Deep Learning II: NLP and Transformer

PS690 Computational Methods in Social Science

Jiawei Fu
Department of Political Science
Duke University

November 4, 2025

Overview

Natural Language Processing (NLP) tasks

- A typical NLP pipeline starts with a tokenizer that splits the text into words or word fragments.
- Each token in the vocabulary is mapped to a unique D-dimensional word embedding.
- Finally, the embedding matrix representing the text is passed through a series of K transformers, called a **transformer model**.

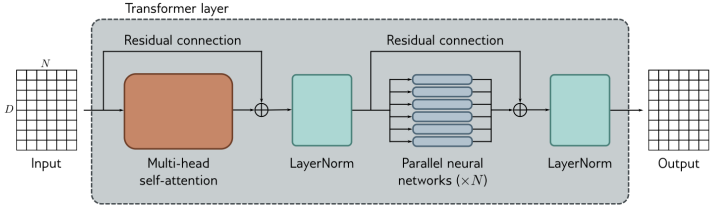


Figure: [Prince, 2023]

Natural Language Processing (NLP) tasks

- Transformers represent one of the most important developments in deep learning.
- One major advantage of transformers is that transfer learning is very effective.
- A transformer model can be trained on a large body of data and then the trained model can be applied to many downstream tasks using some form of fine-tuning.
- There are three types of transformer models.
  1. An encoder transforms the text embeddings into a representation that can support variety of tasks.
  2. A decoder predicts the next token to continue the input text.
  3. Encoder-decoders are used in sequence-to-sequence tasks, where one text string is converted into another (e.g., machine translation).
- Caution: The architecture of a transformer may appear complex—or even daunting—to newcomers, as it involves multiple components working together.
- But it's worth it: you'll be able to explain to your grandma and grandpa how ChatGPT works.

Motivation for Transformer

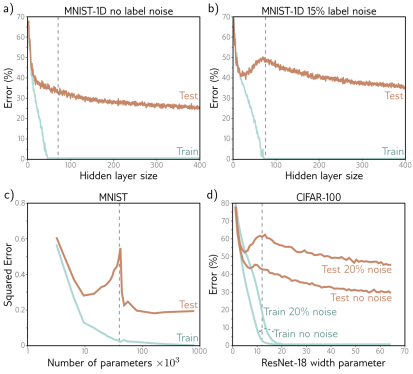
- Why do we need transformer? Why not use CNN?
- Convolutional networks, which are specialized for processing data that lie on a regular grid.
- They are particularly suited to processing images, which have a very large number of input variables, precluding the use of fully connected networks.
- Natural language is different from image. NLP task is also different from image task.
- Consider the following sentence:
  "The restaurant refused to serve me a ham sandwich because it only cooks vegetarian food. In the end, they just gave me two slices of bread. Their ambiance was just as good as the food and service."
- The goal is to design a network to process this text into a representation suitable for downstream tasks.
- For example, it might be used to classify the review as positive or negative, or to answer questions such as "Does the restaurant serve steak?".

Motivation for Transformer

- We found that language is ambiguous; it is unclear from the syntax alone that the pronoun it refers to the restaurant and not to the ham sandwich.
- To understand the text, the word it should somehow be connected to the word restaurant.
- In the parlance of transformers, the former word should pay attention to the latter. This implies that there must be connections between the words and that the strength of these connections will depend on the words themselves.
- They are based on a processing concept called attention, which allows a network to give different weights to different inputs, with weighting coefficients that themselves depend on the input values, thereby capturing powerful inductive biases related to sequential and other forms of data.
- Before we introduce attention mechanism, let us learn double descent and residual connection first.

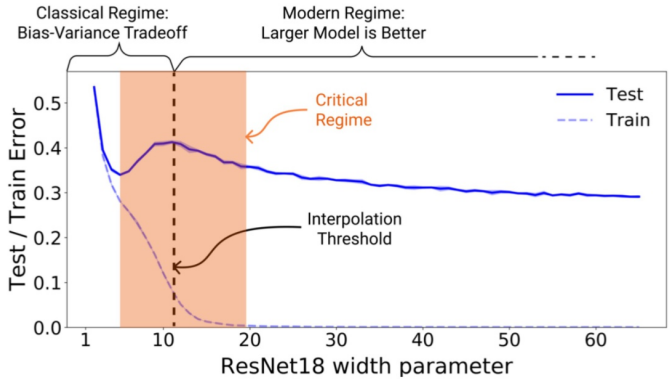
Double Descent

- In previous ML lectures, we have seen bias-variance trade-off. How about in the DL?
- We use 10,000 training examples in a MNIST-ID dataset, test with another 5,000 examples and examine the training and test performance as we increase the capacity (number of parameters) in the model.



