applied sciences

MDPI

Review

Contemporary Approaches in Evolving Language Models

Dina Oralbekova [1,2,*], Orken Mamyrbayev [1], Mohamed Othman [3], Dinara Kassymova [2] and Kuralai Mukhsina [1]

[1] Institute of Information and Computational Technologies, Almaty 050060, Kazakhstan; morkenj@mail.ru (O.M.)
[2] Information and Communication Technologies Department, Institute Automation and Telecommunication, Academy of Logistics and Transport, Almaty 050026, Kazakhstan
[3] Department of Communication Technology and Networks, Universiti Putra Malaysia (UPM), Serdang 43400, Selangor D.E., Malaysia; mothman@upm.edu.my
* Correspondence: d.oralbekova@aues.kz; Tel.: +77-711-310-188

Abstract: This article provides a comprehensive survey of contemporary language modeling approaches within the realm of natural language processing (NLP) tasks. This paper conducts an analytical exploration of diverse methodologies employed in the creation of language models. This exploration encompasses the architecture, training processes, and optimization strategies inherent in these models. The detailed discussion covers various models ranging from traditional n-gram and hidden Markov models to state-of-the-art neural network approaches such as BERT, GPT, LLAMA, and Bard. This article delves into different modifications and enhancements applied to both standard and neural network architectures for constructing language models. Special attention is given to addressing challenges specific to agglutinative languages within the context of developing language models for various NLP tasks, particularly for Arabic and Turkish. The research highlights that contemporary transformer-based methods demonstrate results comparable to those achieved by traditional methods employing Hidden Markov Models. These transformer-based approaches boast simpler configurations and exhibit faster performance during both training and analysis. An integral component of the article is the examination of popular and actively evolving libraries and tools essential for constructing language models. Notable tools such as NLTK, TensorFlow, PyTorch, and Gensim are reviewed, with a comparative analysis considering their simplicity and accessibility for implementing diverse language models. The aim is to provide readers with insights into the landscape of contemporary language modeling methodologies and the tools available for their implementation.

Keywords: NLP; language modeling; neural networks; transformer; BERT; GPT

Citation: Oralbekova, D.; Mamyrbayev, O.; Othman, M.; Kassymova, D.; Mukhsina, K. Contemporary Approaches in Evolving Language Models. *Appl. Sci.* **2023**, *13*, 12901. https://doi.org/10.3390/app132312901

Academic Editors: Affan Yasin, Javed Ali Khan and Lijie Wen

Received: 30 October 2023
Revised: 21 November 2023
Accepted: 27 November 2023
Published: 1 December 2023

1. Introduction

Language modeling (LM) has steadily evolved into a cornerstone of modern computational linguistics and natural language processing (NLP), laying the foundations for sophisticated text analysis, generation, and understanding. The quintessence of language modeling lies in its ability to delineate probability distributions over sequences of words, thus furnishing a computational fabric for analyzing linguistic phenomena [1]. Contemporary trajectories in language modeling are significantly informed by robust mathematical and computational frameworks, big data analytics, neurophysiological methods, and traditional laboratory experimentation, all of which converge to elucidate the social and cognitive mechanisms underpinning language learning and processing [2].

The epochs of language modeling can be broadly demarcated into two epochs: the era of statistical models and the epoch of neural-network-based models. The statistical models, primarily hinged on probability theory and formal language theory, were adept at handling relatively small corpus sets, with models like n-grams and Hidden Markov

Models (HMMs) being emblematic of this era [3]. Conversely, the advent of neural-network-based models heralded a data-driven approach, leveraging substantially larger corpus sets and sophisticated neural architectures to attain unprecedented levels of performance across a spectrum of NLP tasks.

A notable inflection point in this evolutionary narrative was the inception of pre-trained language models (PLMs) like BERT, GPT, LLAMA, and BARD, which significantly elevated the state of the art in numerous content-understanding tasks [4]. These models, undergirded by the transformer architecture, have not only outperformed traditional models utilizing HMMs but have also demonstrated remarkable efficiency and simplicity in configuration and operation during both training and analysis phases.

Moreover, an assortment of neural network approaches, including those based on n-gram and hidden Markov models (HMMs), alongside modern neural network architectures such as transformer-based models (e.g., BERT, GPT), have been meticulously examined, encapsulating their architectures and training and optimization processes. These explorations have precipitated a plethora of modifications and enhancements to both standard and neural network architectures, thereby propelling the frontiers of language modeling to novel vistas of capabilities and applications.

Furthermore, the discourse around language modeling has extended to its applicability to agglutinative languages, presenting unique challenges and opportunities for the NLP community. The burgeoning ecosystem of libraries and tools like NLTK, TensorFlow, PyTorch, and Gensim has catalyzed the development, deployment, and evolution of language models, thereby democratizing access to sophisticated language modeling capabilities across the global research community.

In conclusion, the alacritous advancements in language modeling methodologies and technologies have not only significantly augmented the capabilities of NLP systems but also engendered a rich tapestry of research inquiries and applications across a broad spectrum of computational linguistic and NLP tasks. This article endeavors to provide a comprehensive exposition of the contemporary approaches in evolving language models, elucidating their architectures and training and optimization processes.

2. Statistical Approaches to Language Modeling

At the outset, traditional models such as n-grams and Hidden Markov Models (HMMs) are considered. These methods are statistical in nature and can be employed to evaluate the performance of emerging NLP models. Essentially, these models assign probabilities to sequences of words. N-grams and HMMs have a wide range of applications in various tasks of text analysis. They are used for the development of algorithms for automatic terminology extraction, the construction of language and acoustic models, and the extraction of process models from unstructured and structured data, among other functions. Although there are more efficient methods of text analysis available, it is crucial to understand that these models are more interpretable and can be tailored to specific NLP tasks. They can also serve as a benchmark for assessing emerging models and methods.

2.1. N-Gram

The n-gram model is one of the statistical models used for language modeling, employed for predicting the next word in a sequence or for assessing the probability of a sentence. The core principle revolves around predicting a sequence of n words or characters. In an n-gram model, the probability of producing the next word is determined based on transition probabilities and observations.

The general formula for calculating $P(w)$ depending on the number of words in a sentence is presented as follows (1):

$$P(w_1, \ldots, w_n) = P(w_n|w_{n-1}, \ldots, w_1) \cdot P(w_{n-1}, \ldots, w_1), \quad (1)$$

where w represents individual words, and n is the number of words.

There are various types of n-gram models (Figure 1), such as those listed below.

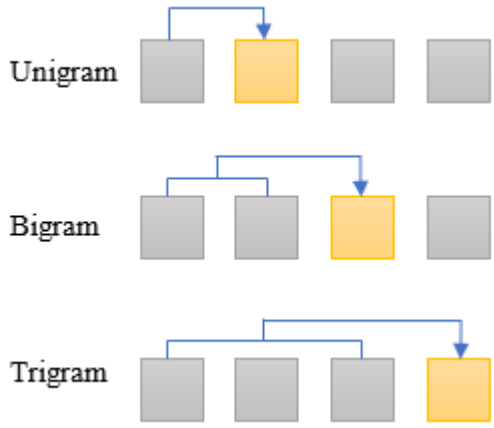

Figure 1. Types of n-gram models.

- Unigrams, which evaluate each word independently;
- Bigrams, which consider the probability of a word given the previous word;
- Trigrams, which consider the probability of a word given the two preceding words, and so on.

Unigrams play a fundamental role in n-gram models, assessing the probability of each word occurring independently within a specific context. Known for their simplicity and efficiency, unigram models serve as a foundational reference point for more intricate language models. Despite assuming independence, unigrams make substantial contributions to language comprehension and are essential in diverse natural language processing tasks, such as information retrieval, text classification, and sentiment analysis. Integrating unigrams into an n-gram model enhances its adaptability and suitability for various linguistic applications.

N-grams are relatively simple and efficient, yet they come with drawbacks:

- Data sparsity problem: This issue arises with low-frequency words or the absence of word groups in training data. To alleviate this, a Laplace smoothing method or Neyman–Morgenstern smoothing was proposed in [5]. In the realm of language modeling, n-gram models inherently fail to account for the long-term contextual dependencies present within word sequences.
- Model size problem: This leads to an exponential increase in possible combinations, demanding high memory and computational resources.
- Ignoring full sentence context: N-gram models consider only the previous context, overlooking the interrelation and broader context between words. Recurrent neural networks and their variants, as well as transformer-based models, have been proposed [6,7].
- Incorrect modeling of incomplete sentences.: N-grams work well with complete word sequences, but errors occur with incomplete phrases or sentences, a common scenario in real life. Methods of stream data processing or incremental model updating were applied to mitigate this drawback [8,9].

Some Modifications and Improvements

Chelba et al. [10] investigated the effective memory depth of RNN models by employing them for n-gram modeling. Experiments on a small UPenn Treebank corpus indicated that the LSTM cell with dropout is a superior model for encoding n-gram state data compared to a feedforward RNN. When maintaining the sentence independence assumption, the LSTM n-gram achieves performance comparable to the LSTM LM for $n = 9$ and slightly surpasses it when $n = 13$. The authors of [11] introduced the n-distant-max language model, which significantly mitigated the shortcomings of n-gram models with a smaller parameter for evaluation. However, a limitation of the n-distant-max model was identified: it only considers the word with the highest probability, even though there are instances in which a word is associated with more than one word. It was demonstrated in [12] that

by incorporating multi-granularity information about unseen and domain-specific words through word-based n-gram adaptation, the performance of the overall pre-trained model saw significant modeling improvements. This includes the Transformer-based Domain-aware N-gram Adaptor, which effectively learns and integrates semantic representations of various word combinations into a new domain. Li et al. [13] combined n-grams with neural LMs, allowing the neural component to focus on deeper language comprehension while also offering a flexible means of tuning the LM by switching foundational n-grams without altering the neural model. This approach demonstrated that one can effectively adapt to a domain by merely switching to the domain-specific n-gram model area without any additional training.

2.2. Hidden Markov Model (HMM)

The Hidden Markov Model (HMM) was one of the prevalent models used before the adoption of artificial neural networks and is applied in various NLP tasks, including language modeling (Figure 2). An HMM simulates the operation of a Markov process with hidden internal states. The Markov process itself represents a sequence of random events in which the probability of each event depends solely on the current state of the process and is independent of prior states [14]. In language modeling, an HMM is utilized to model word sequences while accounting for fundamental linguistic structures. An HMM consists of two primary elements: a hidden Markov chain and an observable sequence. The former comprises a series of hidden states with an unknown structure for observation. In many instances, these states can correspond to parts of speech. The latter element, the observable sequence, represents a series of characters or words observed during the modeling process. The HMM is characterized by the following parameters:

- $S = \{s_1, s_2, \ldots, s_n\}$—the set of hidden states;
- $O = \{o_1, o_2, \ldots, o_m\}$—the set of observable symbols or words;
- $A = \{a_{ij}\}$—the transition probability matrix, where a_{ij} is the probability of transitioning from state s_i to state s_j;
- $B = \{b_j(k)\}$—the observation probability matrix, where $b_j(k)$—is the probability of observing the symbol o_k in state s_j;
- $\pi = \{\pi_i\}$—the initial probability vector, where π_i—is the probability of starting in state s_i.

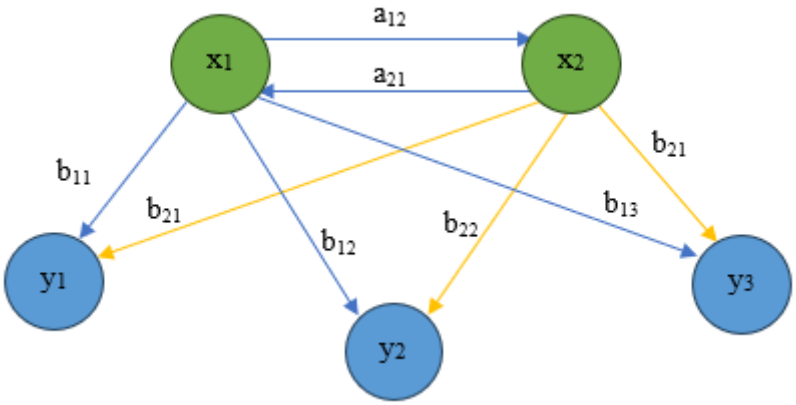

Figure 2. Example of a Hidden Markov Model.

In Figure 2, X represents the hidden state which depends on Y. Y in turn is the observable element, that is, a word, whereas a signifies the transition probability from one state to another and b stands for the emission probabilities.

