ISSN: 2710-0588

The Development of NLP Models

Keming Zhang

University Of Rochester, Rochester, NY, USA

Abstract

This article reviews the development of Natural Language Processing (NLP) models from early statistical approaches to modern large language models (LLMs). Beginning with probability-based n-gram models, it outlines their limitations in data sparsity and long-term dependency. It then introduces neural network-based models, including word embeddings, logistic regression, multi-layer perceptrons, and Word2Vec, followed by sequential models such as RNNs and LSTMs that capture temporal dependencies. The shift to pre-trained models, marked by Word2Vec and the Transformer architecture, enabled scalable transfer learning and laid the foundation for state-of-the-art models like BERT, GPT, and their derivatives. Applications in text classification are illustrated through experiments with RoBERTa, hybrid BERT-LightGBM models, and fine-tuning techniques such as LoRA. Finally, the article discusses the broader ecosystem of large models, including chat models, multimodal models, and agent frameworks that integrate planning, memory, and tool use. The review emphasizes both theoretical principles and practical workflows, highlighting the necessity of iterative learning, coding practice, and ecosystem familiarity for effectively leveraging NLP technologies.

Keywords

Natural Language Processing (NLP); Word Embeddings; Recurrent Neural Networks (RNN); Long Short-Term Memory (LSTM); Transformer; BERT; Large Language Models (LLMs); Fine-Tuning.

1. Foundamental NLP Models and Related Techniques

1.1. Introduction

NLP is a technology that enables machines to process human language, relying heavily on language models. Language models have evolved rapidly, from statistical models to neural network models, then to pre-trained models, and finally to large language models (LLMs) with hundreds of millions or even billions of parameters. Since NLP is a subset of machine learning and deep learning, the earlier statistical language models are strongly tied to probability theory, whereas modern models rely more on machine learning and deep learning concepts.

1.2. Statistical Probability-Based Language Models: n-gram

The main purpose of a language model can be understood as predicting the probability distribution of the next vocabulary word given the preceding text. This naturally suggests using conditional probability.

$$\text{Chain Rule: } P(w_1,w_2,\ldots w_n)=\prod_{i=1}^{n} P(w_i|w_1,w_2,\ldots w_{i-1}) \quad (1)$$

To reduce computational complexity, the Markov assumption is applied—where the probability of a word depends only on a limited number (k) of preceding words.

$$\text{Unigram: } k=0,\ P(w_i|w_1,w_2,\ldots w_{i-1})\approx P(w_i) \quad (2)$$

ISSN: 2710-0588

$$\text{Bigram: } k=1,\ P(w_i|w_1,w_2,\ldots w_{i-1})\approx P(w_i|w_{i-1}) \tag{3}$$

$$\text{Trigram: } k=2,\ P(w_i|w_1,w_2,\ldots w_{i-1})\approx P(w_i|w_{i-1},w_{i-2}) \tag{4}$$ [1]

This approach allows the prediction of the most likely next vocabulary token, but it suffers from data sparsity, lack of long-term dependency, and poor generalization ability.

1.3. Neural Network-Based Language Models

As time progressed, language models entered the neural network era. It is essential to first understand word embeddings, a technique used in these models, and the principle of model learning procedure through gradient descent (GD), which underpins optimization in deep learning models.

1.3.1. Word Embedding

Word embeddings map every unique vocabulary in a text to high-dimensional numerical vectors. Each vector dimension represents different semantic topics, capturing the meaning of word. Using a high-quality corpus can produce better embeddings.

Here is an example of reducing word vectors to 2D points and drawing them on the coordinate plane using techiniques like PCA:

king + man – woman = queen.

Figure 1. 2D Word Embedding

1.3.2. Introduction to Gradient Descent through Logistic Regression

A logistic regression model consists of a linear regression function combined with a sigmoid function for binary classification, with outputs between 0 and 1 representing probabilities.

When z_i is large and positive, $y_i=1$ is likely; When z_i is large and negative, $y_i=0$ is likely.

$$\text{Linear Regression: } z_i=(w_1*x_{i1})+(w_2*x_{i2})+\ldots+(w_m*x_{im})+b_0 \tag{5}$$

$$\text{Sigmoid Function: } P(y_i=1|x_i)=\sigma(z_i)=\frac{1}{1+e^{-z_i}} \tag{6}$$

ISSN: 2710-0588

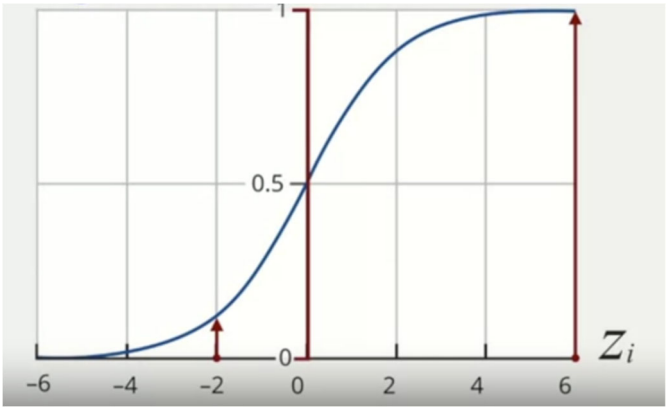
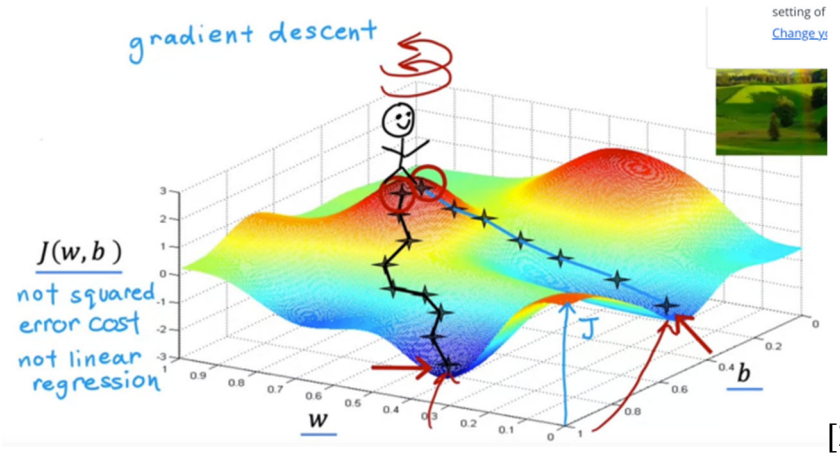


Figure 2. Sigmoid Curve

Gradient descent can be visualized as a person descending a mountain, randomly selecting the way down the hill, iteratively optimizing the weights (w) and biases (b) with a suitable learning rate (α) to minimize loss (cost) until reaching the valley. Early stopping is necessary when the validation loss stops decreasing, rather than training until the training loss is minimized.



