Understanding Input Embedding In Transformers

Introduction

When processing natural language, neural networks cannot directly interpret raw text. Instead, words, subwords, or characters must be converted into numerical representations before being input into a machine learning model.

The Input Embedding Layer facilitates this conversion by transforming tokenized words into dense numerical vectors that encode their semantic meaning and contextual information.

This article covers:

- The definition and significance of input embeddings.
- The process of converting tokenized words into embeddings.
- The functioning of embedding layers in Transformers such as BERT, GPT, and T5.
- The differences between learned embeddings and pretrained embeddings.

---

1. Definition of Input Embedding

Purpose and Necessity

Input embedding refers to the process of mapping each token (word, subword, or character)

to a fixed-length vector representation in a continuous numerical space.

Why is Input Embedding Important?

- Neural networks require numerical input rather than textual representations.
- Token IDs alone (e.g., 101, 2345, 5678) lack semantic meaning.
- Embeddings capture word relationships, enabling models to understand context effectively.

Example: Given the sentence:

```
"The cat sat on the mat."
```

Tokenization results in:

```
["The", "cat", "sat", "on", "the", "mat"]
```

Each token is mapped to a corresponding Token ID:

```
[101, 2345, 5678, 3456, 101, 6789]
```

The Embedding Layer then converts these token IDs into numerical vectors:

```
[
```

```
    [0.2, 0.8, -0.5, 0.1],  # "The"
    [0.5, 0.3, 0.9, -0.7],  # "cat"
    [0.1, 0.9, 0.4, -0.3],  # "sat"
    ...
]
```

At this stage, each token is represented numerically, allowing further processing by the model.

---

2. Generation of Input Embeddings

The embedding layer functions as a lookup table that stores vector representations of words in a high-dimensional space.

Steps in the Embedding Process

1. Tokenization → Splitting text into tokens.
2. Mapping Tokens to IDs → Assigning unique numerical identifiers to tokens based on a predefined vocabulary.
3. Embedding Lookup → Replacing token IDs with their corresponding dense vector representations.

Example: Mapping Token IDs to Embeddings

Token	Token ID	Embedding Vector
“The”	101	`[0.2, 0.8, -0.5, 0.1]`
“cat”	2345	`[0.5, 0.3, 0.9, -0.7]`
“sat”	5678	`[0.1, 0.9, 0.4, -0.3]`

---

3. Input Embeddings in Transformer Models

In Transformer architectures such as BERT and GPT, input embedding is composed of:

1. Token Embedding → Represents word meanings.
2. Positional Encoding → Adds positional information to preserve word order.
3. Segment Embedding (used in BERT) → Differentiates between multiple sentences.

Formula for Final Input Representation

```
Final Input Vector = Token Embedding + Positional Encoding + Segment
Embedding
```

Example in a Transformer Model:

```
"The cat sat" → Token Embeddings → Positional Encoding → Final
Transformer Input
```

---

4. Learned vs. Pretrained Embeddings

Two primary types of embeddings are used in NLP:

Embedding Type	Description	Examples
Pretrained Embeddings	Word vectors trained separately on large datasets.	Word2Vec, GloVe, FastText
Learned Embeddings	Word embeddings adjusted during model training.	BERT, GPT, T5

Differences Between Pretrained and Learned Embeddings

- Pretrained embeddings remain fixed after training, whereas learned embeddings update dynamically during model training.
- Transformers predominantly utilize learned embeddings, enabling adaptability to varying contexts.

---

5. Benefits of Input Embeddings in NLP

- Preserves Word Semantics → Facilitates understanding of word relationships.

- Efficient Representation of Large Vocabularies → Supports extensive word coverage.
- Enhances Contextual Awareness → Assists models in grasping sentence structures.
- Computational Efficiency → More compact than one-hot encoding, improving processing speed.

---

6. Applications of Input Embeddings

1. Conversational AI

- Utilized in models such as ChatGPT, Google Bard, and virtual assistants.

2. Search Engines & Recommendation Systems

- Applied in Google Search, YouTube recommendations, and content filtering.

3. Machine Translation

- Enables translation models (e.g., Google Translate) to encode multilingual text.

4. AI Coding Assistants

- Used in AI-powered tools like GitHub Copilot and AlphaCode for code generation.

---

7. Conclusion

The Input Embedding Layer is fundamental to modern NLP models, facilitating the transformation of raw text into meaningful numerical representations.

Key Takeaways

- Input embeddings map tokens to dense numerical vectors.
- Embedding layers store semantic word representations.
- Transformers rely on dynamically updated learned embeddings.
- Positional encoding helps preserve word order.
- Embeddings play a vital role in diverse NLP applications.

For a deeper dive into Transformers and advanced NLP concepts, stay tuned for further articles.

Further Reading & References

 Research Paper: Attention Is All You Need (Vaswani et al., 2017)
 Illustrated Transformer Guide: Jay Alammar’s Illustrated Transformer
 Hugging Face Transformer Library: Hugging Face Guide

For more insights into NLP and AI advancements, visit EasyExamNotes.com.

 Have questions? Share your thoughts in the comments.

 Follow for the latest AI and NLP updates.

Related posts:

1. Transformer Architecture in LLM
2. Positional Encoding in Transformers
3. Multi-Head Attention in Transformers
4. Artificial Intelligence Intelligence Tutorial for Beginners
5. Difference between Supervised vs Unsupervised vs Reinforcement learning
6. What is training data in Machine learning
7. What other technologies do I need to master AI?
8. How Artificial Intelligence (AI) Impacts Your Daily Life ?
9. Like machine learning, what are other approaches in AI ?
10. Best First Search in AI
11. Heuristic Search Algorithm
12. Hill Climbing in AI
13. A* and AO* Search Algorithm
14. Knowledge Representation in AI

15. Propositional Logic and Predicate Logic
16. Resolution and refutation in AI
17. Deduction, theorem proving and inferencing in AI
18. Monotonic and non-monotonic reasoning in AI
19. Probabilistic reasoning in AI
20. Bayes' Theorem
21. Artificial Intelligence Short exam Notes
22. Why 512 Dimensions in Transformer Model Architecture
23. Self Attention in Transformer