In the domain of speech recognition, HMMs are typically employed in the construction of an acoustic model. They model the temporal sequences of phonemes and determine the most probable sequence of a model's states. This, in effect, accommodates both the transition probabilities between states and the observation probabilities. Moreover, HMMs

are used in tasks like speaker identification and analysis and speech classification, among others [15].

HMMs have certain limitations to consider in the context of language modeling:

- Insufficient realism and model simplification: HMMs make independence assumptions that might be inconsistent in real-world conditions. In actual scenarios, observations might depend on hidden states with a more intricate structure [16,17].
- The emergence of advanced models: To address this limitation, more sophisticated models have been introduced, such as the Conditional Random Fields (CRFs) and Deep Hidden Markov Models (DHMMs). CRFs cater to dependencies between observations and hidden states by considering contextual information. Meanwhile, DHMMs meld the principles of deep learning and HMMs, allowing for the modeling of intricate dependencies and providing a more precise description of linguistic structures.

While HMMs were prevalent in the past, recent advancements in deep learning, such as recurrent neural networks (RNNs) and transformer-based models like BERT and GPT, have largely supplanted HMMs in many NLP tasks. These contemporary models better capture long-term dependencies and have achieved state-of-the-art results across a broad spectrum of NLP applications. Nonetheless, understanding HMMs and their principles can provide valuable insights into the fundamentals of sequence modeling and probabilistic modeling in language processing.

3. Neural Network Models

Some of the pivotal mechanisms in the domain of NLP have been artificial neural networks (ANNs), the deployment of which has ushered in new horizons for natural language generation. The evolution and amalgamation of various neural network architectures have paved the way for addressing an array of challenges, among which is the automated extraction of intricate language structures and patterns, realized through increasingly powerful and precise language models. ANNs employ embeddings, which denote continuous word representations for aligning phrases at both the semantic and syntactic levels. Subsequent subsections will delve into various facets of ANNs' application for language modeling and their contribution to the advancement of this field.

3.1. Word Representation

3.1.1. Word2Vec

Word2Vec is a neural model designed for deriving vector representations (embeddings) of words from vast textual corpora which was introduced by Mikolov and his colleagues [18,19]. It was conceived with the intent of capturing the semantic properties of words, facilitating their representation as numerical vectors. Notably, there is no need for corpus annotation as the Word2Vec application entails training on raw, unlabeled data. In the Word2Vec model, there is an encapsulation of inter-relational nuances and similarities among words.

Distinguishing it from other neural network architectures tailored for learning word vector representations, Word2Vec proved to be highly computationally efficient. It showcased the capacity to handle vast lexicons encompassing millions or even billions of words while evading dense matrix multiplications. In collaboration with his peers, Mikolov introduced two training methodologies based on hierarchical softmax and negative sampling. To address the inherent complexity of words, word embeddings offer a solution by linking each word to a dense vector of real numbers usually falling within the range of 50 to 300 dimensions. Throughout the training process, the model fine-tunes the embedding parameters, iteratively adapting them to minimize the loss function. This function measures the difference between the predicted outcome and the ground truth, providing guidance for the optimization process.

A salient property of vectors discerned through Word2Vec models pertains to their translational characteristics, furnishing crucial linguistic and relational parallels. Specifically, Mikolov and his team's investigations illuminated that vector arithmetic can engender

qualitative analogies and affinities between words. Word2Vec retains the property of preserving semantic relationships among words in their vector representations. Semantically congruent words are proximal in the embedding space, rendering them invaluable for a plethora of tasks in natural language processing.

There exist two primary variants of the Word2Vec model: Skip-gram and Continuous Bag of Words (CBOW). Both variants employ neural networks with one or two hidden layers for training embeddings.

Continuous Bag of Words (CBOW) Model

In the CBOW model, context words are utilized to predict a target word. For every target word, a window of context words is chosen, and the neural network is trained to predict the target word based on these context words.

The CBOW architecture employs a projection layer designed to forecast the target word when presented with a context window that contains c words to both the left and right of the target word (Figure 3). At the input layer, each context word is transformed by an embedding matrix W into a dense vector of dimension k, and these resulting vectors represent context words are then averaged dimension-wise, culminating in a single k-dimensional vector. The embedding matrix W is shared across all context words. Notably, due to the aggregation of context words, their sequence does not influence the summation process, rendering this model akin to the bag-of-words approach, albeit with continuous representation. The primary objective of the CBOW model is to maximize the average logarithmic likelihood (2).

$$\frac{1}{T}\sum_{t=1-q<j<q,\ j\neq 0}^{T}\sum \log\left(p\left(w_t\middle|w_{t+j}\right)\right) \tag{2}$$

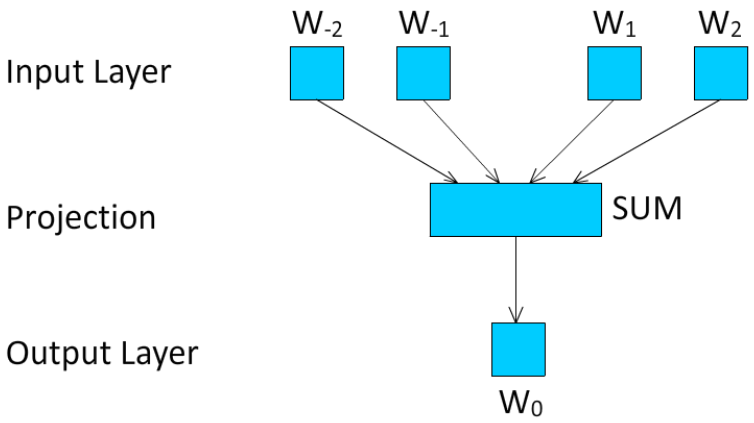

Figure 3. Continuous Bag of Words model [20].

Within this context, the variable q denotes the number of context words positioned on both sides of the target word. In the fundamental CBOW model, the resultant representation of the average vector, derived from the outputs of the projection layer, is passed through a softmax function. This softmax function makes predictions across the entire vocabulary of the corpus, facilitated by the use of backpropagation to optimize the objective of maximizing the logarithmic likelihood (3).

$$p\left(w_t\middle|w_{t+j}\right) = \frac{exp\left({v'_{w_t}}^{T} v_{w_{t+j}}\right)}{\sum_{w=1}^{V} exp\left({v'_{w_t}}^{T} v_{w_{t+j}}\right)} \tag{3}$$

Skip-Gram Model

In the Skip-gram model, the target word is used to predict its surrounding context words. For every target word, multiple examples are created in which context words are selected from a window around the target word, and the neural network is trained to predict these context words based on the target word.

While the CBOW model is trained to predict a target word based on its surrounding context words, the Skip-gram model aims to predict these surrounding context words given a specific target word. Once again, the order of the words is not considered. With a context size of c, the Skip-gram model is trained to predict c words surrounding the target word (Figure 4). The objective of the Skip-gram model is to maximize the average logarithmic likelihood (4).

$$\frac{1}{T}\sum_{t=1-q<j<q,\ j\neq 0}^{T}\sum log\left(p\left(w_{t+j}|w_t\right)\right) \tag{4}$$

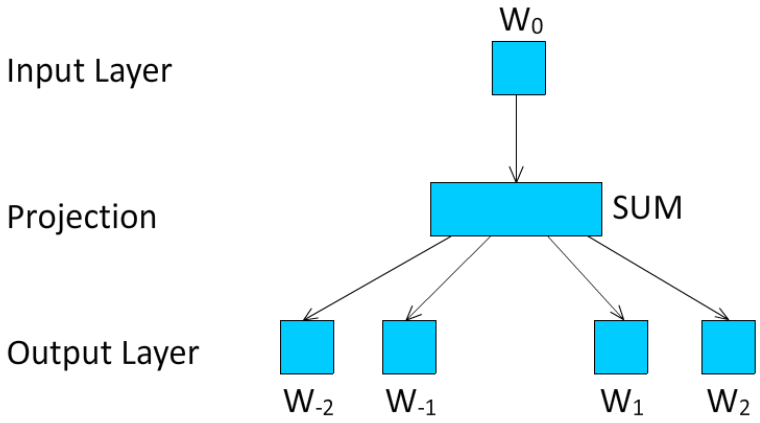

Figure 4. Skip-gram model [21].

Here, the value q represents the size of the training context. Higher values of q result in more training samples, which could potentially lead to increased accuracy at the expense of a longer training time. In its most basic form, the Skip-gram model utilizes a softmax function (5):

$$\left(w_{t+j}|w_t\right) = \frac{exp\left(v'_{w_{t+j}}{}^{T} v_{w_t}\right)}{\sum_{w=1}^{V} exp\left(v'_{w}{}^{T} v_{w_t}\right)} \tag{5}$$

Here, V denotes the number of words in the vocabulary.

3.1.2. GloVe

Global co-occurrence-based models provide an alternative to predictive methods, like the local context window approach employed by word2vec. Co-occurrence techniques are often high-dimensional and memory-intensive. Dimensionality reduction methods such as latent semantic analyses are applied, but they typically fail to capture the nuanced semantic relationships of words and are dominated by frequently occurring terms. On the other hand, predictive methods like word2vec, which rely on local context, often struggle to gather comprehensive corpus statistics.

In 2014, Pennington and his colleagues introduced a logarithmic bilinear model that merges the strengths of global co-occurrence methods with shallow window techniques [22]. They named this model GloVe, a fusion of the words "**Glo**bal" and "**Ve**ctor". GloVe is trained using a least squares optimization with a cost function (6):

$$J = \sum_{i=1,j=1}^{V} f\left(X_{ij}\right)\left(u_i^T v_j - log\left(X_{ij}\right)\right)^2 \tag{6}$$

where V represents the size of the vocabulary, X_{ij} indicates the number of times words i and j co-occur in the corpus, f is a weighting function that diminishes the influence of frequent occurrences, and u_i and v_j are the vector representations of the words. Typically, a truncated power form is used for the function f (7):

$$f(X_{ij}) = f(x) = \begin{cases} \left(\frac{X_{ij}}{X_{max}}\right)^2, & if\ X_{ij} < X_{max} \\ 1, & otherwise \end{cases} \quad (7)$$

The parameter X_{max} is determined during the training process based on the data available in the corpus. It is noteworthy that the model trains the context vectors U and word vectors V independently of each other, and the vector embeddings created by the GloVe method are formed by adding these two vector representations together ($U + V$). Analogous to word2vec, GloVe's vector embeddings are capable of expressing semantic and syntactic relationships through vector addition and subtraction operations. Furthermore, the word embeddings obtained using GloVe surpass word2vec in many natural language processing tasks, especially in scenarios in which global context is crucial, such as in named-entity recognition tasks.

GloVe has an advantage over word2vec when working with smaller datasets or when there are not enough data to accurately reflect local contextual dependencies. However, word-embedding models have some well-known limitations, like the inability to handle out-of-vocabulary words and a limited capacity to account for antonyms, polysemy (words with multiple meanings), and biases.

3.1.3. FastText

Word-embedding techniques like word2vec and GloVe offer varied vector representations for words in a vocabulary but fall short at capturing a language's internal structure; this is particularly noticeable in morphologically rich languages in which they overlook syntactic connections between words. Improving vector representations for linguistically complex languages becomes feasible by incorporating character-level information.

To enhance the quality of vector representations for morphologically rich languages, the FastText approach [23] was introduced. FastText excels in its training speed, outperforming previous methods in efficiency and handling out-of-vocabulary (OOV) words. FastText introduces character n-gram embeddings, representing words as the average of these embeddings. Unlike the Word2Vec model, which emphasizes embedding entire words, FastText delves into the word structure at the character n-gram level. Both Word2Vec and FastText leverage the CBOW and Skip-gram methods for vector computation.

FastText can handle words not in the vocabulary, generating embeddings for OOV words by embedding their character n-grams. In FastText, each word is depicted as the average vector representation of its character n-grams and the word itself. For example, considering the word "apple" with $n = 3$, its representation comprises the character n-grams <ap, app, ppl, ple, le> and <apple>. Consequently, the word embedding for "apple" is determined as the cumulative vector representation of its character n-grams and the word.

It was presented that fastText has various scoring functions to retain subword information. For a given word w, the set of n-grams present in w is denoted as $T_w \subset \{1,..., T\}$, where T represents the n-grams' dictionary size. A vector representation V is assigned to each n-gram n and context word (C_w) and vector (C_v). Consequently, the derived scoring function is defined as follows [24]:

$$F(w, C_w) = \sum_{n \in T_w} V^T C_v$$

3.1.4. Co-Occurrence Matrix

A co-occurrence matrix is a tool used in linguistic modeling to study and represent relationships between words within textual data. The underlying principle is grounded

in the assumption that words with similar meanings often appear adjacent to each other within a text. The matrix provides a way to visualize and assess how frequently word pairs co-occur in the same context [25].