[2]

Figure 3. Gradient Descent Procedure

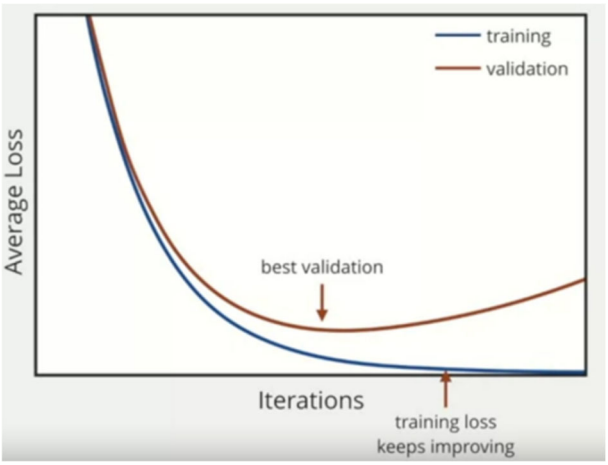


Figure 4. Illustration of Early Stopping

$$f_{\vec{w},b}(\vec{x})=g(z) \tag{7}$$

ISSN: 2710-0588

$$\text{Logistic Loss Function: } L(f_{\vec{w},b}(\vec{x}_i),y_i)=\begin{cases}-\log(f_{\vec{w},b}(\vec{x}_i)) & \text{if } y_i=1\\ -\log(1-f_{\vec{w},b}(\vec{x}_i)) & \text{if } y_i=0\end{cases} \tag{8}$$

$$\text{Cost Function: } J(\vec{w},b)=-\frac{1}{m}\sum_{i=1}^{m}\left[y_i\log(f_{\vec{w},b}(\vec{x}_i))+(1-y_i)\log(1-f_{\vec{w},b}(\vec{x}_i))\right] \tag{9}$$

Gradient Descent:
repeat[

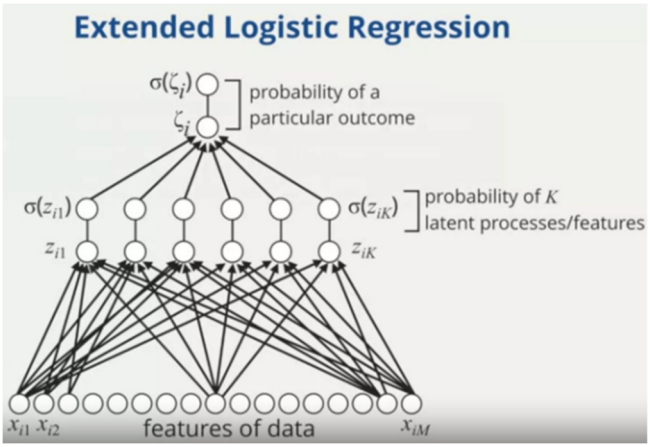
$w_j=w_j-\alpha\frac{\partial}{\partial w_j}J(\vec{w},b)$ $\quad\frac{\partial}{\partial w_j}J(\vec{w},b)=\frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}_i)-y_i)x_{ji}$

$b=b-\alpha\frac{\partial}{\partial b}J(\vec{w},b)$ $\quad\frac{\partial}{\partial b}J(\vec{w},b)=\frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}_i)-y_i)$

] simultaneous updates

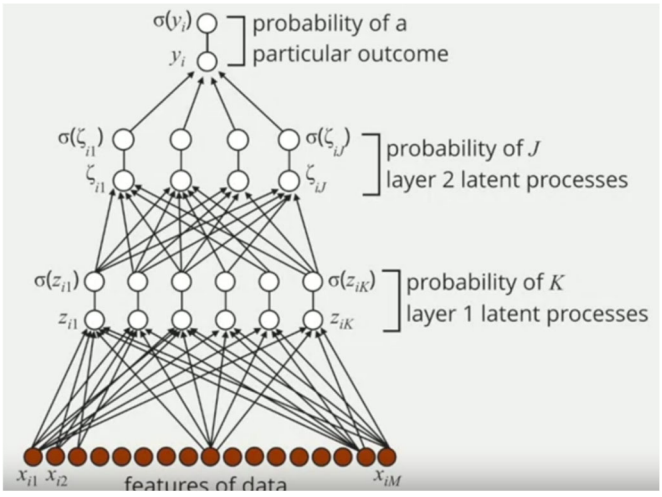
1.3.3. Multi-layer perceptron(MLP)

MLPs extend logistic regression by processing data through multiple layers with multiple features inside each layer. activation functions, such as sigmoid, tanh, and ReLU, is chosen to be applied accordingly to avoid collinearity issue. And then final probabilities are produced via softmax function for binary or muti-class classification tasks.