Double Descent

- We observe double descent; It results from an interaction of two phenomena
  1. First, the test performance becomes temporarily worse when the model has just enough capacity to memorize the data.
  2. The test performance continues to improve with capacity even after the training performance is perfect.

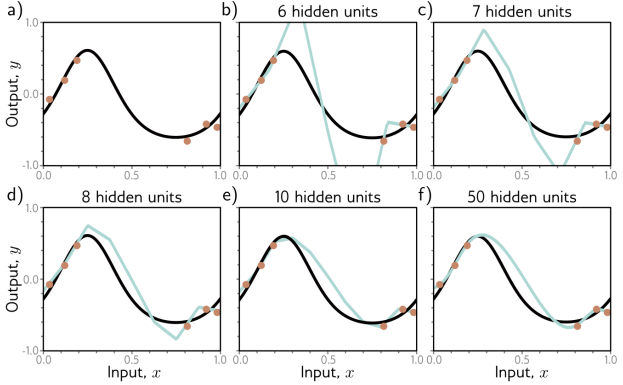


Double Descent

- It is unclear why performance should be better in the over-parameterized regime, given that there are now not even enough training data points to constrain the model parameters uniquely.
- We observe that once the model has enough capacity, then the model fits the training data almost perfectly.
- Therefore, further capacity cannot help the model fit the training data any better; any change must occur **between** the training points.

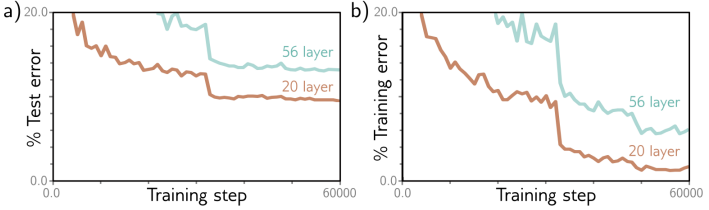
Double Descent

- The model's behavior between data points is critical because, in high-dimensional space, the training data are extremely **sparse** (curse of dimensionality).
- The putative explanation for double descent is that as we add capacity to the model, it interpolates between the nearest data points increasingly smoothly.



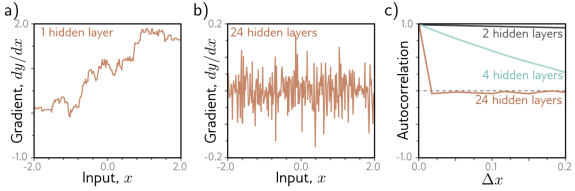
Shattered gradients

- In principle, we can add as many layers as we want. And we may expect adding layers improve performance.
- For example, the VGG network, which has eighteen layers, outperforms AlexNet, which has eight layers.
- However, image classification performance decreases as further layers are added.



Shattered gradients

- This phenomenon is not completely understood.
- One explanation for this phenomenon is called **shattered gradients**: correlation between gradients in standard networks decays exponentially with depth resulting in gradients that resemble white noise whereas.
- In general, we would expect gradients be "similar" for two nearby inputs as in (a).



- We observe that for a deep network, a tiny change in the input results in a completely different gradient in (b). Gradients of deep networks resemble white noise
- In (c), the autocorrelation of the gradient shows that nearby gradients become unrelated (have autocorrelation close to zero) for deep networks.