To create a co-occurrence matrix from a text corpus, one starts by defining a context window around each word. Within this window, the frequency with which the given word appears alongside other words is recorded. The outcome is a square matrix in which each element [i, j] indicates the number of times words i and j appear together in proximity:

$$M[i,j] = \sum_{d=1}^{D} \sum_{k=1}^{N_d} f\left(w_i, w_j, c_k\right) \quad (8)$$

Here, D represents the number of documents in the corpus, N_d is the number of words in document d, and $f\left(w_i, w_j, c_k\right)$ is a function that equals 1 if the words w_i and w_j appear within context c_k and 0 otherwise.

The constructed co-occurrence matrix can be utilized for various tasks, such as identifying semantic relationships between words, generating word vector representations (word embeddings), and predicting the next word in a text sequence.

It is important to note that the co-occurrence matrix might be large and sparse, especially for extensive text corpora. This could necessitate additional processing and dimensionality reduction to ensure its effective use in linguistic modeling tasks.

Comparative Analysis of the Methods Word2Vec, Skip-Gram, FastText, and GloVe in the Language Modeling of Agglutinative Languages

Embedding words in agglutinative and other languages poses a complex challenge that demands thorough analysis and development. In the context of agglutinative languages, where morphological information is often expressed through affixes, challenges arise in accurately representing the diverse forms of a single word. This complicates the task of word vectorization as it requires accounting for numerous variations in morphological changes, leading to an increased dimensionality of embedded representations. To address these challenges, the development of vectorization methods is proposed, considering the specific morphological structures of languages and ensuring the effective preservation of semantic information.

In Ref. [26], the application of fastText at the encoding level for the Chinese, Japanese, and Korean languages produced superior results compared to byte-level one-hot encoding for convolutional networks in text classification tasks. Park et al. introduced an enhanced method of distributed word representations for the Korean language which decomposes words in Korean down to the so-called jamo level, which goes beyond the character level, allowing for better utilization of subword knowledge [27].

Different text categories in Turkish utilized the word2vec word vector for classification [28]. After extracting word vectors, each text was represented as the average vector of its constituent words. As a result, the word2vec algorithm outperformed TF-IDF in classifying Turkish documents. Moreover, word2vec demonstrated excellent results in a sentiment analysis of Turkish texts [29]. To address the challenge of detecting semantic and syntactic patterns, two approaches were implemented, namely, hypernyms in Turkish based on biased classification embedding and semantic projection [30]. These similar models significantly enhance system quality compared to baseline algorithms, provided that the corpus includes several instances of hypernymy. Metin Bilgin compared CBoW and SkipGram for sentiment analysis and showed that CBoW performs better in this task on a Turkish dataset than SkipGram. The same author and his colleagues [31] conducted experiments using the Distributed Memory (DM) and Distributed Bag of Words (DBoW) algorithms of Doc2Vec, revealing that an insufficient volume of data negatively affects sentiment analysis results.

In Ref. [32], the embedding of words was expanded, leveraging subword information for the Arabic language in the domain of machine translation tasks. Nehad M. Abdel

Rahman Ibrahim introduced a model enhancing both the accuracy and efficiency of the clustering of Arabic articles, achieved through the application of word-embedding-based clustering and discretization techniques. Abdulateef et al. [33] proposed an unsupervised score-based method that amalgamated four distinct techniques: the vector space model, continuous bag of words (CBOW), clustering, and a statistically based method. This method surpassed other methods in the extraction of key sentences from Arabic text. In Ref. [34], the authors compared the Word2Vec and Lemma2Vec models in tasks pertaining to the disambiguation of Arabic words in context. They achieved this without resorting to external resources or any specific contextual/sense embedding model. Their findings revealed that both Lemma2Vec and Word2Vec performed commendably across various sentence characteristics.

Khusainova et al. [35] crafted three datasets for the Tatar language and compared the performance of Skip-gram, FastText, and GloVe models on datasets of both the Tatar and English languages. Models trained on English datasets generally outperformed their Tatar counterparts, with differences particularly notable in semantic and syntactic categories. The study suggests that the higher effectiveness of English models may be related to the nature of the questions posed and discrepancies in the training corpora. However, some correlations between Tatar and English models are observed in specific categories. The lack of large parallel corpora for both languages is acknowledged as a limitation in the comparison.

In Ref. [36], the word2vec and GloVe methods were implemented for semantic, syntactic, and capital evaluation tasks based on an Azerbaijani dataset. The results highlighted that enhancing the dimensionality of continuous vectors and training data amplifies the effectiveness of word vectors for both architectures. Conversely, it was observed that compared to word2vec, GloVe produced more accurate word embeddings for semantic evaluation tasks in Azerbaijani. In contrast, for syntactic evaluation tasks, the inverse was true.

Assessment of the Results from Applying Methods Like Word2Vec, Skip-Gram, FastText, and GloVe for the Language Modeling of Agglutinative Languages

In the Chinese, Japanese, and Korean languages, the FastText method offers significant advantages in text classification tasks over byte-level one-hot encoding. This underscores the importance of effective text representation, especially for languages that employ an ideographic writing system.

The Turkish language, with its unique morphology and structure, demonstrates that methods like Word2Vec are successfully employed across various scenarios, including text classification and sentiment analysis. However, the specific choice in word modeling approach, for instance, opting between CBOW and SkipGram or employing Doc2Vec, may be contingent on the task at hand and data availability. This highlights the importance of the experimental validation of diverse methods to optimize performance across different NLP applications.

Furthermore, when exploring semantic relations, such as hypernymy in the Turkish language, word-embedding-based methods, augmented by semantic projection and biased classification, can provide deeper and more insightful representations of relationships between words.

For the Arabic language, the focus is placed on morphological word embedding, considering its intricate morphology. There is also an emphasis on employing various methods for clustering and key sentence extraction to enhance content analysis and automated text annotation efficacy. Specifically, the use of subwords and unsupervised methods can be particularly advantageous for machine translation and summarization, respectively.

Regarding the Tatar language, research emphasizes the importance of word embedding method selection, taking into account the language's specific linguistic features and contextual situation. Comparing various word embedding methods, such as Skip-gram, FastText, and GloVe, reveals that efficacy can vary significantly depending on the language's unique characteristics.

As for the Azerbaijani language, the results are promising using Word2Vec and GloVe for semantic and syntactic evaluations, with embedding dimensionality and data volume emerging as crucial factors for optimizing the quality of embeddings.

3.2. Pretrained Language Models

The transformer is a sophisticated deep learning architecture employing an attention mechanism to analyze text sequences. In contrast to conventional models reliant on recurrent neural networks, the transformer does not rely on sequential connections, excelling at capturing extensive dependencies in text. It comprises two pivotal elements: attention and transformer blocks. The attention mechanism empowers the model to assign varying weights to words within a sequence, directing its focus to the most pivotal portions of the text. Transformer blocks are layers applying nonlinear transformations to input representations, facilitating the acquisition of language patterns and structures [37].

Transformers, forming the foundation for large language models in natural language processing (NLP), have revolutionized the field by endowing models with unparalleled proficiency in understanding and generating text consistently. Transformers exhibit remarkable efficacy across various NLP tasks for several reasons. Firstly, their high parallelizability enables the simultaneous processing of multiple parts of a sequence, accelerating both training and inference.

Moreover, transformers adeptly capture long-term dependencies in text, enhancing the comprehension of overall context and fostering the generation of more cohesive text. Their flexibility and scalability render them easily adaptable to diverse tasks and domains. This section delves into large language models pre-trained with a transformer in NLP, elucidating their operational mechanisms and discussing their merits and drawbacks.

3.2.1. BERT

In 2018, Google introduced BERT, which continues to be a frequently used modeling architecture due to its high efficacy in various NLP tasks. This model is a pre-trained one based on unlabeled textual data from open sources like BookCorpus and Wikipedia. Rooted in the transformer framework, its unique bidirectional nature allows the model to capture intricate dependencies between words. By fine-tuning the model on labeled data from specific tasks, BERT achieves competitive performance.

In its structure, BERT employs the encoder component of the transformer architecture. There are both base and large versions of the model, differing in the number of layers and the number of self-attention heads they have, with 12 and 24 layers, respectively. The total count of adjustable parameters reaches several hundred million, necessitating significant computational resources. Although most of the high achievements initially came from using the 24-layer BERT, issues often arose due to the limited computational power of machines. To address this challenge, the ELECTRA model was proposed. ELECTRA is designed to be more efficient than BERT, optimizing both training time and model size while retaining, or in some cases even surpassing, the performance of larger models like BERT. It rethinks the pre-training process and utilizes a replaced token detection task, distinguishing it from BERT's masked language model approach [38].

Like GPT, BERT uses contextual embedding. For each word, a fixed number of 768-dimensional embedding vectors are determined, serving as the model's input sequences. BERT outputs a vector with 768 parameters for each word/phrase based on its internal calculations. These output sequences then become embeddings, representing the sentence sequence in the context of the data. In simpler terms, an initial input to BERT has a basic vector layer, but after processing through BERT, a contextual vector is produced using the entire word vector.

The model comprises input formatting and embedding extraction parts (Figure 5). Special tokens like [CLS] and [SEP] are used to denote the beginning and end of a sentence, respectively. [PAD] is used for padding, and [UNK] indicates an unknown word. The embedding layers include token embeddings, segment embeddings, and position embeddings.

For token embeddings in BERT, the WordPiece method is commonly used for tokenization. Segment embedding examines the semantic relationship of up to two sentences. Position embedding retains the position information of a word to create a word sequence.

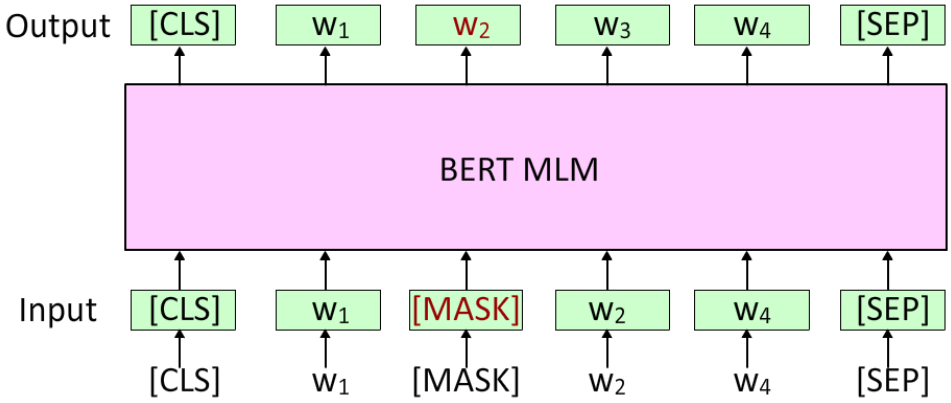

Figure 5. BERT masked language model example.

Even though BERT uses bidirectional encoding, it operates quite differently from recurrent neural networks. This is because BERT maps the obtained position embeddings of each token to segment embeddings to represent a word's property and its location. In other words, all that is needed is to format the input text using the embedding layer and let the encoder provide the appropriate output.

BERT's Pre-Training Algorithm

There are two main approaches, with one being the Masked Language Model (MLM). The MLM is trained to predict words that are randomly masked in a sentence. For instance, consider the sentence: "I [MASK] in Almaty". Here, the word "live" has been masked. The model then tries to predict this masked word, and its training is optimized based on the associated loss. This method enables the model to learn to capture semantic relationships and the context within a sentence. This specific technique is central to language modeling in BERT.

Next sentence prediction (NSP): to further train the model, pairs of sentences are presented, and the model must determine whether the second sentence is a continuation of the first (Figure 6).

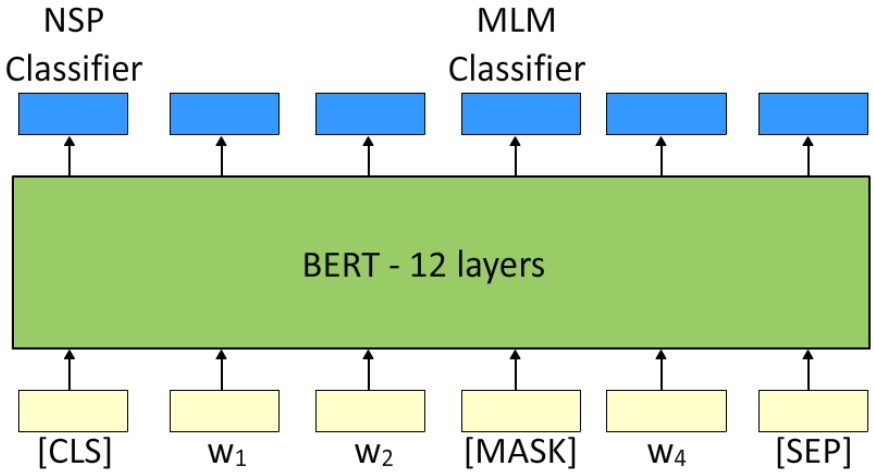

Figure 6. Example of a next sentence prediction model.