[3]

Figure 5. Extended Logistic Regression



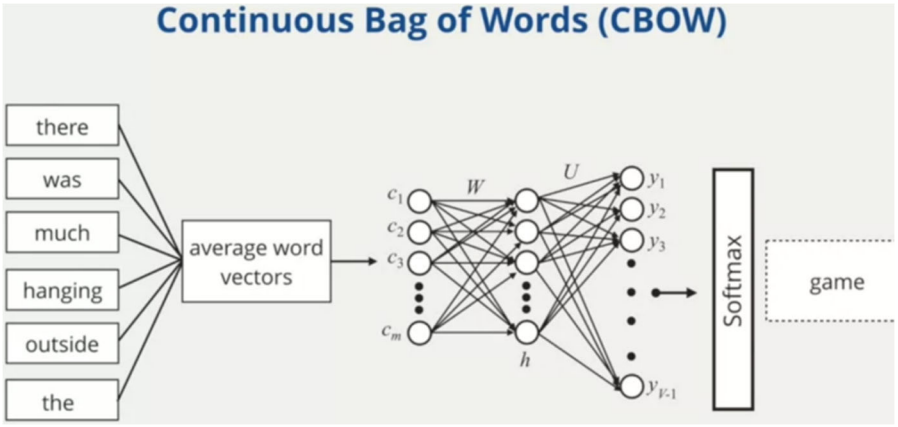
[3]

Figure 6. ELG Probability Calculation

ISSN: 2710-0588

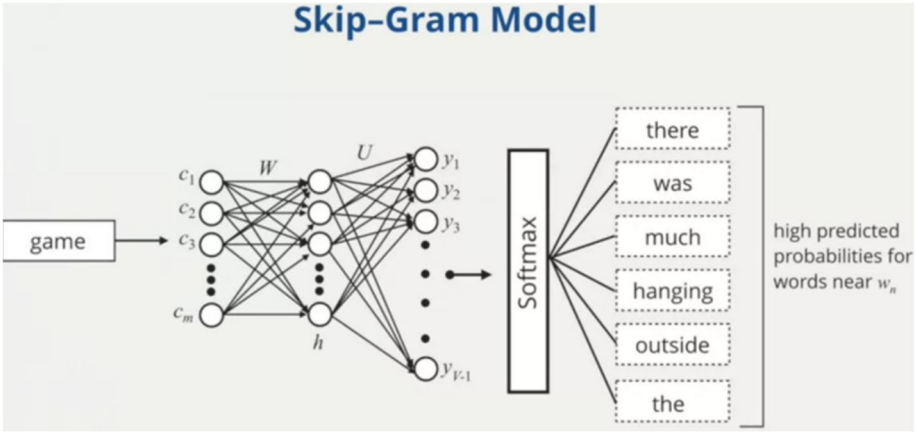
1.3.4. Word2vec Model

The Word2Vec model effectively combines word embeddings with neural network architectures, enabling two main approaches: predicting a target word based on its surrounding context words (CBOW) and predicting the surrounding context words from a given target word (Skip-Gram). However, this model has certain limitations. One major drawback is that it does not account for word order, which can be crucial for semantic understanding. Additionally, it struggles with handling polysemous words (words with multiple meanings) and cannot effectively deal with out-of-vocabulary or newly emerging words, indicating areas that require further improvement.



[3]

Figure 7. Illustration of CBOW



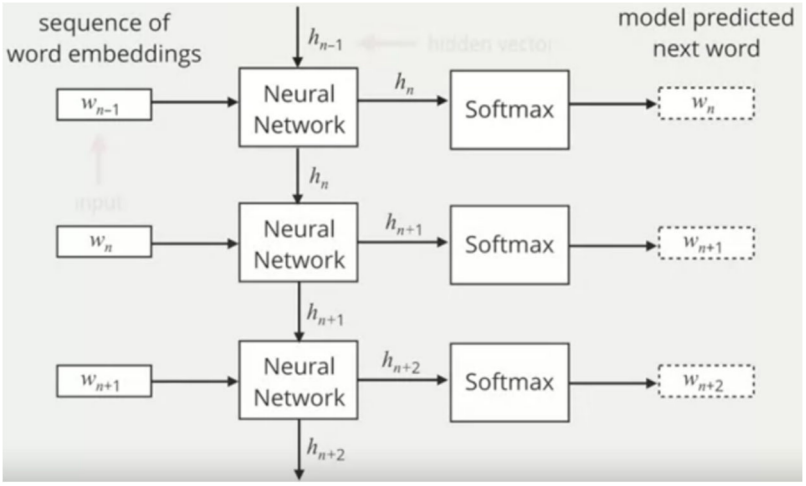
[3]

Figure 8. Illustration of Skip-Gram Model

1.4. Time Series models: RNN&LSTM:

The RNN (Recurrent Neural Network) model leverages a unidirectional chain-like architecture that effectively captures the influence of word order on semantic understanding. It introduces the concept of time steps (time_step). For example, if time_step = 3, the model uses the values of the preceding three tokens to predict the next token. Then, it shifts forward, using the last two tokens along with the newly predicted token to predict the following token, and this process continues iteratively.

ISSN: 2710-0588



[3]

Figure 9. Illustration of RNN [3]

LSTM (Long Short-Term Memory) is a specialized type of recurrent neural network designed to address the problems of gradient vanishing and gradient exploding commonly encountered in standard RNNs. LSTM introduces memory cells along with three types of gates—forget gate, input gate, and output gate—which selectively retain past information and incorporate new information. This mechanism allows the model to more effectively capture long-term dependencies in sequential data. However, LSTMs have notable drawbacks: they are computationally intensive, and because they process tokens sequentially, they are difficult to parallelize.