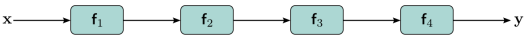
Residual Connections

- Training is difficult when gradients behave like white noise.
- An important modification to the architecture of neural networks that greatly assists in training very deep networks is that of **residual connections**.
- Standard neural network is sequential. For example, a three-layer network is defined by

$$
\begin{aligned}
h_1 &= f_1(x, \phi_1) \\
h_2 &= f_2(h_1, \phi_2) \\
h_3 &= f_3(h_2, \phi_3) \\
y &= f_4(h_3, \phi_4)
\end{aligned}
$$

  can be think of as a series of nested functions:

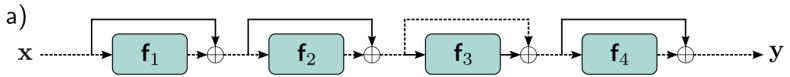
$$y = f_4[f_3[f_2[f_1[x, \phi_1], \phi_2], \phi_3], \phi_4]$$



Residual Connections

- Residual network is defined as:

$$
\begin{aligned}
h_1 &= x + f_1(x, \phi_1) \\
h_2 &= h_1 + f_2(h_1, \phi_2) \\
h_3 &= h_2 + f_3(h_2, \phi_3) \\
y &= h_3 + f_4(h_3, \phi_4)
\end{aligned}
$$



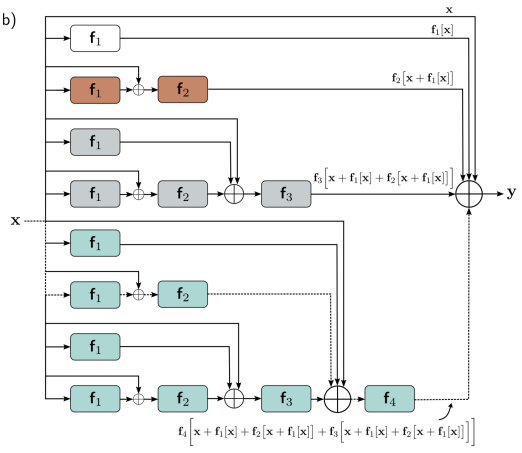
- Each additive combination of the input and the processed output is known as a residual block.
- This creates a direct path for the gradient to bypass these layers.

Residual Connections

- We can see that the final network output is a sum of the input and four smaller networks

$$
\begin{aligned}
y = x &+ f_1[x] \\
&+ f_2[x + f_1[x]] \\
&+ f_3[x + f_1[x] + f_2[x + f_1[x]]] \\
&+ f_4[x + f_1[x] + f_2[x + f_1[x]] \\
&\quad + f_3[x + f_1[x] + f_2[x + f_1[x]]]]
\end{aligned}
$$

- Therefore, residual connections turn the original network into an ensemble of these smaller networks; thus, The error surface is moderated.



Residual Connections

- The term 'residual' refers to the fact that in each block the function learns the residual between the identity map and the desired output:

$$f_k(h_{k-1}, \phi_k) = h_k - h_{-1}$$

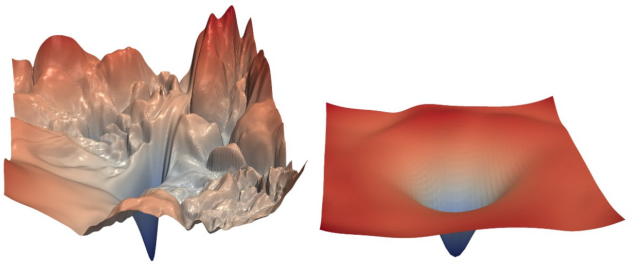


Figure 9.14 (a) A visualization of the error surface for a network with 56 layers. (b) The same network with the inclusion of residual connections, showing the smoothing effect that comes from the residual connections. [From Li *et al.* (2017) with permission.]

Attention

- The fundamental concept that underpins a transformer is *attention*.
- Consider the following two sentences:

I swam across the river to get to the other bank.
I walked across the road to get cash from the bank.

- Here the word 'bank' has different meanings in the two sentences.
- However, this can be detected only by looking at the context provided by other words in the sequence.
- In the first sentence, the words 'swam' and 'river' most strongly indicate that 'bank' refers to the side of a river, whereas in the second sentence, the word 'cash' is a strong indicator that 'bank' refers to a financial institution.
- We see that to determine the appropriate interpretation of 'bank', a neural network processing such a sentence should attend to, in other words rely more heavily on, specific words from the rest of the sequence.