One of BERT's main advantages is its multilingual capabilities. It is trained on a vast volume of polysemous texts, allowing it to work with data in multiple languages.

Additionally, its bidirectional nature, which considers not only the preceding context in a sentence but also the subsequent one, enables the model to understand semantic relationships between phrases. Once pre-trained, BERT can be fine-tuned for specific tasks such as named-entity recognition, text classification, text generation, and more.

Since the introduction of BERT, several improvements and variations have been made to further optimize its performance and applicability. Some of these enhancements are provided below (Table 1).

Table 1. Brief comparison of BERT models and variants based on various characteristics.

Model	Key Features	Features of Pretraining	Parameters
BERT	Based on the Transformer architecture. Uses masked language models (MLM) for pre-training.	Masking 15% of words in a sentence. Predicting the masked words.	110 M (BERT-base)
RoBERTa	Optimized BERT. Removes NSP.	More data and batches during pre-training. Reduced use of masking. Uses MLM.	340 M (BERT-large)
ALBERT	Factorization of weight parameters. No NSP.		125 M (RoBERTa-base)
XLNet	Uses Transformer-XL. Combines BERT and GPT.	Uses permutation language models (PLM).	355 M (RoBERTa-large)
DeBERTa	Introduction of enriched attention matrices. Dynamic weighting of different attention layers.	MLM	12 M (ALBERT-base)
DistilBERT	Compressed version of BERT—40% fewer parameters while maintaining 97% performance.	MLM	18 M (ALBERT-large)
Electra	Replacement of pretraining with discriminator and generator.	Uses Replaced Token Detection instead of MLM.	110 M (XLNet-base)

FinBERT (Financial Sentiment Analysis with Pre-trained Language Models) is a variation of BERT specifically trained for processing financial data to perform sentiment analysis [39]. It is adept at understanding specialized "investment jargon" and precisely gauging the emotional tone of a text. It is trained using text from financial news services and data from the FiQA dataset.

RoBERTa (A Robustly Optimized BERT Pretraining Approach) is an enhanced version of BERT developed by Facebook AI. While built on BERT's foundations, RoBERTa incorporates a more refined pretraining process, such as longer sequences, larger datasets, and dynamic training parameter tuning. It employs an advanced methodology, significantly more computational resources, and substantially more data. Additionally, RoBERTa introduces dynamic masking that changes during training [40].

ALBERT (A Lite BERT for Self-supervised Learning of Language Representations) is a model developed by Google Research which focuses on optimizing BERT's parameter usage. It utilizes a "factorized embedding parameterization" architecture to reduce the total number of network parameters, allowing for more scalable models with constrained resources [41].

XLNet (Generalized Autoregressive Pretraining for Language Understanding) melds concepts from BERT and Transformer-XL. Unlike BERT, which predicts masked words, XLNet considers the complete sentence context, aiding the model in capturing long-term dependencies [42].

DeBERTa (Decoding-enhanced BERT) is built upon RoBERTa with added decoding enhancements, dynamic masking, and a more intricate attention mechanism. In tests, it exhibited superiority over human benchmarks in the SuperGLUE challenge [43].

DistilBERT (Distillation of BERT Model for Sequence-to-Sequence Tasks) is a streamlined version of BERT created using the knowledge distillation method. It learns to transfer the knowledge from a more complex model, like BERT, into a more compact form, making it faster and more efficient in operation [44].

Electra (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) is a model designed by Google Research. It introduces an efficient pretraining method in which random words are replaced with words predicted by the model. This enables a more effective utilization of supplementary resources. Despite this, DistilBERT

remains a more resource-conservative version, even though it still demands substantial computational resources [45].

These advancements show how researchers consistently aim to refine and expand concepts associated with BERT, targeting higher performance and versatility in various natural language processing tasks.

After its preliminary training, the BERT model can undergo further training for specific tasks through fine-tuning. This enables BERT to grasp more intricate and higher-level semantic nuances in the text, making it a potent tool for diverse NLP tasks.

The BERT model boasts numerous advantages. However, like all technologies, it also has its downsides:

- BERT, especially its larger versions like BERT Large, encompasses a considerable number of parameters. This leads to high computational demands during training and deployment, presenting challenges for systems with restricted computational capacities. Additionally, BERT has a set maximum sequence length it can handle, which can be an issue for texts with long contexts or documents.
- BERT processes words and symbols alike, potentially leading to unnecessary processing of punctuation and other non-essential text elements. Moreover, BERT does not consider the internal order of characters within words (morphological structure), which can be a limitation for languages with rich morphology.
- Effectively training BERT requires vast and varied data volumes. This can be challenging for languages with limited available data, such as agglutinative languages. BERT also views context within a specific window, possibly overlooking more distant dependencies between words.
- BERT might capture noisy data during pre-training, which could affect its generalization ability on new data.

It is important to note that many of these drawbacks can be mitigated or overcome with improved model versions, additional fine-tuning and adaptation methods, and the integration of BERT with other techniques and architectures.

Enhancements and Modifications for Agglutinative Languages

In Ref. [46], two models, ARBERT and MARBERT, were introduced for data classification tasks focusing on the Arabic language. Both ARBERT and MARBERT utilized the same architecture as BERTBase; however, MARBERT did not incorporate NSP (Next Sentence Prediction) and used the WordPiece tokenizer. The dataset was expanded with Twitter data for pre-training the new MARBERT model. Fouzi Harrag and his team introduced a combined ARABERT and GPT2 model using transfer learning to detect text produced by various AI bots in Arabic. For generating fake sentences, GPT2-Small-Arabic was used. The proposed model was found to significantly outperform RNN-based models.

In Ref. [47], a BERT-BGRU model was introduced for the task of recognizing Arabic-named entities. Here, BERT combined multiple attention layers, and fine-tuning BERT as a base input representation model effectively reduced the training time for the subsequent deep-neuralnetwork-based model.

Bozuyla et al. [48] implemented a Turkish specialized transformer, BerTURK, to determine the authenticity of Turkish COVID-19 news on social networks. The model's efficiency was achieved through a minimal fine-tuning strategy, boasting a prediction accuracy of 98.5%.

AL-Qurishi et al. [49] developed AraLegal-BERT, a bidirectional encoder transformer trained on legal documents. It followed the same procedure as in AraBERT by Antoun et al. (2020) [50].

For the development of the AraLegal-BERT model, a lexicon of around 64,000 words was used which was constructed based on a subword segmentation approach to divide all tokens into prefixes, suffixes, etc. [51]. Comparative analyses showed that a model pre-trained based on a specific domain performed better than the standard BERT models.

In the 2022 [52], the BERT model was enhanced by feeding it with two sentences: the first one containing word expressing sentiment and the second one providing information about the aspect, essentially assisting in aspect-based sentiment analysis. Experiments conducted on three Arabic datasets reached a peak accuracy of 89.51% for hotel reviews.

Alammary, in 2022, presented a systematic review of BERT models for Arabic-language text classification, including models like Multilingual BERT, AraBERT, MARBERT, ArabicBERT, ARBERT, XLM-RoBERTa, QARiB, GigaBERT, and ArabALBERT. Based on the analysis, the author concluded that MARBERT, QARiB, ARBERT, AraBERT, and ArabicBERT are the most effective models. A plausible explanation for this finding is the size of the pre-training corpora used for training these models [53].

As evident from the review, research on BERT-like models for the Arabic language is advancing and increasingly addressing diverse NLP tasks.

Acikalin et al. [54] introduced two methods in the context of sentiment analysis: (a) fine-tuning the BERT model and (b) applying the base BERT model after translating Turkish texts into English. Their findings indicated that (a) using the BERT model for sentiment analyses yields better results than other methods, and (b) English-specific BERT models outperform the multilingual BERT model when there are ample training data available.

Mutlu et al. [55] developed a manually annotated dataset of Turkish tweets containing sentences of varying sentiment and polarity. They implemented base models and adapted models named TBERTmarked, TBERTmarked-TS, and T-BERTmarked-MP. As a result, the proposed models demonstrated superior performance in a target sentiment analysis.

In Ref. [56], a language model capable of understanding linguistic phenomena of the Korean language was developed and trained on smaller datasets. By leveraging linguistic features and part-of-speech (PoS) tagging during pre-training, models based on transformer architecture and bidirectional language representations proved effective not just for Korean but also for other agglutinative languages.

In Ref. [57], a clinical BERT model tailored to Japanese clinical narratives was introduced. The model was pre-trained on a vast domain-specific dataset, facilitating accurate phrase generation in complex tasks like question-answering or recognizing causal relationships using genuine clinical texts.

In Ref. [58], an enhanced version of the multilingual BERT, termed E-MBERT, was proposed. It integrates a target language by expanding the vocabulary, encoders, and decoders and then continues pre-training. This approach can be applied to any language with limited datasets not supported by multilingual BERT, especially for named-entity recognition tasks. Feng et al. [59] developed a language-agnostic BERT sentence-embedding model supporting 109 languages. Various methods were examined, including masked language modeling (MLM), translation language modeling (TLM), dual-encoder translation ranking, and additive margin softmax. The introduction of a pre-trained multilingual language model significantly reduces the need for parallel training data, achieving an 80% reduction while attaining 83.7% bi-text retrieval accuracy over 112 languages on Tatoeba. The combined methods outperform previous benchmarks and remain competitive in monolingual transfer learning. Additionally, parallel data mined from CommonCrawl using their model proved effective for training NMT models for English–Chinese and English–German.

Mansurov et al. [60] introduced a BERT-based model, named UzBERT, pre-trained on news websites for the Uzbek language. The proposed model was trained on a small corpus of approximately 140 million words. Based on experiments, its accuracy on the masked language model task was significantly higher than that of the multilingual BERT. UzBERT has a smaller vocabulary size compared to mBERT, which means it demands fewer resources for fine-tuning. This characteristic is theorized to enhance its ability to capture the subtleties of the language, as it was exclusively trained on Uzbek texts.

Meanwhile, on the global stage research is pivoting toward multilingual and universal strategies, notably the development of E-MBERT and the crafting of language models that support scores of different languages, providing remarkable performance through

cross-lingual embeddings. Thus, the adaptability and modifiability of the BERT architecture render it promising for text-processing tasks across various languages and domains. Each of the above-mentioned research endeavors underscores the importance of specialized pre-training and language-specific adaptation to enhance model efficacy.

3.2.2. GPT

GPT (Generative Pre-trained Transformer) is a series of neural language models developed by OpenAI. GPT basically uses the goal of training a language model with an autoregressive approach, predicting the next word in a sequence based on the context of the preceding words. Unlike BERT, GPT does not use MLM or NSP tasks. The models utilize the transformer architecture but only employ the decoder blocks. One of GPT's defining features is its ability to generate texts as opposed to merely predicting the likelihood of subsequent words. The original GPT-1 model was trained on the BooksCorpus dataset (Table 2). The BooksCorpus contained about 7000 unpublished books, aiding in training the language model on unseen data. GPT employed a 12-layer decoder, utilizing only the transformer structure with masked self-attention for language model training. The masking ensured that the language model could not access subsequent words to the right of the current word during training.

Model Training Details:

- A Byte-Pair Encoding (BPE) vocabulary with 40,000 merges was used.
- The model utilized a 768-dimensional state to encode tokens into word embeddings. Positional embeddings were also learned during training.
- A 12-layer model with 12 attention heads per self-attention layer was employed.
- A 3072-dimensional state was used for the positional feedforward layer.
- The Adam optimizer was used with a learning rate of 2.5×10^{-4}.
- Attention, residual, and embedding dropout were applied for regularization, with a dropout rate of 0.1. A modified L2 regularization was also applied to weights without bias.
- The GELU activation function was used.
- The model was trained for 100 epochs on mini-batches of a size of 64 and a sequence length of 512. In total, the model had 117 million parameters.

One of the most significant achievements of this model is its impressive zero-shot performance across various tasks. Studies have shown that the model achieved zero-shot capabilities in various NLP tasks such as question-answering, coreference resolution, and sentiment analysis, all thanks to its pre-training. GPT has proven that a language model serves as an effective pre-training objective that can aid the model in generalizing well. The architecture facilitates transfer learning and can execute a myriad of NLP tasks with minimal fine-tuning. This model demonstrated the power of generative pre-training and paved the way for subsequent models that could further harness this potential with larger datasets and more parameters.