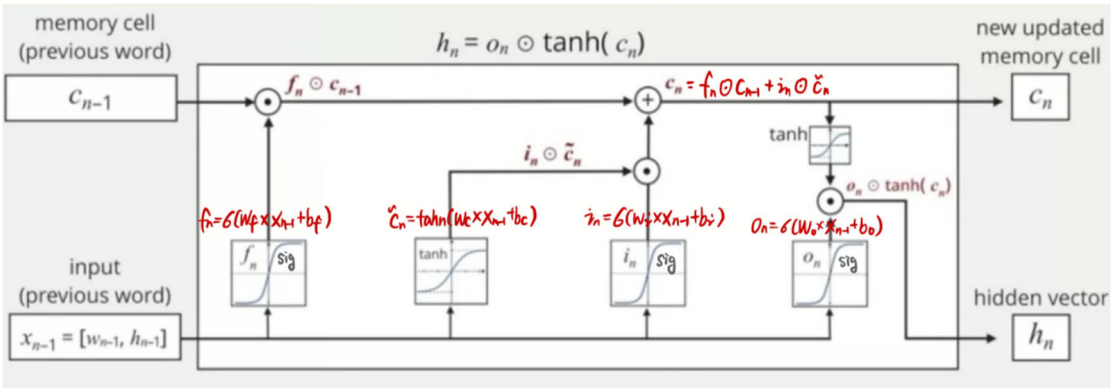


Figure 10. Illustration of LSTM [3]

1.5. Pre-trained Model Development and NLP Downstream Applications

"Pre-training falls under the category of transfer learning." For simpler models, a common approach is to randomly initialize parameters and then optimize them through (stochastic) gradient descent to minimize the loss function. Pre-training, on the other hand, involves first training the model on a large corpus to obtain a set of generalized parameters, which can then be further fine-tuned for specific tasks as needed. As shown in the figure, the introduction of Word2Vec in 2013 marked the beginning of the pre-training paradigm, while the proposal of the Transformer model in 2017 laid the foundation for the emergence of a series of state-of-the-art large-scale models.

ISSN: 2710-0588

Figure 11. NLP Model development [4]

Natural Language Processing (NLP) refers to enabling machines to understand human language, process it, and complete various downstream tasks. Common NLP tasks include sentiment analysis, named entity recognition (NER), text summarization, and text similarity, among others.

The primary differences among these tasks lie in the format of inputs and outputs, which in turn determine the choice of models.

Pre-trained models can generally be divided into two categories:

Auto-encoding models for NLU (Natural Language Understanding): These models mask certain tokens in the input and leverage contextual information to reconstruct the original tokens. They primarily use the encoder component of the Transformer architecture, with BERT being the most representative example.

Auto-decoding models for NLG (Natural Language Generation): These models predict the next token based on preceding tokens (or, in some cases, predict preceding tokens based on succeeding ones), but they can only consider information in a single direction. They mainly utilize the decoder component of the Transformer architecture, with GPT being the typical example[5].

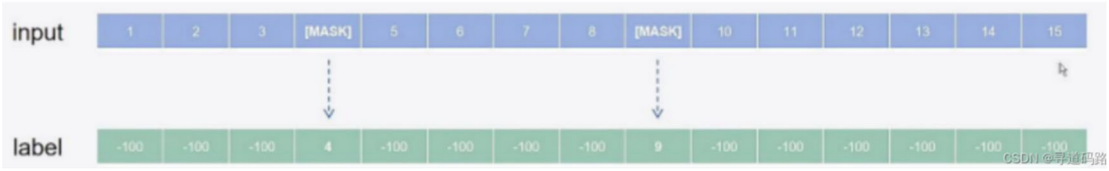


Figure 12. Illustration of Auto-Encoding

Figure 13. Illustration of Auto-decoding

Different types of models exhibit varying effectiveness when handling downstream tasks. Auto-encoding models are better suited for text understanding tasks such as sentiment analysis and named entity recognition. In contrast, auto-regressive models excel at generation-oriented tasks, such as text summarization, conversational question answering, and other generative applications.

ISSN: 2710-0588

2. State-of-the-Art Models

2.1. Transformer Architecture

The Transformer architecture primarily consists of two components: an encoder (sequence encoder) and a decoder (sequence decoder). It is also known as a seq2seq model, as it can take an input sequence of text and generate an output sequence. An encoder block can include K identical but independently trained encoder layers, and similarly, a decoder block can include J identical but independently trained decoder layers.

As illustrated, within the encoder, a sentence is first tokenized and each token is transformed into a d-dimensional word embedding. At this stage, these embeddings only represent the meaning of individual tokens. In the next step, positional embeddings are added to encode the position of each token within the sentence. By summing the word embedding and positional embedding vectors, we obtain representations that capture both semantic meaning and positional information. These vectors are then passed through the attention network, added back to the original vectors (residual connection), and normalized to produce updated vectors that effectively encode sentence-level meaning and contextual relationships. Finally, these vectors are processed through a fully connected (feed-forward) layer and forwarded to the specific downstream task. This describes the basic workflow of a single encoder in a Transformer[6].