Attention

- Moreover, we also see that the particular locations that should receive more attention depend on the input sequence itself.
- In the first sentence it is the second and fifth words that are important whereas in the second sentence it is the eighth word.
- In a standard neural network, different inputs will influence the output to different extents according to the values of the weights that multiply those inputs.
- Once the network is trained, however, those weights, and their associated inputs, are fixed.
- By contrast, attention uses weighting factors whose values depend on the specific input data.

Attention

- Suppose that we have a set of input tokens $x_1, ..., x_N$ in an embedding space and we want to map this to another set $y_1, ..., y_N$ having the same number of tokens but in a new embedding space that captures a richer semantic structure.
- With attention, y_n should depends on x_n and all others, and be stronger for those inputs x_m that are particularly important for determining the modified representation of y_n.
- A simple way to achieve this is using linear combination:

$$y_n = \sum_{m=1}^{N} a_{nm} x_m,$$

  where a_{nm} are called attention weights: n^{th} output pays to input x_m.
- We require $a_{nm} \geq 0$ and $\sum_{m=1}^{N} a_{nm} = 1$ so that if an output pays more attention to a particular input, this will be at the expense of paying less attention to the other inputs.

Self-attention

- The next question is how to determine the coefficients a_{nm}. Before we discuss this in detail, it is useful to first introduce some terminology taken from the field of information retrieval.
- Consider the problem of choosing which movie to watch in an online movie streaming service.
- One approach would be to associate each movie with a list of attributes describing things such as the genre (comedy, action, etc.), the names of the leading actors, the length of the movie, and so on.
- The user could then search through a catalogue to find a movie that matches their preferences.
- We could automate this by encoding the attributes of each movie in a vector called the *key*.
- The corresponding movie file itself is called a *value*.
- Similarly, the user could then provide their own personal vector of values for the desired attributes, which we call the *query*.

Self-attention

- We can think of the user 'attending' to the particular movie whose key most closely matches their query.
- Following the analogy with information retrieval, we can view each of the input vectors x_m as a value vector that will be used to create the output tokens.
- We also use the vector x_m directly as the key vector for input token n.
- That would be analogous to using the movie itself to summarize the characteristics of the movie.
- Finally, we can use x_n as the query vector for output y_n, which can then be compared to each of the key vectors.

Self-attention

- How much the token represented by x_n should attend to the token represented by x_m?
- The natural choice is to similarity between those two vectors.
- One simple measure of similarity is to take dot product and pass the results through a softmax function:

$$a_{nm} = softmax_m[x_n^T x_m] = \frac{exp(x_n^T x_m)}{\sum_{m'=1}^{N} exp(x_n^T x_{m'})}$$

- So in summary, each input vector x_n is transformed to a corresponding output vector y_n by taking a linear combination of input vectors in which the weight a_{nm} applied to input vector x_m is given by the softmax function.

Self-attention

- So far, the transformation from input vectors x_n to output vectors y_n is fixed and has no capacity to learn from data because it has no adjustable parameters.
- We can attach some adjustable parameters to query, key, and value, analogous to a 'layer' in a standard neural network.

$$Q = X\Omega_q, K = X\Omega_k, V = X\Omega_v$$

- Ω_k has dimensionality $D \times D_k$, where D_k is the length of the key vector.
- Ω_q must have the same dimensionality $D \times D_k$ as W_k so that we can form dot products between the query and key vectors. A typical choice is $D_k = D$.
- Ω_v is a matrix of size $D \times D_v$, where D_v governs the dimensionality of the output vectors.
- Note: although those transformation is linear, the overall self-attention computation is nonlinear due to attention weights.