Since the debut of the original GPT model, several enhancements and variations have been introduced to optimize its performance and capabilities. Some of the improvements include the following:

1. GPT-2 is the successor of the original GPT model. GPT-2 was trained on a significantly larger dataset and boasts a higher capacity (1.5 billion parameters). It showcased impressive results in text generation. However, due to concerns about its potential misuse, its release was restricted. To compile a comprehensive and high-quality dataset, the authors scraped the Reddit platform and extracted data from outgoing links in highly rated articles. The resultant dataset, named WebText, consisted of 40 GB of textual data from over 8 million documents. This dataset was considerably larger than the Book Corpus used to train the original GPT model. All Wikipedia articles were excluded from WebText since many test sets incorporate Wikipedia articles.

GPT-2 had 1.5 billion parameters; this is tenfold the number in GPT-1, which had 117 million parameters. The primary differences from GPT-1 included the following:

- GPT-2 featured 48 layers and utilized 1600-dimensional vectors for word embeddings.

- It employed a larger vocabulary of 50,257 tokens.
- A more substantial batch size of 512 and a larger context window of 1024 tokens were used.
- Layer normalization was moved to the input of each sub-block, and additional layer normalization was added after the final self-attention block [61].

GPT-2 demonstrated that training on larger datasets and having a higher number of parameters enhanced the language model's ability to comprehend tasks, surpassing the current state of the art in many zero-shot conditions. Additionally, the decrease in the perplexity of language models showed no signs of plateauing and continued to drop as the number of parameters increased.

An intriguing capability of GPT-2 is its zero-shot task transfer. Zero-shot learning is a specific instance of task transfer in which no examples are provided and the model understands the task based solely on the provided instruction. Instead of reshuffling sequences, as was carried out with GPT-1 for fine-tuning, inputs in GPT-2 were provided in a format expecting the model to grasp the task's nature and supply answers. This was designed to mimic behavior in zero-shot task transfer scenarios. For instance, for an English-to-French translation task, the model was presented with an English sentence followed by the word "French" and a hint (:). The model was expected to recognize this as a translation task and provide the French equivalent of the English sentence.

2. GPT-3 boasts a considerable number of parameters, standing at 175 billion, making it one of the most extensive neural models to date. It was trained on a combination of five distinct corpora, each assigned a specific weight. High-quality datasets were prioritized, with the model being trained on them for more than a single epoch. The datasets used include Common Crawl, WebText2, Books1, Books2, and Wikipedia.

The architecture of GPT-3 remains consistent with GPT-2, featuring 96 layers, each containing 96 attention heads. The word-embedding count was increased to 12,888, and the context window size was expanded to 2048 tokens. The Adam optimizer was utilized with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\varepsilon = 10^{-8}$. Additionally, alternating dense and locally striped sparse attention patterns were implemented.

GPT-3 was evaluated on an array of language models and NLP datasets. It performed admirably on NLP tasks such as closed-book question-answering, schema resolution, and translation, often outpacing the most contemporary or comparably fine-tuned models.

However, certain shortcomings of this version were identified:

- Its unidirectional nature struggles with tasks like natural language inference, gap-filling, and some reading comprehension tasks and loses coherence when framing long sentences, often repeating text sequences [62].
- The model weighs each token equally and lacks task-awareness or goal-directed token forecasting. To mitigate this flaw, suggestions included enhancing training objectives, employing reinforcement learning for model fine-tuning, and incorporating additional modalities.
- The model's vast architecture complicates inferencing and offers reduced language and result interpretability.
- Like other model versions, it can generate phishing emails, spam, and texts with racial or ethnic biases.

3. The GPT-3.5 model comes in three variants with parameters of 1.3B, 6B, and 175B. While this model is trained based on GPT-3 data, it undergoes an additional fine-tuning process with human feedback to adjust the model's behavior. The model was trained on data up to the fourth quarter of 2021 and serves as the foundation for the ChatGPT dialogue bot.

4. GPT-4 (March 2023) represents the evolutionary stage in the GPT series, surpassing its predecessor, GPT-3.5, by enhancing capabilities like processing longer text segments and images. The essence of GPT-4's training encompasses transferring the rich knowledge acquired during preliminary training to diverse new tasks. Relying on large-scale pre-training, this approach enables the model to grasp linguistic patterns and context, which can then be fine-tuned for optimal performance in specific tasks. Memory and attention mechanisms play a crucial role, ensuring GPT-4's effective learning even with limited

data. Enhanced linguistic capabilities and multilingual support augment its proficiency in producing higher-quality and coherent responses in various interactions.

Table 2. Evolution of the GPT model [63].

GPT Versions	Year of Publication	Parameters	Architecture Improvements	Training Dataset
GPT-1	2018	117 million	12 decoder layers and 768 hidden layers, including softmax and linear layers	BookCorpus
GPT-2	2019	1.5 billion	48 decoder layers, 1600 hidden layers, and the normalization layer was extended	WebText
GPT-3	2020	175 billion	96 decoder layers and 12,288 hidden layers	Common Crawl
GPT-3.5	First version: 2021	175 billion	WebText2	
GPT-4	Last version: March 2023	Undefined	Combines Transformer and CNN+RNN models; it can process and analyze images	Undefined

Other improvements and modifications of the GPT model for agglutinative languages are described below.

In recent research efforts, various adaptations and evaluations of large language models, such as GPT-3, for non-English languages have been undertaken. This section provides an overview of studies that explore the performance, challenges, and applications of GPT-based models in linguistic contexts beyond English.

1. For the Arabic Language:

Antoun et al. [64] developed the first refined model for Arabic language generation, AraGPT2, trained from scratch on a large corpus of Arabic Internet texts and news articles. Their larger model, ARAGPT2-MEGA, which has 1.46 billion parameters, has shown success in various tasks including synthetic news generation and zero-shot question-answering. Regarding text generation, their best model achieves a perplexity of 29.8 on held-out Wikipedia articles. A human evaluation study showed that AraGPT2-MEGA significantly excels in creating news articles that are hard to differentiate from those written by humans. In [65], researchers evaluated the efficacy of 24 models for detecting sentiment and sarcasm in Arabic texts. The findings indicated that the AraGPT2 variants show relatively lower performance, especially when compared to BERT-based models, suggesting a potential unsuitability for classification tasks, i.e., revealing potential limitations of the GPT-2 architecture for Arabic in text-classification tasks.

Nagoudi et al. [66] created a set of five autoregressive language models named JASMINE, based on GPT-3 for the Arabic language, with parameters ranging from 300 million to 13 billion. The models were trained on approximately 400 GB of texts representing various domains and dialects of the Arabic language. JASMINE underwent rigorous testing on various natural language processing (NLP) tasks across different training regimes, and the results showed that the models can generate text indistinguishable from that written by a human.

In Ref. [67], an ensemble learning model for Arabic sentiment analysis was proposed which incorporates a bidirectional long short-term memory (BiLSTM) model and a Generative Pre-trained Transformer (GPT) model. Experiments were conducted that involved separate applications of BiLSTM and GPT and a comparison with the ensemble model. The proposed ensemble model demonstrated an accuracy improvement of approximately 7% compared to individually trained models, indicating that model accuracy can be enhanced when combining different architecture".

2. For the Korean Language:

Kim et al. [68] emphasized the challenges and opportunities associated with using large-scale language models, like GPT-3, especially in the context of non-English data such

as the Korean language, and various aspects of model performance. The authors introduced HyperCLOVA, a Korean version of the 82B GPT-3, trained on a Korean-centric corpus of 560B tokens. It was enhanced through "kenization" (a play on tokenization), adapted for the Korean language, with a customized training configuration. Thus, the research underscores the importance of adapting large language models to specific languages and cultures, supporting this with results obtained from the Korean version of GPT-3, HyperCLOVA. The study also suggests approaches to optimize AI design and integration processes in application development without deep technical expertise. Many studies emphasize that large language models (LLMs) not only learn language patterns during training but also the social biases present in the data they are trained on, which can pose risks when they are used. Due to linguistic and cultural characteristics in South Korea, existing bias research and resources are challenging to apply, emphasizing the need for localized datasets on this issue. To address this, KOSBI—a new dataset on social biases in the Korean language—was introduced. It includes 34,000 pairs of contexts and sentences, covering 72 demographic groups in 15 categories [69]. Experiments showed that using moderation based on filtering can reduce social biases in content generated by models, such as HyperCLOVA and GPT-3, by an average of 16.47%.

In Ref. [70], researchers introduced another dataset, "Sensitive Questions and Acceptable Answers" (SQUARE), which comprises 49,000 sensitive questions and 88,000 answers (of which 42,000 are acceptable and 46,000 are unacceptable) in the Korean language. This dataset was created using HyperCLOVA technology and is based on real news headlines. Experimental data indicate that using SQUARE enhances the ability of models, such as HyperCLOVA and GPT-3, to generate acceptable answers, underscoring the practical utility of the new dataset. However, comparative experiments applying both corpora to address set tasks have not yet been investigated.

3. For Japanese:

Kasai et al. [71] assessed the performance of large language model (LLM) APIs, including ChatGPT, GPT-3, and GPT-4, in relation to Japanese national medical licensing exams over the past five years. The study provides the IGAKU QA test dataset. The results showed that GPT-4 outperforms the other models, successfully passing all exams during the specified period. Despite this, the study identified critical flaws in existing LLM APIs. Furthermore, the analysis revealed high costs for the API and a reduced maximum context size for the Japanese language due to the peculiarities of tokenizing non-Latin scripts.

Lai et al. [72] evaluated the multilingual capabilities of ChatGPT and other large language models (LLMs) in performing various natural language processing (NLP) tasks, such as part-of-speech tagging, named-entity recognition, relation extraction, natural language text generation, question-answering, commonsense reasoning, and text summarization. The evaluation was conducted in 37 languages with varying levels of available resources. The results show that ChatGPT exhibited diminished efficiency in zero-shot learning across different languages in the considered NLP tasks, underscoring the importance of developing models specific to particular tasks and languages to enhance performance.

3.2.3. Large Language Model Meta AI (LLAMA)

A base model refers to a model trained on a large volume of unlabeled training data in an unsupervised manner. This state of the model is leveraged for various NLP tasks without the need for retraining or with minimal parameter tuning.

Many models have emphasized scaling, believing that more parameters lead to higher-quality outcomes. However, the costliness of the training process does not always resonate well with every researcher. Studies, such as [73], have highlighted in their research that high performance can be achieved by training models based on quality-rich and voluminous data.

This year, Meta released its large language model, LLaMa, with up to 65 billion parameters, which, like its predecessors, employs the transformer architecture. The primary focus was on expanding the training data and reducing the weight count, ensuring that it does not exert substantial computational pressure during data training. The BPE algorithm was

used for data tokenization. Ultimately, the model was trained based on 1.4 trillion tokens gathered solely from open sources like Wikipedia, ArXiv, and GitHub [74]. The model provides a user-friendly interface for conducting experiments, encompassing automatic parameter optimization and an auto-stop feature when deemed necessary. It incorporates a robust tool for analyzing and visualizing experiment outcomes, assisting researchers in understanding why certain models demonstrate greater efficiency than others. LLaMA also has been adapted and refined for medical applications, as evidenced by a model called ChatDoctor which utilized LLaMA and was fine-tuned using a large dataset of patient–doctor dialogues to achieve better performance in medical consultations [75].

Initially, to gain access to the model weights, one had to submit an application for academic research or other specific purposes. Later, all the weights were posted on the imageboard 4chan, leading to their active dissemination among researchers and developers. With 13 million parameters, LLaMa outperformed GPT-3 on various test tasks using fewer tunable parameters. The following improvements were made to the transformer model: (1) A normalization layer was incorporated at every sub-layer of the model, enhancing the stability of the training process. In practice, the root mean square normalization function is applied to normalize the input signal of each transformer layer. This approach revisits invariance properties and the capability of adapting the implicit learning rate. (2) The SwiGLU activation function was used instead of the standard ReLU activation function. This modification noticeably enhances training efficiency. (3) The use of rotary embeddings borrowed from the GPT-Neo-X project. These embeddings are added at every layer of the network, aiding in accounting for positions within the text.