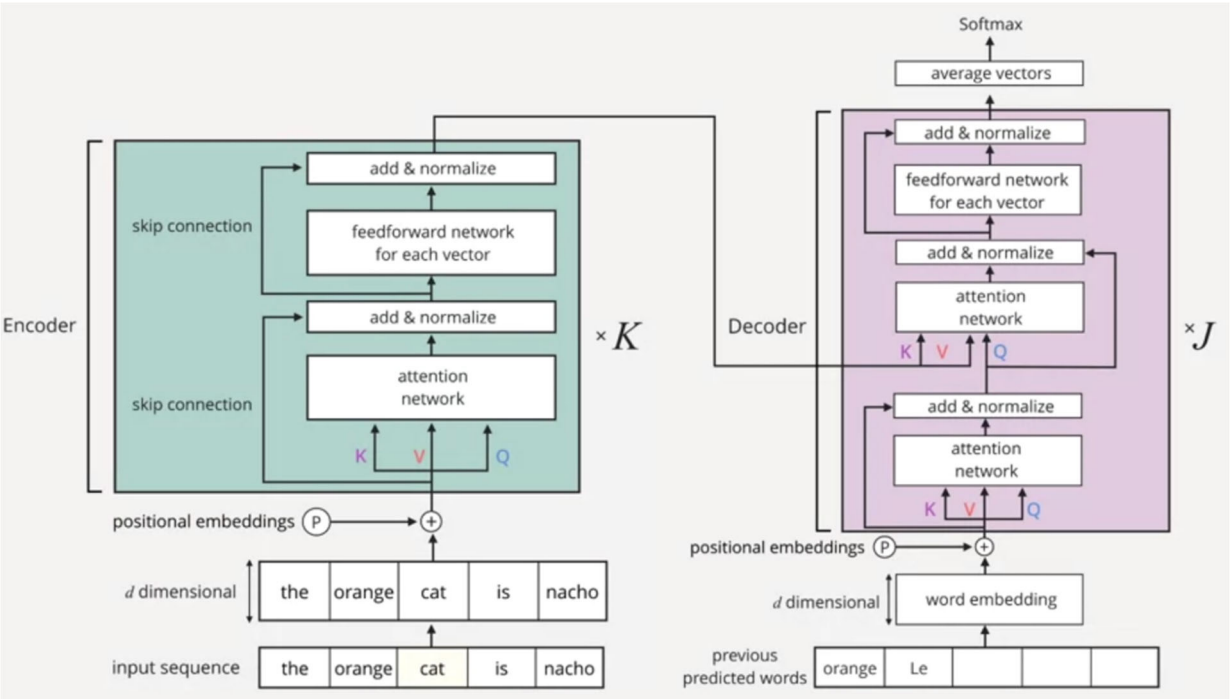
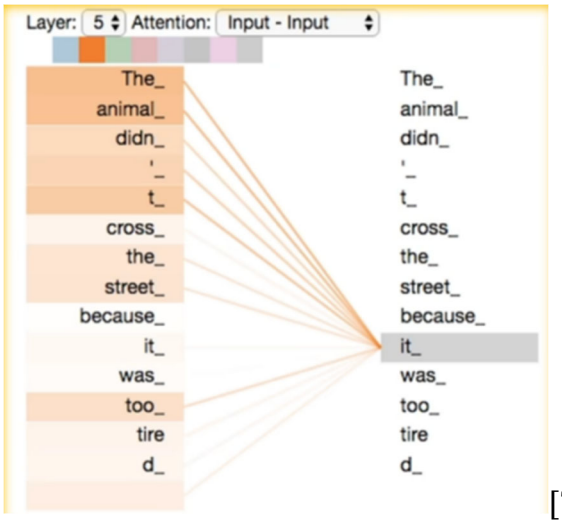


Figure 14. Illustration of Transformer [3]

From the encoder's workflow, it is clear that the attention mechanism plays a critical role, as it allows the model to understand contextual relationships between tokens and enables parallel computation, significantly reducing computational costs. To illustrate the principle, consider determining the referent of the word *"it"* in a sentence. The attention mechanism calculates a probability distribution over other words based on their relevance to *"it"*, with the highest probability indicating the most likely referent. After this step, the updated word vector incorporates the full contextual meaning of the sentence.

ISSN: 2710-0588



[7]

Figure 15. Illustration of Self-attention

Inner product: $c_1 \cdot c_k = \sum_{i=1}^{d} c_{1,i} \times c_{k,i}$

Take exponentiation to make the inner product all greater than zero.
Calculate the relative similarity between word k to word i:

$$r_{k \to i} = \frac{\exp(c_k \cdot c_i)}{\exp(c_k \cdot c_1) + \exp(c_k \cdot c_2) + \exp(c_k \cdot c_3) + \ldots + \exp(c_k \cdot c_N)}$$

$$\tilde{c_k} = \sum_{n=1}^{N} r_{k \to n} \times c_n \tag{13}$$

This summarizes the fundamental principle of the attention mechanism, where c_k is represented as the Query (Q), other token vectors c_i in the sentence are treated as Keys (K), and c_i in the case of self-attention, they also serve as Values (V).
The decoder structure shares many similarities with the encoder, but a key difference is that the decoder does not operate independently—it is connected to the encoder through cross-attention. The input sentence is first processed by the encoder to produce a set of output vectors, which are used as K and V in the decoder. These are combined with the partially decoded tokens (serving as Q) via the attention mechanism to produce updated word representations. Overall, the decoder uses both the encoder's full output and the previously decoded tokens to predict the next token in the output sequence.

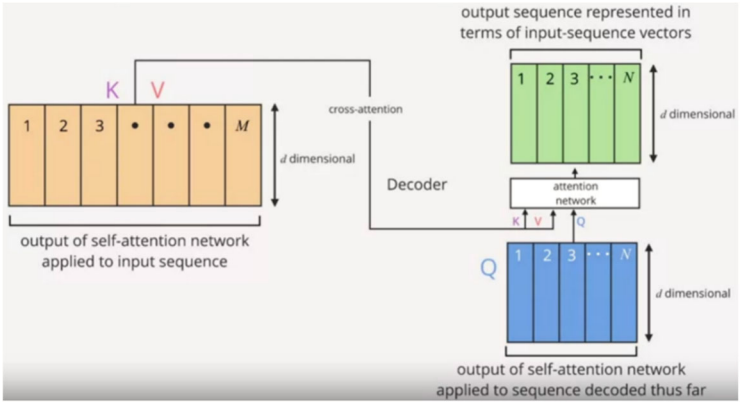


Figure 16. illustration of Decoder [3]

ISSN: 2710-0588

In practice, a more generalized form of attention, called **multi-headed attention**, is often employed. This approach introduces multiple projections and concatenations, allowing the model to capture different aspects of token relationships and features more effectively.

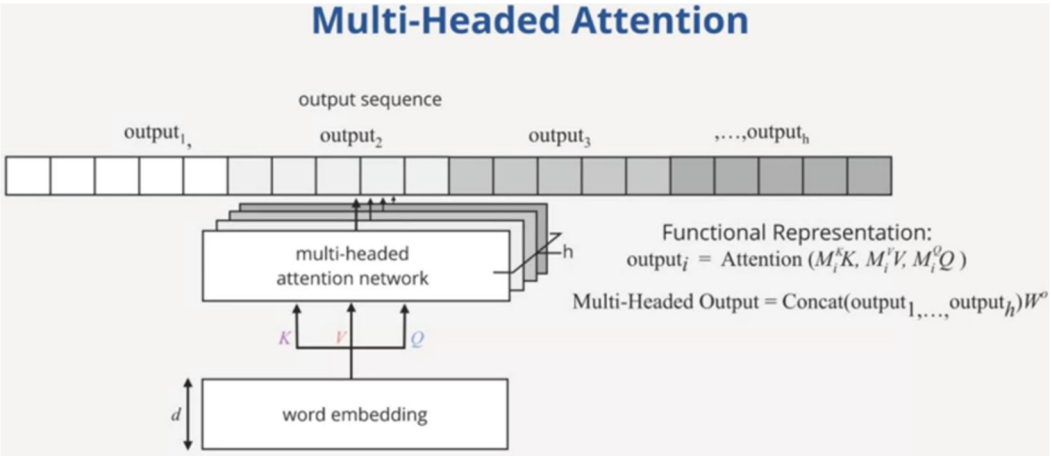


Figure 17. Illustration of Multi-headed Attention [3]

2.2. BERT Model Principles and Its Derivative Models

BERT (Bidirectional Encoder Representations from Transformers) as another pretrained model uses Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM masks words in a sentence and requires BERT to predict them, while NSP determines if one sentence follows another.