Self-attention

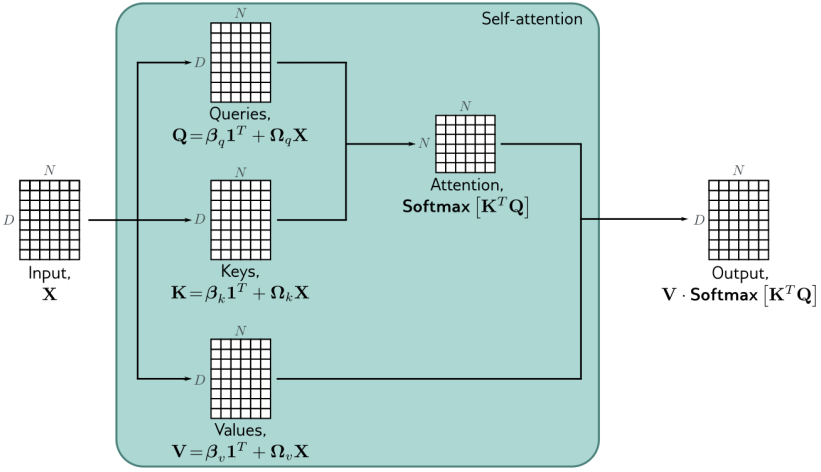
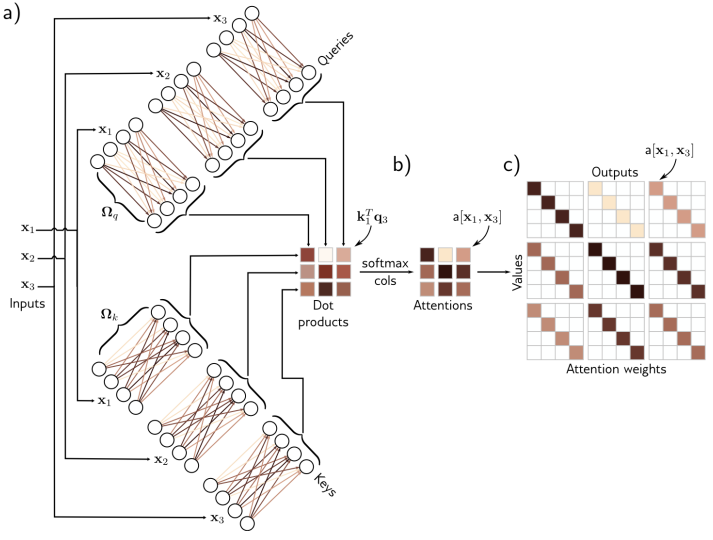


Figure: [Prince, 2023]

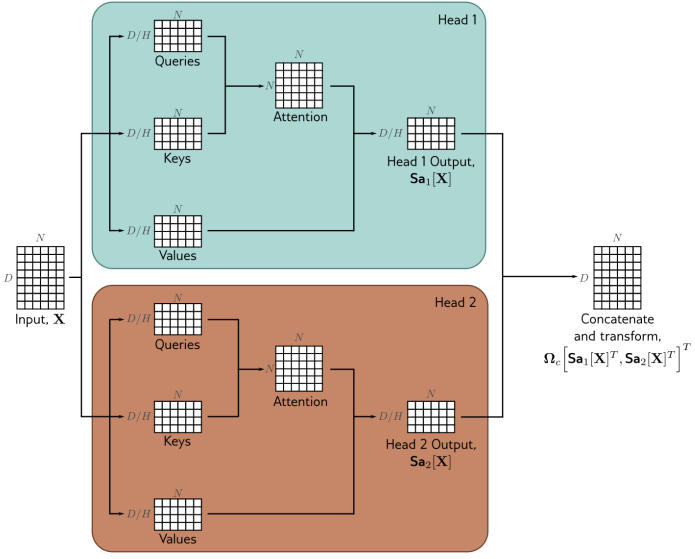
Self-attention



Multi-head attention

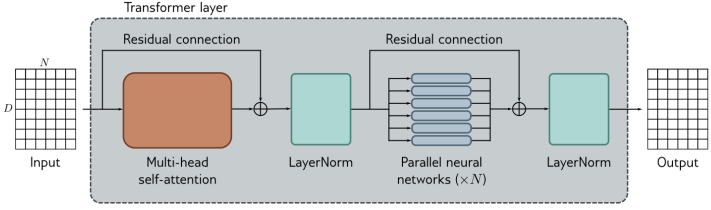
- There might be multiple patterns of attention that are relevant at the same time.
- In natural language, for example, some patterns might be relevant to tense whereas others might be associated with vocabulary.
- Using a single attention head can lead to averaging over these effects. Instead we can use multiple attention heads in parallel.
- We can see that it is the similar idea in the CNN with multiple filters.

Multi-head attention



Others

- Self-attention is just one part of a larger transformer mechanism.
- There are many other technical details we omit that are important for the performance:
  1. Positional encoding: order is important when the inputs correspond to the words in a sentence. The sentence The woman ate the raccoon has a different meaning than The raccoon ate the woman.
  2. Layer normalization: removes the need for the network to deal with extremely large or extremely small values; it is crucial for ensuring effective training.



