung, P. (2021).** Language models are few-shot multilingual learners. Proceedings of the 1st Workshop on Multilingual Representation Learning, Association for Computational Linguistics, pp. 1–15. DOI: 10.18653/v1/2021.mrl-1.1.

40. **Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., Le, Q. V. (2019).** Xlnet: Generalized autoregressive pretraining for language understanding. Advances in Neural Information Processing Systems, pp. 5753–5763.

41. **Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., Ahmed, A. (2020).** Big bird: Transformers for longer sequences. Proceedings of the 34th International Conference on Neural Information Processing Systems, pp. 17283–17297.

Article received on 08/07/2024; accepted on 02/10/2024.
**Corresponding author is Alexander Gelbukh.*