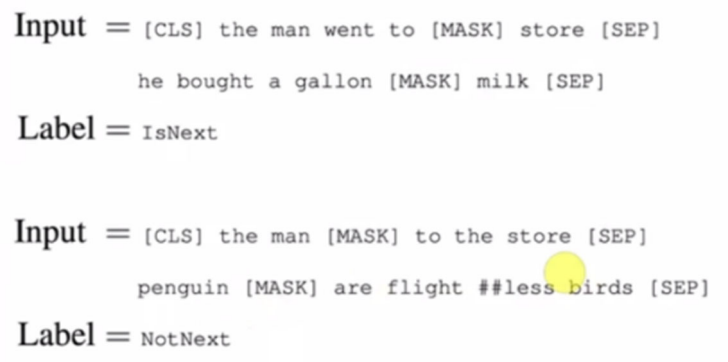


Figure 18. MLM and NSP

The BERT model can be understood as a process that takes word tokens as input and outputs high-quality word embeddings, with each token in the sentence represented by a 768-dimensional vector. BERT utilizes WordPiece embeddings, which consist of three components: input_ids, token_type_ids, and attention_mask.

input_ids represent the embeddings of the tokens.

token_type_ids serve as indicators to distinguish between two different sentences.

attention_mask differentiates actual tokens from padding tokens.

The first output position of BERT corresponds to the special token [CLS], which captures the overall meaning of the entire sentence and is commonly used for classification tasks. Below are four specific applications of BERT.

ISSN: 2710-0588

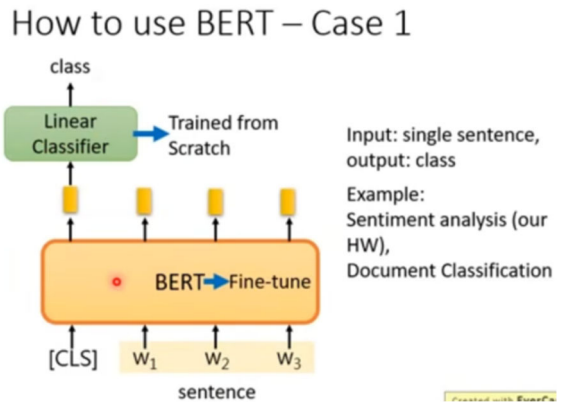


Figure 19. Bert Use Case 1 [8]

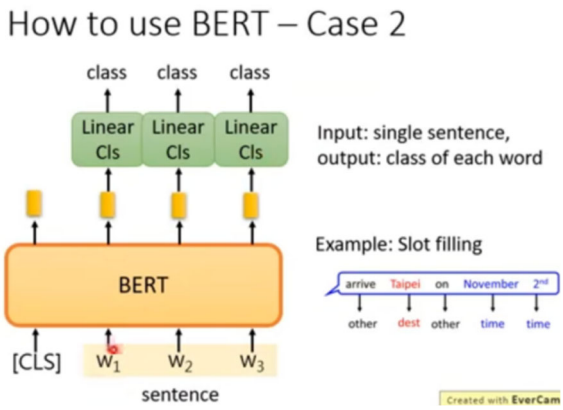


Figure 20. BERT Use Case 2 [8]

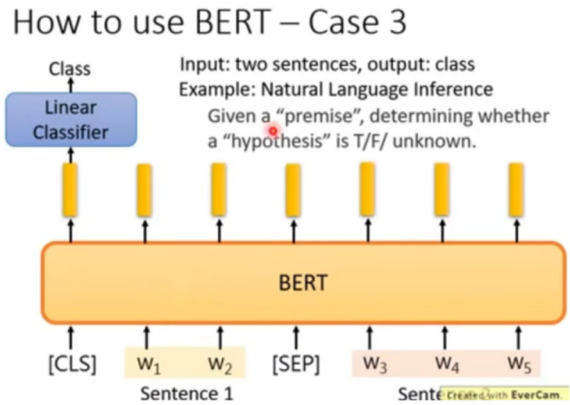


Figure 21. BERT Use Case 3 [8]

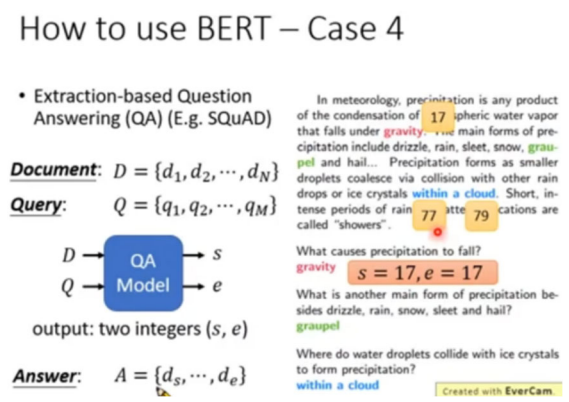


Figure 22. BERT Use Case 4.1 [8]