Transformer Language Models

- The transformer processing layer is a highly flexible component for building powerful neural network models with broad applicability.
- Now, We explore the application of transformers to natural language. This has given rise to the development of massive neural networks known as large language models (LLMs).
- Transformers can be applied to many different kinds of language processing task, and can be grouped into three categories according to the form of the input and output data.
  1. In a problem such as sentiment analysis, we take a sequence of words as input and provide a single variable as output. Here a transformer is acting as an 'encoder' of the sequence.
  2. Other problems might take a single vector as input and generate a word sequence as output, for example if we wish to generate a text caption given an input image. In such cases the transformer functions as a 'decoder', generating a sequence as output.
  3. In sequence-to-sequence processing tasks, both the input and the output comprise a sequence of words, for example if our goal is to translate from one language to another. In this case, transformers are used in both encoder and decoder roles.

Decoder transformers

- Decoder transformer models can be used as generative models that create output sequences of tokens.
- For example, a class of models called GPT which stands for generative pre-trained transformer.

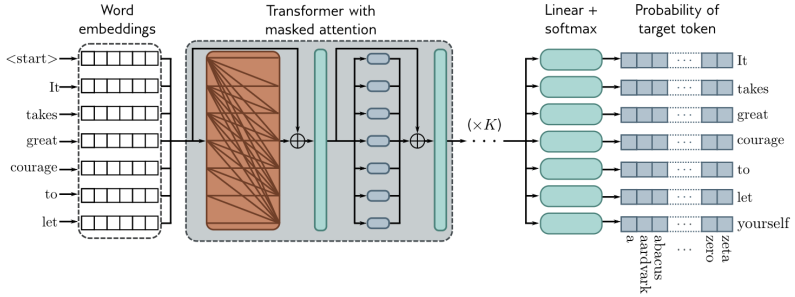


Figure: [Prince, 2023]

Prompting

- A user can interact with such models using a natural language dialogue, making them very accessible to broad audiences.
- A **prompt** is a text string that a user issues to a language model to get the model to do something useful.
- The performance of the model now depends on the form of the prompt, leading to a new field called prompt engineering

Basic Prompt Templates

Summarization	{input} ; `tldr;`
Translation	{input} ; `translate to French:`
Sentiment	{input}; `Overall, it was`
Fine-Grained-Sentiment	{input}; `What aspects were important in this review?`

Figure: [Jurafsky and Martin, 2025]

Prompt

- It is often possible to improve a prompt by including some labeled examples in the prompt template.
- We call such examples **demonstrations**.
- The task of prompting with examples is sometimes called **few-shot prompting**, as contrasted with **zero- shot prompting** which means instructions that don't include labeled examples.

Prompt

Definition: This task is about writing a correct answer for the reading comprehension task. Based on the information provided in a given passage, you should identify the shortest continuous text span from the passage that serves as an answer to the given question. Avoid answers that are incorrect or provides incomplete justification for the question.

Passage: Beyoncé Giselle Knowles-Carter (born September 4, 1981) is an American singer, songwriter, record producer and actress. Born and raised in Houston, Texas, she performed in various singing and dancing competitions as a child, and rose to fame in the late 1990s as lead singer of R&B girl-group Destiny's Child. Managed by her father, Mathew Knowles, the group became one of the world's best-selling girl groups of all time. Their hiatus saw the release of Beyoncé's debut album, Dangerously in Love (2003), which established her as a solo artist worldwide, earned five Grammy Awards and featured the Billboard Hot 100 number-one singles "Crazy in Love" and "Baby Boy".

Examples:
Q: In what city and state did Beyoncé grow up?
A: Houston, Texas

Q: What areas did Beyoncé compete in when she was growing up?
A: singing and dancing

Q: When did Beyoncé release Dangerously in Love?
A: 2003

Q: When did Beyoncé start becoming popular?
A:

Figure: [Jurafsky and Martin, 2025]

Prompt

- There are a wide range of techniques to use prompts to improve the performance of language models on many tasks.
- Another popular method is **chain-of-thought prompting** on difficult reasoning tasks.
- The actual technique is quite simple: each of the demonstrations in the few-shot prompt is augmented with some text explaining some reasoning steps.
- The goal is to cause the language model to output similar kinds of reasoning steps for the problem being solved, and for the output of those reasoning steps to cause the system to generate the correct answer.
- I also find that asking LLM generate **confidence level** can also improve the performance.