A refined version, LLaMa 2, was recently released. One of the major changes includes an increased context length and attention to Grouped Query Attention (GQA). The context window size was expanded from 2048 to 4096 tokens, enabling the model to process and generate more extensive information. This is particularly beneficial when dealing with long documents, chat histories, and summarization tasks. Moreover, certain modifications were made to the attention mechanism to handle the scale of contextual data. Various attention paradigms were compared within the study, including using multiple heads, a multi-query format with a single key-value projection, and a GQA format incorporating eight key-value projections. This allowed for more efficient handling of vast amounts of data and context.

LLaMa 2 introduced a significantly updated and expanded training dataset. The new dataset is anticipated to be 40% larger in volume compared to the data used for training the initial LLaMa model. In total, the researchers utilized over 2 trillion tokens for training. This indicates that the model was trained on a much larger volume of information, which might enhance its performance and generalization capability.

Researchers noted that when working with long contexts or large batch sizes, the complexity of the attention mechanism in the original MHA format significantly increases. Thus, employing multiple key-value projections and Grouped Query Attention (GQA) assists in managing this complexity and provides more efficient processing of contextual data.

3.2.4. StableLM

A base model refers to a model trained on a large volume of unlabeled training data in an unsupervised manner. This state of the model is leveraged for various NLP tasks without the need for retraining or with minimal parameter tuning.

StableLM was trained on unlabeled data from Wikipedia, YouTube, and PubMed, all part of The Pile dataset, amounting to 1.5 trillion tokens. The initial version of the model contains between 3 and 7 billion parameters, with a high likelihood of scaling up to 65 billion. The open-source base model facilitates customization for various tasks aligning with scientific and other objectives. However, it is worth noting that this model supports a limited set of languages, which constrains its application breadth. StableLM is predicated on the NeoX transformer model. A significant advantage of StableLM is its ability to maintain a consistent internal state, thereby ensuring lexical and grammatical coherence in the generated text. This feature empowers StableLM to produce extensively long and

verbose texts, positioning it as a competitive tool for automatic text generation. The stability of the internal state and the capability to produce high-quality texts with minimal training overhead are the primary assets of this model [76].

The training process of StableLM is executed through a masked language modeling task which entails presenting a sequence of text data with specific masked tokens. The subsequent step involves the model attempting to restore these masked tokens. This training methodology fosters the development of StableLM's capability to generate text that adheres to grammatical norms and holds semantic relevance.

3.2.5. Bard

Bard is one of the modern deep learning models released by Google in March 2023. Notably, Bard addresses the issue of reducing computational power requirements by creating an efficient AI model. Through reducing the number of parameters during training, the model becomes faster, cost-effective, and energy efficient. Its architecture also boasts greater scalability, enabling easy adaptation to a myriad of NLP tasks and applications [77].

Based on the Transformer-XL architecture, Bard represents a large language model (LLM) designed for processing extensive text sequences. This architecture, an extension of the original transformer, excels in language modeling tasks by adapting to extended text sequences.

Comprising two main components, an encoder and a decoder, Bard operates through a stack of internal attention layers. Self-attention facilitates the model's capacity to capture long-range word relations, an essential trait for generating coherent natural-language text. Bard also integrates a positional encoding layer, assisting the model in understanding the positions of words within the input text. Additionally, there is a residual connection layer, which prevents overfitting while maintaining model generalization, and a normalization layer, stabilizing the training dynamics for sequential learning. The model was trained using a comprehensive dataset encompassing various textual forms, such as books, articles, and code. Furthermore, by employing an MLM objective, the model predicts occluded words in a given text sequence.

Unlike the ChatGPT chatbot, Bard, when generating responses, leverages current online information by accessing the Internet, offering more accurate and specific answers to queries. Other distinctions from ChatGPT are Bard's multilingual capabilities, which enable it to understand natural language in various linguistic contexts, catering to a broader audience. Furthermore, Bard was implemented based on the LaMDA language model (Language Model for Dialogue Applications) [78]. Following the advent of the Bard chatbot, Google opted to integrate it into its search engine, enhancing query processing to yield superior results. This enhancement arises from the model's ability to analyze the context and semantics of a query, which refines the precision of the results. This proves particularly beneficial for processing niche queries that traditional search engines might struggle to satisfactorily address.

Experimental outcomes have demonstrated that the Bard model surpasses the GPT model in recognizing user intents, as well as in sentiment analysis and text classification.

However, the Bard model does have its shortcomings. Despite its capacity to generate seemingly coherent text, a more profound understanding of factual accuracy, originality in expression, and the ability to capture the intentions and objectives of the text are essential for its optimal operation. Furthermore, the model's efficacy largely hinges on the quality of training data, necessitating meticulous curation and a timely examination of the influence of various datasets on language modeling.

4. Tools and Frameworks for Language Modeling

There are plenty of frameworks available for developing language models. Listed below are commonly used libraries and tools that are user-friendly and open-source.

NLTK is an open-source library that offers a plethora of tools for natural language processing. It provides tools for text processing, information extraction, and language model training (https://www.nltk.org/ (accessed on 19 July 2023)).

NLTK provides a user-friendly interface for various tasks, including tokenization, stemming, lemmatization, syntactic analysis, and sentiment analysis. Researchers, developers, and data processing specialists worldwide widely utilize NLTK to develop applications in natural language processing and text data analysis. One of the key advantages of NLTK is its extensive collection of corpora, incorporating textual data from diverse sources such as books, news articles, and social media platforms. These corpora serve as a rich source of data for training and testing models in the field of natural language processing [79].

Despite the wide benefits of the framework, there are some limitations.

- It may not scale efficiently for large-scale production systems;
- Some components may run slower than optimized platforms.

TensorFlow is an open-source deep learning framework that is widely used to implement language models. It offers many features that simplify the creation and training of language models, including the following:

– Libraries for handling textual data;
– Algorithms for training language models;
– Tools for evaluating language models (https://www.tensorflow.org/ (accessed on 19 July 2023)).

TensorFlow stands out as a widely adopted deep learning framework supported by a vibrant and robust community. It offers specialized libraries tailored for tasks such as processing textual data, training language models, and evaluating model performance. The ecosystem of TensorFlow is resilient, providing robust support for deploying models across diverse environments. However, TensorFlow falls short on both usability and speed, and debugging deep learning models within the TensorFlow framework can pose challenges [80].

PyTorch is an another popular open-source deep learning framework used for language model implementations (https://pytorch.org/ (accessed on 19 July 2023)). It has functionalities similar to TensorFlow but is often lauded for its more user-friendly syntax.

PyTorch provides many capabilities and is particularly influential in deep learning areas such as natural language processing (NLP) and computer vision.

Some of the outstanding characteristics of PyTorch include its high execution speed, which is achievable even when processing complex graphs. This flexible library can run efficiently on simple processors or in conjunction with graphics processors. PyTorch has powerful APIs that make the library extensible and also provide a set of tools for natural language processing.

Strengths:

- A dynamic computation graph allows for easier debugging and model modification;
- It is a popular choice for researchers due to its flexibility.

Weaknesses:

- A typical machine learning toolkit;
- In-depth knowledge of fundamental NLP algorithms is necessary [81].

Hugging Face is an open-source library specializing in transformer-based language models. It encompasses pre-trained models as well as tools for fine-tuning and evaluating them (https://huggingface.co/ (accessed on 19 July 2023)). The library stands out for its extensive collection of pre-trained transformer models, offering a repository that includes models for tasks such as text classification, named-entity recognition, summarization, and translation.

Key information about Hugging Face in language modeling includes the following:

- It is known for its focus on transformer architectures, making it particularly suitable for tasks that benefit from the attention mechanism and contextual embeddings;
- The library provides a rich selection of pre-trained transformer models, enabling users to leverage state-of-the-art language representations for their specific NLP applications;

- It offers tools for fine-tuning pre-trained models on domain-specific data and evaluating their performance on targeted tasks. This allows users to adapt models to their specific needs;
- It is designed to be user-friendly, providing straightforward interfaces for loading pre-trained models, conducting inference, and fine-tuning for custom tasks. This accessibility contributes to its popularity among both beginners and experienced practitioners.

Despite these strengths, it is essential to note that Hugging Face's specialization in transformer architectures may limit its suitability for tasks that require different model structures. Additionally, customization options might be more constrained compared to lower-level frameworks for users with specific requirements beyond transformers.

Gensim is also an open-source library primarily known for topic modeling and similar tasks. While it provides tools based on Hidden Markov Models (HMM), it is better known for its word-embedding functionalities, like Word2Vec, rather than HMM-based language models. It offers tools for training, evaluating, and visualizing models.

Gensim excels in handling input data that exceed the capacity of RAM, employing algorithms for efficient processing. With user-friendly interfaces, Gensim offers a multi-core implementation of algorithms like Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA). Common applications of the library involve text similarity searches and the transformation of words and documents into vectors.

Some of the key advantages of Gensim are as follows:

- A user-friendly interface;
- Scalability;
- Effective implementation of popular algorithms like LSA and LDA.

Despite the wide benefits of the framework, there are some limitations:

- It was primarily designed for unsupervised text modeling;
- It often requires integration with other libraries such as NLTK [81].

Evaluation Criteria in Language Modeling

There are various evaluation criteria that can be utilized to assess language models. Some of the most common criteria include the following:

Perplexity—a measure of how well a language model predicts the next word in a sequence [82–88]. A lower perplexity value indicates that the language model is better at predicting the subsequent word (9).

$$Perplexity = \prod_{i=1}^{n} p(w_i|w_1, ..., w_{i-1}) \quad (9)$$

where n is the total number of words in the corpus, w_i represents the i-th word in the corpus, and, $p(w_i|w_1, ..., w_{i-1})$ is the probability of the i-th word given the preceding words as predicted by the model.

Bits-per-character (BPC)—a metric that measures the average number of bits required to encode a character in a language model [89–91]. A lower BPC indicates that the language model is more efficient at encoding characters (10).

$$Bits - per - character = -\frac{1}{L} \cdot \sum_i log_2(P(x_i)) \quad (10)$$

where L is the length of the text, x_i represents the i-th character in the text, and $P(x_i)$ is the probability of the i-th character appearing according to the language model.

Cross-entropy—another measure of how well a language model predicts the next word in a sequence [92,93]. It is similar to Perplexity but is calculated differently. Cross Entropy compares two probability distributions, $P(x)$ and $Q(x)$. In the context of language

models, we are comparing the predicted word probability distribution against the actual word probability distribution (11).

$$Cross - entropy = -\sum_i P(x_i) log(Q(x_i)) \quad (11)$$

Here, $P(x_i)$ is the probability of the i-th word appearing according to the language model, while $Q(x_t)$ is the probability of the t-th word in the actual sequence.

There is also human evaluation which involves having people rate the quality of a text produced by a language model. While this approach may be more subjective than alternative tests, it helps in assessing the overall superiority of the language model's outputs. The best evaluation metrics for a specific language model depend on the task it is being used for. For instance, if the language model is being used for machine translation, Perplexity or cross-entropy might be more relevant. If it is used for text generation, human evaluations might be more pertinent.

In addition to the aforementioned metrics, there are several other factors to consider when evaluating language models. These include the following:

- The size of the language model's training dataset;
- The architecture of the language model;
- The computational resources used for training the model;
- The specific task for which the language model is used.

Language model evaluation remains an active area of research. As language models become more potent, new evaluation metrics are being developed to better measure their performance.

5. Key Challenges of Language Modeling for Languages with Morphological Structure

Many NLP research studies focus on popular languages with simple morphological structures, such as English and Chinese. Based on these studies, it is challenging to assess the effectiveness of various methods and their enhancements for other languages, especially for agglutinative languages which have complex morphological structures [94]. This question remains unresolved since the efficiency of these methods for languages of different typologies has not been fully determined.

Researchers Gerz et al. [95] implemented n-gram and neural language models for 50 languages. They found that a model's effectiveness heavily depends on the language's morphological structure. Their results suggest that the complexity of modeling varies for each language, which stems from the diverse morphological makeup of these languages. These languages fall into different linguistic groups, such as agglutinative (mostly Turkic languages), isolating, fusional, and introflexive.

From this, one can infer that a language's morphological structure influences the complexity of language modeling. There are several aspects related to language modeling for languages with rich morphology:

- Languages with rich morphological diversity have a multitude of grammatical forms and rules, leading to an increase in word form variations, making modeling and understanding them more challenging. Complex morphology can result in ambiguity in which the same word form can have different grammatical meanings in various contexts, complicating precise interpretation and text generation.
- Different language types (agglutinative, isolating, fusional, etc.) have their unique morphological characteristics. This demands the development of specialized modeling methods that effectively account for these differences. Many languages with rich morphological structures, like several agglutinative languages, are often under-researched and have limited data for model training, impacting the quality of produced language models.
- In some languages, words can be lengthy due to the presence of various morphemes. This poses challenges when processing them with neural models, especially if the model has a limited context length.