ISSN: 2710-0588

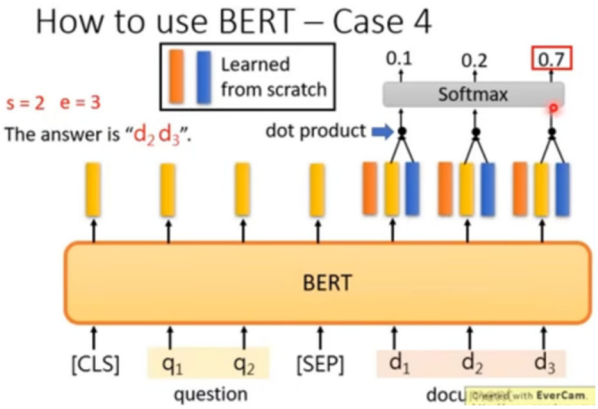


Figure 23. BERT Use Case 4.2 [8]

Other Derivative Models from BERT:

RoBERTa: A Robustly Optimized BERT Pretraining Approach

RoBERTa addresses the shortcomings of BERT's pretraining and is designed to be more robust. Its improvements include training on a much larger corpus, using bigger batch sizes, more training steps, and other updated hyperparameter configurations. It also removes the Next Sentence Prediction (NSP) objective, adopts full-sentence training (FULL-SENTENCES), and applies dynamic masking. This demonstrates that refining existing methods can sometimes be more effective than entirely new innovations.

ALBERT: A Lite BERT for Self-supervised Learning of Language Representations

As the name suggests, ALBERT was designed to make BERT lighter and more efficient. It reduces the number of parameters in BERT, replaces the NSP objective with Sentence Order Prediction (SOP), introduces n-gram masking, and uses the LAMB optimizer instead of AdamW—an optimizer particularly effective with large batch sizes. ALBERT shows that the size of a model's parameters is not an absolute measure of its capability.

SBERT: Sentence Embeddings using Siamese BERT-Networks

SBERT is designed for sentence-level tasks, such as generating fixed-dimensional sentence embeddings with rich semantic information for applications like text classification and semantic similarity computation. It is primarily obtained through fine-tuning BERT.

2.3. Model Application in Text Classification

2.3.1. Training a RoBERTa Model

A HuggingFace pre-trained 3-layer RoBERTa-wwm-ext model (hfl/rbt3) was used to classify China Unicom call-text data to determine whether a user shows a tendency to purchase services. Essentially, this involves performing a straightforward fine-tuning of the pre-trained rbt3 model using labeled China Unicom call-text data. After training, the model can provide relatively accurate predictions of users' service purchase intentions. However, the actual performance is less than satisfactory, mainly due to the limited size of the dataset and the lack of further fine-grained model adjustments.

General workflow for solving transformer-based NLP tasks (reference):

Step 1: Import relevant packages

Step 2: Load dataset → *Datasets*

Step 3: Split dataset → *Datasets*

Step 4: Preprocess dataset → *Tokenizer + Datasets*

Step 5: Create model → *Model*

Step 6: Define evaluation function → *Evaluate*

ISSN: 2710-0588

Step 7: Configure training parameters → *TrainingArguments*

Step 8: Create trainer → *Trainer + Data Collator*

Step 9: Model training, evaluation, and prediction (on dataset) → *Trainer*

Step 10: Generate predictions → *Pipeline*

2.3.2. Fine-Tuning RoBERTa with Additional Features

First, a RoBERTa model is fine-tuned using the *text* and *label* columns from the call-text dataset. After training, the pooled layer features from RoBERTa (768-dimensional vectors) are extracted. In parallel, additional tabular features (38 + 7 features) are derived from other columns of the dataset. These textual and tabular features are then combined and fed into a LightGBM model for final classification to predict whether a customer will subscribe to a service. This approach not only explores the model's capabilities more comprehensively but also leverages additional relevant data features from the call dataset to achieve better predictive performance.

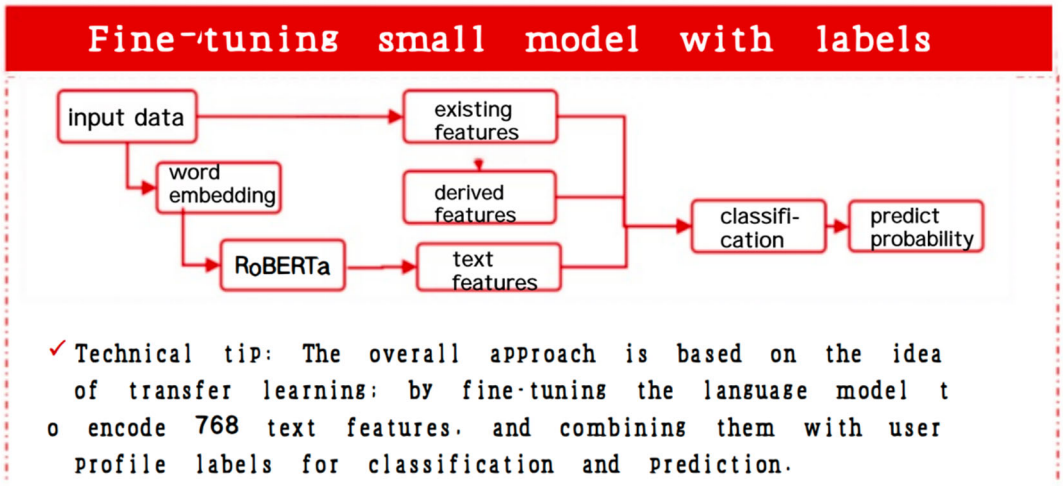


Figure 24. RoBERTa model Application

2.3.3. Fine-Tuning Large Models (ChatGLM + LoRA)

Large language models such as ChatGLM can be fine-tuned using lightweight parameter-efficient methods like LoRA (Low-Rank Adaptation), allowing targeted improvements without retraining the entire model from scratch.

2.4. Large Language Model Ecosystem

2.4.1. Base Models and Chat Models:

A base model is a foundational model trained on massive amounts of diverse text data, with the primary goal of predicting the next token in a sequence. However, the text it generates is not necessarily tailored to follow instructions or engage in dialogue. For example, when the base model qwen-1.8B was given the prompt "How is AI developing in recent days?" It generated a block of text similar to an article or essay. The response was informative but not conversational, showing that base models focus on continuing text rather than interpreting instructions or engaging interactively.