Prompt

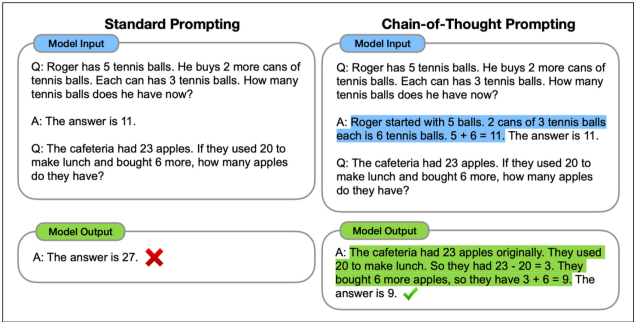


Figure: [Jurafsky and Martin, 2025]

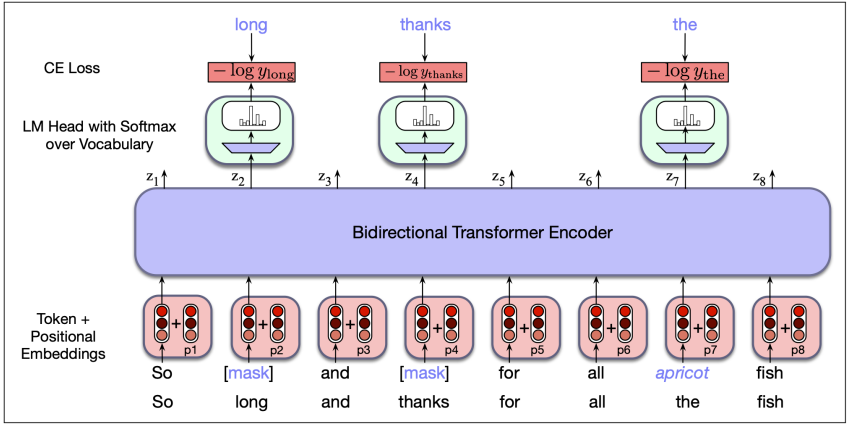
Encoder transformers

- For social science research, currently, encoder is more popular.
- An example of such a model is BERT (and its descendants like RoBERTa, or SpanBERT), which stands for bidirectional encoder representations from transformers.
- The term 'bidirectional' refers to the fact that the network sees words both before and after the masked word and can use both sources of information to make a prediction.
- Encoder models like BERT exploit **transfer learning**.
- During pre-training, the parameters of the transformer architecture are learned using self-supervision from a large corpus of text.
- The goal here is for the model to learn general information about the statistics of language.
- In the fine-tuning stage, the resulting network is adapted to solve a particular task using a smaller body of supervised training data.

Pre-training

- In the pre-training stage, the network is trained using self-supervision. This allows the use of enormous amounts of data without the need for manual labels.
- For BERT, the self-supervision task consists of predicting missing words from sentences from a large internet corpus.
- A randomly chosen subset of the tokens, say 15%, are replaced with a special token denoted $\langle \text{mask} \rangle$, or replaced with another randomly sampled token.
- The model is trained to predict the original inputs. This forces the transformer network to understand some syntax.
- For example, it might learn that the adjective red is often found before nouns like house or car but never before a verb like shout. It also allows the model to learn superficial common sense about the world.

Pre-training

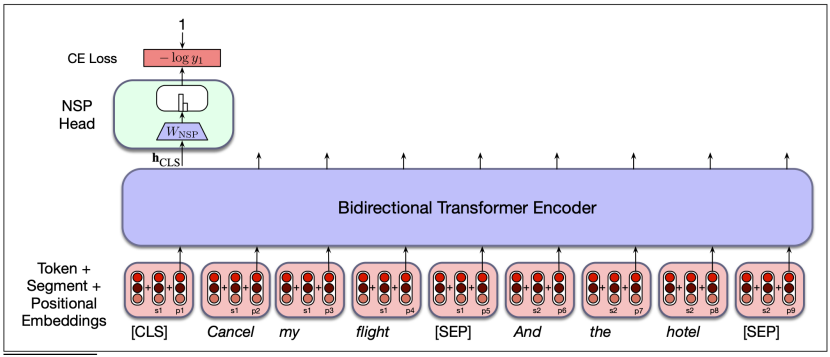


- Training corpora: the common crawl (automatically-crawled web pages), The Pile (text scraped from web, books and Wikipedia).