Considering these complexities, research in language modeling should account for the morphological intricacies of various languages and refine methods specifically tailored to their unique requirements.

6. Conclusions

In this article, we navigated through the complex and evolving landscape of language modeling, highlighting the breadth and depth of methodologies and technologies that have been developed and refined to advance the field of NLP. From foundational statistical approaches like n-grams and Hidden Markov Models to the groundbreaking advancements brought forth by neural network models, the journey of language modeling reflects a continuous quest for more sophisticated, accurate, and efficient ways to process and understand human language.

The shift from traditional models like n-grams and Hidden Markov Models to neural-network-based approaches marks a pivotal change, with the latter offering nuanced means of capturing semantic relationships and contextual subtleties through models like Word2Vec, GloVe, and FastText. The emergence of pretrained language models such as BERT, GPT, LLAMA, StableLM, and Bard, primarily based on the transformer architecture, represents a significant leap forward, significantly enhancing performance across a spectrum of NLP tasks.

This article also highlights the exploration of language models for agglutinative languages, emphasizing the need for versatile models that accommodate the diverse linguistic structures worldwide. This endeavor is crucial for achieving inclusivity in language technology.

While the analysis of various tools and frameworks—NLTK, TensorFlow, PyTorch, and Gensim—demonstrate a rich landscape of options for language modeling, each with its unique strengths and challenges, future work must focus on enhancing these tools for greater accessibility and efficiency.

Future work in language modeling can be directed toward solving some key challenges, such as the following:

- Managing linguistic diversity: There is an increasing demand for the development of models that can effectively handle a wide range of languages, especially those with limited resources and morphologically complex languages. Future research should be focused on creating more inclusive models that are capable of learning from limited data and can be easily adapted to different linguistic contexts.
- Energy efficiency and scalability: Current models demand significant computational resources. Future studies should investigate ways to make these models more energy-efficient and scalable, which would enable their deployment in more varied environments with limited resources.
- Explainability and transparency: As models grow more complex, ensuring their explainability and transparency becomes essential. Research aimed at making these models more understandable to users and developers will be critical for their responsible and effective usage.

In conclusion, the field of language modeling is poised for continued innovation and growth. The developments thus far have significantly enhanced our ability to process and generate human language, bridging the gap between humans and machines. The future of language modeling, while challenging, holds immense potential for further breakthroughs that could redefine our interaction with technology and deepen our understanding of language and communication.

Author Contributions: Conceptualization, D.O. and O.M.; methodology, D.K.; formal analysis, K.M.; investigation, D.O.; resources, M.O.; data curation, D.O. and D.K.; writing—original draft preparation, D.O.; writing—review and editing, K.M.; visualization, D.K.; supervision, D.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No. AP19174298).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wei, C.; Wang, Y.; Wang, B.; Kuo, C.J. An Overview on Language Models: Recent Developments and Outlook. *arXiv* **2023**, arXiv:2303.05759.
2. Kempe, V.; Brooks, P.J. Modern Theories of Language. In *Encyclopedia of Evolutionary Psychological Science*; Shackelford, T.K., Weekes-Shackelford, V.A., Eds.; Springer: Cham, Germany, 2021. [CrossRef]
3. Li, H. Language models: Past, present, and future. *Commun. ACM* **2022**, *65*, 56–63. [CrossRef]
4. Hombaiah, S.A.; Chen, T.; Zhang, M.; Bendersky, M.; Najork, M. Dynamic Language Models for Continuously Evolving Content. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Virtual Event, Singapore, 14–18 August 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 2514–2524. [CrossRef]
5. Jurafsky, D.; Martin, J.H. Speech and Language Processing—2023—Draft of January 7, 2023. Available online: https://web.stanford.edu/~jurafsky/slp3/ (accessed on 7 July 2023).
6. Qian, P.; Naseem, T.; Levy, R.P.; Astudillo, R.F. Structural Guidance for Transformer Language Models. Annual Meeting of the Association for Computational Linguistics. 2021. Structural Guidance for Transformer Language Models. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Virtual, 1–6 August 2021; Volume 1, pp. 3735–3745. [CrossRef]
7. Kobayashi, G.; Kuribayashi, T.; Yokoi, S.; Inui, K. Transformer Language Models Handle Word Frequency in Prediction Head. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2023, Toronto, ON, Canada, 9–14 July 2023; pp. 4523–4535. [CrossRef]
8. Kaji, N.; Kobayashi, H. Incremental Skip-gram Model with Negative Sampling. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 7–11 September 2017; pp. 363–371, Association for Computational Linguistics. [CrossRef]
9. Li, Z.; Huang, W.; Xiong, Y.; Ren, S.; Zhu, T. Incremental learning imbalanced data streams with concept drift: The dynamic updated ensemble algorithm. *Knowl. Based Syst.* **2020**, *195*, 105694. [CrossRef]
10. Chelba, C.; Norouzi, M.; Bengio, S. N-gram Language Modeling using Recurrent Neural Network Estimation. *arXiv* **2017**, arXiv:1703.10724.
11. Aouragh, S.; Yousfi, A.; Laaroussi, S. A new estimate of the n-gram language model. *Procedia Comput. Sci.* **2021**, *189*, 211–215. [CrossRef]
12. Diao, S.; Xu, R.; Su, H.; Jiang, Y.; Song, Y.; Zhang, T. Taming Pre-trained Language Models with N-gram Representations for Low-Resource Domain Adaptation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Online, 1–6 August 2021; Volume 1: Long Papers, pp. 3336–3349.
13. Li, H.; Cai, D.; Xu, J.; Watanabe, T. Residual Learning of Neural Text Generation with n-gram Language Model. In Proceedings of the Findings of the Association for Computational Linguistics: Conference on Empirical Methods in Natural Language Processing, Abu Dhabi, United Arab Emirates, 7–11 December 2022; pp. 1523–1533.
14. Rabiner, L.R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. IEEE* **1989**, *77*, 257–286. [CrossRef]
15. Neustein, A. *Advances in Speech Recognition: Mobile Environments, Call Centers and Clinics*; Springer: Berlin/Heidelberg, Germany, 2010. [CrossRef]
16. Anandika, A.; Mishra, S.P.; Das, M. Review on Usage of Hidden Markov Model in Natural Language Processing. In *Intelligent and Cloud Computing. Smart Innovation, Systems and Technologies*; Mishra, D., Buyya, R., Mohapatra, P., Patnaik, S., Eds.; Springer: Singapore, 2021; Volume 194. [CrossRef]
17. Gao, X.; Zhu, N. Hidden Markov Model and its Application in Natural Language Processing. *Inf. Technol. J.* **2013**, *12*, 4256–4261. [CrossRef]
18. Mikolov, T.; Corrado, D. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781. Available online: https://arxiv.org/pdf/1301.3781.pdf (accessed on 25 July 2023).
19. Mikolov, T.; Yih, W.; Zweig, G. Linguistic Regularities in Continuous Space Word Representations. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Atlanta, Georgia, 10–12 June 2013; pp. 746–751.

20. Ling, W.; Dyer, C.; Black, A.W.; Trancoso, I. Two/Too Simple Adaptations of Word2Vec for Syntax Problems. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, CO, USA, 31 May–5 June 2015; pp. 1299–1304. [CrossRef]
21. Landthaler, J.; Waltl, B.; Huth, D.; Braun, D.; Matthes, F.; Stocker, C.; Geiger, T. Extending Thesauri Using Word Embeddings and the Intersection Method. In Proceedings of the Second Workshop on Automated Semantic Analysis of Information in Legal Texts Co-Located with the 16th International Conference on Artificial Intelligence and Law, London, UK, 16 June 2017.
22. Pennington, J.; Socher, R.; Manning, C. Glove: Global Vectors for Word Representation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; Volume 14, pp. 1532–1543. [CrossRef]
23. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching Word Vectors with Subword Information. *Trans. Assoc. Comput. Linguist.* **2017**, *5*, 135–146. [CrossRef]
24. Mojumder, P.; Hasan, M.; Hossain, F.; Hasan, K. A Study of fastText Word Embedding Effects in Document Classification in Bangla Language. In Proceedings of the Cyber Security and Computer Science: Second EAI International Conference, ICONCS 2020, Dhaka, Bangladesh, 15–16 February 2020. [CrossRef]
25. NLP Series: Distributional Semantics | Co-Occurrence Matrix. Available online: https://medium.com/@imamitsehgal/nlp-series-distributional-semantics-occurrence-matrix-401fafa28776 (accessed on 25 July 2023).
26. Zhang, X.; LeCun, Y. Which Encoding is the Best for Text Classification in Chinese, English, Japanese and Korean? *arXiv* **2017**, arXiv:1708.02657.
27. Park, S.; Byun, J.; Baek, S.; Cho, Y.; Oh, A.H. Subword-level Word Vector Representations for Korean. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, South Wharf, Australia, 15–20 July 2018; Volume 1, pp. 2429–2438. [CrossRef]
28. Şahİn, G. Turkish document classification based on Word2Vec and SVM classifier. In Proceedings of the 2017 25th Signal Processing and Communications Applications Conference, Antalya, Turkey, 15–18 May 2017; pp. 1–4. [CrossRef]
29. Pervan, N.; Keles, H.Y. Sentiment analysis using a random forest classifier on turkish web comments. *Commun. Fac. Sci. Univ. Ankara. Ser. A2-A3 Phys. Eng. Phys. Electron. Eng. Astron.* **2017**, *59*, 69–79. [CrossRef]
30. Savaş, Y.; Tugba, Y. Learning Turkish Hypernymy UsingWord Embeddings. *Int. J. Comput. Intell. Syst.* **2018**, *11*, 371. [CrossRef]
31. Bilgin, M.; Şentürk, İ.F. Sentiment analysis on Twitter data with semi-supervised Doc2Vec. In Proceedings of the 2017 International Conference on Computer Science and Engineering (UBMK), Antalya, Turkey, 5–8 October 2017; pp. 661–666. [CrossRef]
32. Shapiro, P.; Duh, K. Morphological Word Embeddings for Arabic Neural Machine Translation in Low-Resource Settings. In Proceedings of the Second Workshop on Subword/Character LEvel Models, New Orleans, LA, USA, 5–7 June 2018; pp. 1–11. [CrossRef]
33. Abdulateef, S.; Khan, N.A.; Che, B.; Shan, X. Multidocument Arabic Text Summarization Based on Clustering and Word2Vec to Reduce Redundancy. *Information* **2020**, *11*, 59. [CrossRef]
34. Al-Hajj, M.; Jarrar, M. LU-BZU at SemEval-2021 Task 2: Word2Vec and Lemma2Vec performance in Arabic Word-in-Context disambiguation. *arXiv* **2021**, arXiv:2104.08110. [CrossRef]
35. Khusainova, A.; Khan, A.; Ramírez Rivera, A. SART—Similarity, Analogies, and Relatedness for Tatar Language: New Benchmark Datasets for Word Embeddings Evaluation. *arXiv* **2019**, arXiv:1904.00365.
36. Huseynov, K.; Suleymanov, U.; Rustamov, S.; Huseynov, J. Training and Evaluation of Word Embedding Models for Azerbaijani Language. In *Digital Interaction and Machine Intelligence. MIDI 2020. Advances in Intelligent Systems and Computing*; Biele, C., Kacprzyk, J., Owsiński, J.W., Romanowski, A., Sikorski, M., Eds.; Springer: Cham, Germany, 2021; Volume 1376. [CrossRef]
37. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
38. Chi, Z.; Huang, S.; Dong, L.; Ma, S.; Zheng, B.; Singhal, S.; Bajaj, P.; Song, X.; Mao, X.; Huang, H. XLM-E: Cross-lingual Language Model Pre-training via ELECTRA. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, Dublin, Ireland, 22–27 May 2022; Volume 1, pp. 6170–6182. [CrossRef]
39. Araci, D. FinBERT: Financial Sentiment Analysis with Pre-trained Language Models. *arXiv* **2019**, arXiv:1908.10063.
40. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692.
41. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv* **2019**, arXiv:1909.11942. [CrossRef]
42. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.; Le, Q. XLNet: Generalized autoregressive pretraining for language understanding. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 8–14 December 2019; Article 517, pp. 5753–5763. [CrossRef]
43. He, P.; Liu, X.; Gao, J.; Chen, W. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. *arXiv* **2020**, arXiv:2006.03654.
44. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv* **2019**, arXiv:1910.01108.