A chat model, on the other hand, is built on top of a base model by incorporating supervised fine-tuning (SFT) and reinforcement learning from human feedback (RLHF). Through reward-based optimization and reinforcement learning, chat models are refined to communicate in a way that is more natural and human-like, while continuously improving their response quality.

ISSN: 2710-0588

With the same example in the chat model, it generated a conversational answer that was structured more like a direct reply to the user's query. It not only summarized AI's progress and applications (e.g., in medicine, finance, transportation) but also used a tone closer to an assistant giving an explanation, showing its ability to follow instructions and maintain dialogue.

2.4.2. Multimodality

Multimodal AI is trained on multiple types of data, enabling them to handle not just text but also images, videos, audio, and other modalities. This allows them to perform a wide range of tasks, such as generating image captions or analyzing audio/video content. An example of such a model is ChatGPT-4, which incorporates multi-modal capabilities[9].

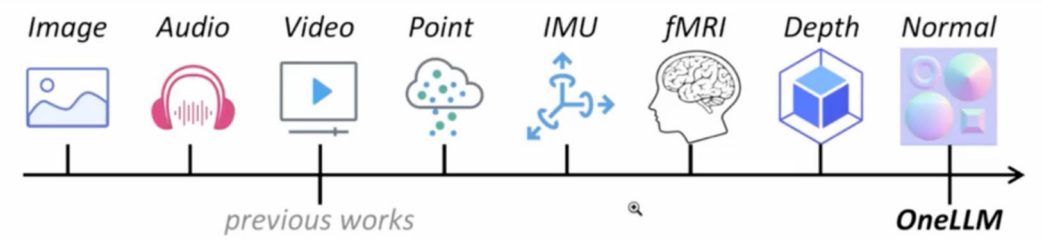


Figure 25. Multimodality

2.4.3. Agentic Models

An agentic model, which often includes an LLM as its core component, is a more comprehensive system composed of multiple modules that work collaboratively to achieve specific goals. Key components include:

Planning: Breaking down a large task into smaller sub-goals, performing self-correction, reflection, and refinement.

Memory: Maintaining both short-term memory (context from user interactions) and long-term memory (project-related knowledge).

Tool Use: Learning to call external APIs or tools to extend the model's capabilities.

2.4.4. Model Training Approaches

Several methods are used to train or adapt large models, including **prompts**, **fine-tuning**, and Retrieval-Augmented Generation (RAG).

Prompting: By crafting system messages or specific instructions, the model can be directed to adopt a certain persona or language style. For instance, if you want a model to act as a financial advisor, you could prompt it with *"You are a professional financial consultant. Provide a risk assessment of investing $10,000 in technology stocks."* The model immediately generates advice in that role, without additional training. Prompting is quick and flexible, but its effectiveness heavily depends on how well the instructions are written.

Fine-Tuning: Fine-tuning adapts a pre-trained model for a specific task using labeled data. For example, a company could fine-tune RoBERTa on thousands of telecom call transcripts labeled as "purchase intent" or "no purchase intent."It can be performed in two ways: full fine-tuning, where most or all parameters of the LLM are updated (which is memory-intensive and time-consuming), or parameter-efficient fine-tuning, where additional lightweight components are trained while the original model parameters remain frozen. A common approach is LoRA (Low-Rank Adaptation), which attaches trainable low-rank matrices to the model and merges them back during inference.

RAG (Retrieval-Augmented Generation): RAG enhances a model by linking it to an external knowledge base. A user query is first transformed into a vector representation, which is then used to retrieve relevant information from a vector database. Documents are typically split into

ISSN: 2710-0588

smaller chunks, indexed with encoders, and retrieved based on similarity to the query. The retrieved information is combined with the user query and passed to the LLM to generate a final, contextually accurate response. This allows LLMs to handle domain-specific tasks with greater expertise.

3. Conclusion

The landscape of large language models is expanding rapidly, making it impossible to fully grasp every model available. However, understanding the underlying principles is crucial. Building a foundational understanding often requires significant mathematical knowledge, which cannot be mastered in one go and must be reinforced iteratively. What's more, practical experience through coding and experimentation is essential. The ecosystem of tools and packages used in real-world applications is extensive, and becoming familiar with them is key to leveraging these models effectively.

References

[1] Wikipedia contributors. (2025). *Word n-gram language model*. In *Wikipedia*. Retrieved August 20, 2025, from https://en.wikipedia.org/wiki/Word_n-gram_language_model

[2] Supervised *Machine Learning: Regression and Classification (DeepLearning.ai)* [Online course]. Coursera. Retrieved June 20, 2025, from https://www.coursera.org/learn/machine-learning

[3] Introduction to *Machine Learning (Duke University)* [Online course]. Coursera. Retrieved June 27, 2025, from https://www.coursera.org/programs/coursera-for-university-of-rochester-mdtfv/learn/machine-learning-duke

[4] Baidu Wenku. (n.d.). Development of NLP models. Retrieved August 26, 2025, from https://wenku.baidu.com/view/d3760cc3de3383c4bb4cf7ec4afe04a1b071b0d3.html

[5] IBM. *NLP vs. NLU vs. NLG: What's the difference?*. IBM Think. Retrieved August 28, 2025, from https://www.ibm.com/think/topics/nlp-vs-nlu-vs-nlg

[6] Bilibili. (2021, June 14). Introduction to Natural Language Processing (NLP) [Video]. Bilibili. https://www.bilibili.com/video/BV14V4y1o7ff/

[7] Bilibili. (2021, October 12). Basic NLP course [Video]. Bilibili. https://www.bilibili.com/video/BV1Di4y1c7Zm/

[8] Bilibili. (2022, March 18). NLP and Machine Learning Applications [Video]. Bilibili. https://www.bilibili.com/video/BV1yU4y1E7Ns/

[9] Alibaba Cloud. (n.d.). *What is LLM (Large Language Model)?*. Alibaba Cloud. Retrieved August 29, 2025, from https://cn.aliyun.com/getting-started/what-is/what-is-llm