Pre-training

- The focus of mask-based learning is on predicting words from surrounding contexts with the goal of producing effective word-level representations
- However, an important class of applications involves determining the relationship between pairs of sentences.
- To capture the kind of knowledge required for applications such as these, some models in the BERT family include a second learning objective called Next Sentence Prediction (NSP).
- In BERT, 50% of the training pairs consisted of positive pairs, and in the other 50% the second sentence of a pair was randomly selected from elsewhere in the corpus.
- The loss is based on how well the model can distinguish true pairs from random pairs.
- After tokenizing the input with the subword model, the token [CLS] is prepended to the input sentence pair, and the token [SEP] is placed between the sentences and after the final token of the second sentence.

Pre-training

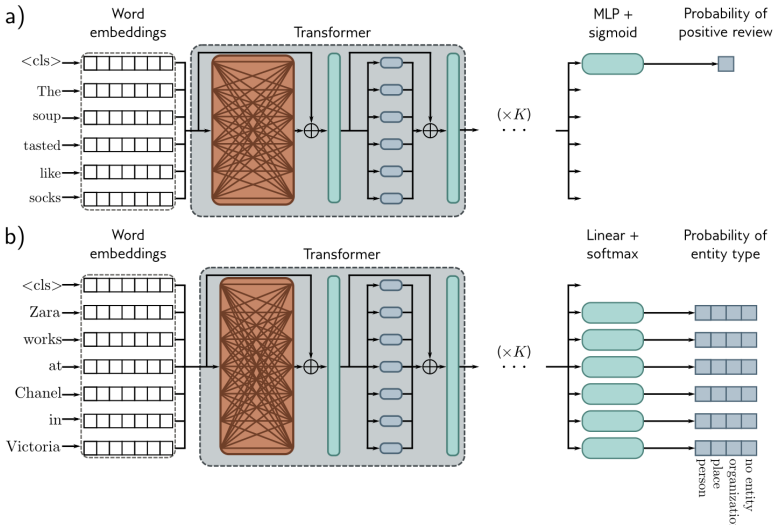


- The embedding of [CLS] basically holds the aggregate representation of all the tokens.

Fine-tuning

- Given a pretrained language model, we can think of the sequence of model outputs as constituting contextual embeddings for each token in the input.
- Once the encoder model is trained it can then be fine-tuned for a variety of different tasks.
- The fine-tuning can be done by adding extra layers to the outputs of the network or by replacing the last few layers with fresh parameters and then using the labelled data to train these final layers.
- For example, **Text classification** like **sentiment analysis**
  - The output vector in the final layer of the model for the [CLS] input represents the entire input sequence and serves as the input to a classifier head, a logistic regression or neural network classifier.
- For example, **Word classification** like **named entity recognition**
  - each input embedding is mapped to entity types.

Fine-tuning


a)
Word embeddings
Transformer
MLP + sigmoid
Probability of positive review
<cls>
The
soup
tasted
like
socks
(×K)
b)
Word embeddings
Transformer
Linear + softmax
Probability of entity type
<cls>
Zara
works
at
Chanel
in
Victoria
(×K)
person
place
organization
no entity

Fine-tuning

- During the fine-tuning stage, the weights and biases in the main model can either be left unchanged or be allowed to undergo small levels of adaptation.
- Typically the cost of the fine-tuning is small compared to that of pre-training.
- A trained over-parameterized model has a low intrinsic dimensionality with respect to fine-tuning, meaning that changes in the model parameters during fine-tuning lie on a manifold whose dimensionality is much smaller than the total number of learnable parameters in the model.

References


Jurafsky, D. and Martin, J. H. (2025).
Speech and language processing 3rd edition draft.
Online Draft.


Prince, S. J. (2023).
Understanding deep learning.
MIT press.