45. Clark, K.; Luong, M.; Le, Q.; Manning, C. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In Proceedings of the International Conference on Learning Representations, 2020, Addis Ababa, Ethiopia, 26–30 April 2020. Available online: https://openreview.net/forum?id=r1xMH1BtvB (accessed on 13 August 2023).
46. Abdul-Mageed, M.; Elmadany, A.; Nagoudi, E.M. ARBERT & MARBERT: Deep Bidirectional Transformers for Arabic. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Virtual, 1–6 August 2021; Volume 1: Long Papers, pp. 7088–7105. [CrossRef]
47. Harrag, F.; Dabbah, M.; Darwish, K.; Abdelali, A. Bert Transformer model for Detecting Arabic GPT2 Auto-Generated Tweets. In Proceedings of the Fifth Arabic Natural Language Processing Workshop, Barcelona, Spain, 12 December 2020; pp. 207–214. Available online: https://aclanthology.org/2020.wanlp-1.19 (accessed on 13 August 2023).
48. Bozuyla, M.; Özçift, A. Developing a fake news identification model with advanced deep language transformers for Turkish COVID-19 misinformation data. *Turk. J. Electr. Eng. Comput. Sci.* **2022**, *30*, 27. [CrossRef]
49. Al-qurish, M.; Alqaseemi, S.; Souissi, R. AraLegal-BERT: A pretrained language model for Arabic Legal text. In Proceedings of the Natural Legal Language Processing Workshop 2022, Abu Dhabi, United Arab Emirates, 8 December 2022; pp. 338–344. [CrossRef]
50. Antoun, W.; Baly, F.; Hajj, H. AraBERT: Transformer-based Model for Arabic Language Understanding. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection, Marseille, France, 12 May 2020; pp. 9–15. Available online: https://aclanthology.org/2020.osact-1.2 (accessed on 21 August 2023).
51. Abdelali, A.; Darwish, K.; Durrani, N.; Mubarak, H. Farasa: A fast and furious segmenter for arabic. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, San Diego, CA, USA, 12–17 June 2016; pp. 11–16. [CrossRef]
52. Abdelgwad, M.M.; Soliman, T.H.A.; Taloba, A.I. Arabic aspect sentiment polarity classification using BERT. *J. Big Data* **2022**, *9*, 115. [CrossRef]
53. Alammary, A.S. BERT Models for Arabic Text Classification: A Systematic Review. *Appl. Sci.* **2022**, *12*, 5720. [CrossRef]
54. Acikalin, U.U.; Bardak, B.; Kutlu, M. Turkish Sentiment Analysis Using BERT. In Proceedings of the 2020 28th Signal Processing and Communications Applications Conference, Gaziantep, Turkey, 5–7 October 2020; pp. 1–4. [CrossRef]
55. Mutlu, M.M.; Özgür, A. A Dataset and BERT-based Models for Targeted Sentiment Analysis on Turkish Texts. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, Dublin, Ireland, 22–27 May 2022; pp. 467–472. [CrossRef]
56. Kim, H.; Kim, S.; Kang, I.; Kwak, N.; Fung, P. Korean Language Modeling via Syntactic Guide. In Proceedings of the Thirteenth Language Resources and Evaluation Conference, Marseille, France, 20–25 June 2022; pp. 2841–2849. Available online: https://aclanthology.org/2022.lrec-1.304 (accessed on 21 August 2023).
57. Kawazoe, Y.; Shibata, D.; Shinohara, E.; Aramaki, E.; Ohe, K. A clinical specific BERT developed with huge size of Japanese clinical narrative. *medRxiv* **2020**. [CrossRef]
58. Wang, Z.; Karthikeyan, K.; Mayhew, S.; Roth, D. Extending Multilingual BERT to Low-Resource Languages. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020, Online, 16–20 December 2020; pp. 2649–2656.
59. Feng, F.; Yang, Y.; Cer, D.; Arivazhagan, N.; Wang, W. Language-agnostic BERT Sentence Embedding. *arXiv* **2022**, arXiv:2007.01852. [CrossRef]
60. Mansurov, B.; Mansurov, A. UzBERT: Pretraining a BERT model for Uzbek. *arXiv* **2021**, arXiv:2108.09814.
61. Radford, A.; Wu, J.; Rewon, C.; Luan, D.; Dario, A.; Sutskever, I. Language Models are Unsupervised Multitask Learners. *OpenAI Blog* **2018**, *1*, 9. Available online: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf (accessed on 21 August 2023).
62. Ray, P.P. ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet Things Cyber Phys. Syst.* **2023**, *3*, 121–154. [CrossRef]
63. Number of ChatGPT Users and Key Stats. 2023. Available online: https://www.namepepper.com/chatgpt-users (accessed on 21 August 2023).
64. Antoun, W.; Baly, F.; Hajj, H. AraGPT2: Pre-Trained Transformer for Arabic Language Generation. In Proceedings of the Sixth Arabic Natural Language Processing Workshop, Kiev, Ukraine, 19 April 2021; pp. 196–207. Available online: https://aclanthology.org/2021.wanlp-1.21 (accessed on 10 September 2023).
65. Farha, I.; Magdy, W. Benchmarking Transformer-based Language Models for Arabic Sentiment and Sarcasm Detection. In Proceedings of the Sixth Arabic Natural Language Processing Workshop, Kyiv, Ukraine, 19 April 2021; pp. 21–31.
66. Nagoudi, E.M.; Abdul-Mageed, M.; Elmadany, A.; Inciarte, A.A.; Khondaker, M.T. JASMINE: Arabic GPT Models for Few-Shot Learning. *arXiv* **2022**, arXiv:2212.10755.
67. Karfi, I.E.; Fkihi, S.E. A combined Bi-LSTM-GPT Model for Arabic Sentiment Analysis. *Int. J. Intell. Syst. Appl. Eng.* **2023**, *11*, 77–84. Available online: https://ijisae.org/index.php/IJISAE/article/view/3144 (accessed on 21 September 2023).
68. Kim, B.; Kim, H.; Lee, S.; Lee, G.; Kwak, D.; Hyeon, J.; Park, S.; Kim, S.; Kim, S.; Seo, D.; et al. What Changes Can Large-scale Language Models Bring? Intensive Study on HyperCLOVA: Billions-scale Korean Generative Pretrained Transformers. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Virtual, 7–11 November 2021; pp. 3405–3424. [CrossRef]

69. Lee, H.; Hong, S.; Park, J.; Kim, T.; Kim, G.; Ha, J. KoSBI: A Dataset for Mitigating Social Bias Risks towards Safer Large Language Model Applications. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, Toronto, ON, Canada, 9–14 July 2023; Volume 5: Industry Track, pp. 208–224. [CrossRef]
70. Lee, H.; Hong, S.; Park, J.; Kim, T.; Cha, M.; Choi, Y.; Kim, B.; Kim, G.; Lee, E.; Lim, Y.; et al. SQuARe: A Large-Scale Dataset of Sensitive Questions and Acceptable Responses Created through Human-Machine Collaboration. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, Toronto, ON, Canada, 9–14 July 2023; Volume 1: Long Papers, pp. 6692–6712.
71. Kasai, J.; Kasai, Y.; Sakaguchi, K.; Yamada, Y.; Radev, D.R. Evaluating GPT-4 and ChatGPT on Japanese Medical Licensing Examinations. *arXiv* **2023**, arXiv:2303.18027.
72. Lai, V.D.; Ngo, N.T.; Veyseh, A.P.; Man, H.; Dernoncourt, F.; Bui, T.; Nguyen, T.H. ChatGPT Beyond English: Towards a Comprehensive Evaluation of Large Language Models in Multilingual Learning. *arXiv* **2023**, arXiv:2304.05613.
73. Hoffmann, J.; Borgeaud, S.; Mensch, A.; Buchatskaya, E.; Cai, T.; Rutherford, E.; Casas, D.D.; Hendricks, L.A.; Welbl, J.; Clark, A.; et al. Training Compute-Optimal Large Language Models. *arXiv* **2022**, arXiv:2203.15556.
74. Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. LLaMA: Open and Efficient Foundation Language Models. *arXiv* **2023**, arXiv:2302.13971.
75. Li, Y.; Li, Z.; Zhang, K.; Dan, R.; Jiang, S.; Zhang, Y. ChatDoctor: A Medical Chat Model Fine-Tuned on a Large Language Model Meta-AI (LLaMA) Using Medical Domain Knowledge. *Cureus* **2023**, *15*, e40895. [CrossRef]
76. StableLM: Stability AI Language Models. Available online: https://github.com/Stability-AI/StableLM (accessed on 3 October 2023).
77. Siad, S.M. *The Promise and Perils of Google's Bard for Scientific Research*; International Centre for Advanced Mediterranean Agronomic Studies: Bari, Italy, 2023. [CrossRef]
78. Qin, H.; Ji, G.P.; Khan, S.; Fan, D.; Khan, F.; Gool, L. How Good is Google Bard's Visual Understanding? An Empirical Study on Open Challenges. *Mach. Intell. Res.* **2023**, *20*, 605–613. [CrossRef]
79. Yao, J. Automated Sentiment Analysis of Text Data with NLTK. *J. Phys. Conf. Ser.* **2019**, *1187*, 052050. [CrossRef]
80. Scikit-Learn vs TensorFlow: A Detailed Comparison. Available online: https://www.simplilearn.com/scikit-learn-vs-tensorflow-article#what_is_tensorflow (accessed on 3 October 2023).
81. 10 Best Python Libraries for Natural Language Processing. Available online: https://www.unite.ai/10-best-python-libraries-for-natural-language-processing/ (accessed on 3 October 2023).
82. Liu, H.; Zhang, C. Reinforcement Learning based Neural Architecture Search for Audio Tagging. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8. [CrossRef]
83. Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q.; Salakhutdinov, R. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. *arXiv* **2019**, arXiv:1901.02860.
84. Antonello, R.; Turek, J.; Huth, A.G. Selecting Informative Contexts Improves Language Model Fine-tuning. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, Online, 6–8 July 2020; arXiv:2005.00175.
85. Shleifer, S.; Weston, J.; Ott, M. NormFormer: Improved Transformer Pretraining with Extra Normalization. *arXiv* **2021**, arXiv:2110.09456.
86. Baevski, A.; Auli, M.L. Adaptive Input Representations for Neural Language Modeling. *arXiv* **2018**, arXiv:1809.10853.
87. Arora, K.; Shuster, K.; Sukhbaatar, S.; Weston, J. Director: Generator-classifiers for supervised language modeling. In Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing, Online, 20–23 November 2022; Volume 1, pp. 512–526.
88. Wang, B.; Ping, W.; Xiao, C.; Xu, P.; Patwary, M.; Shoeyb, M.; Li, B.; Anandkumar, A.; Catanzaro, B. Exploring the Limits of Domain-Adaptive Training for Detoxifying Large-Scale Language Models. *arXiv* **2022**, arXiv:2202.04173.
89. Blevins, T.; Zettlemoyer, L. Better Character Language Modeling through Morphology. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 1606–1613. [CrossRef]
90. Al-Rfou, R.; Choe, D.; Constant, N.; Guo, M.; Jones, L. Character-Level Language Modeling with Deeper Self-Attention. *arXiv* **2018**, arXiv:1808.04444. [CrossRef]
91. Lei, T. When Attention Meets Fast Recurrence: Training Language Models with Reduced Compute. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (2021), Virtual, 7–11 November 2021. [CrossRef]
92. Zhang, S.; Wu, S.; Irsoy, O.; Lu, S.; Bansal, M.; Dredze, M.; Rosenberg, D. MixCE: Training Autoregressive Language Models by Mixing Forward and Reverse Cross-Entropies. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, Toronto, ON, Canada, 9–14 July 2023; Volume 1, pp. 9027–9050. [CrossRef]
93. Kim, G.; Hong, T.; Yim, M.; Nam, J.; Park, J.; Yim, J.; Hwang, W.; Yun, S.; Han, D.; Park, S. OCR-Free Document Understanding Transformer. In *Computer Vision—ECCV 2022. ECCV 2022*; Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T., Eds.; Lecture Notes in Computer Science; Springer: Cham, Germany, 2022; Volume 13688. [CrossRef]

94. Khan, J.A.; Liu, L.; Jia, Y.; Wen, L. Linguistic analysis of crowd requirements: An experimental study. In Proceeding of the 2018 IEEE 7th International Workshop on Empirical Requirements Engineering (EmpiRE), Banff, AB, Canada, 21 August 2018; pp. 24–31.
95. Shareghi, E.; Gerz, D.; Vulić, I.; Korhonen, A. Show Some Love to Your n-grams: A Bit of Progress and Stronger n-gram Language Modeling Baselines. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019; Volume 1 (Long and Short Papers), pp. 4113–4118.