December 2023

Approaches for Incorporating a Variety of Metadata in Transformer Operation

Antonio Gulli

Afroz Mohiuddin

Samer Hassan

Approaches for Incorporating a Variety of Metadata in Transformer Operation

ABSTRACT

A plain transformer model typically leverages only one piece of metadata - position encoding - directly in the transformer model. The use of transformers typically involves expensive and complex external scaffolding before or after output generation to avoid issues such as hallucination, irrelevance, etc. This disclosure describes techniques to incorporate a variety of metadata types into the native architecture of transformer models. The additional signals can help avoid hallucinations, improve relevance, and minimize the need of expensive external scaffolding. Generalizing transformer operation to incorporate a diversity of metadata can be achieved in various ways such as adding a metadata embedding layer, conditioning self-attention on the metadata, conditioning with gated self-attention, employing a different encoder-decoder architecture, etc. Different types of metadata can help in different ways to improve the quality of the output generated by the transformer and reduce hallucinations. The techniques described in this disclosure can also support multimodal data, such as images, audio, video, or text, etc., with the metadata used representing the specific mode.

KEYWORDS

- Transformer model
- Large language model (LLM)
- Natural language processing (NLP)
- Positional encoding
- Metadata embedding
- Conditional self-attention
- Gated self-attention
- Hallucination
- Prompt augmentation
- Output validation

BACKGROUND

Transformers are a type of neural network architecture that have become the state of the art for many natural language processing (NLP) tasks such as machine translation, text summarization, question answering, etc. Transformers are based on the self-attention mechanism that allows learning long-range dependencies in sequential data. Transformers employ a special embedding layer to learn a representation of the position of each word in the text. This representation is used to condition the self-attention mechanism. Such an operation enables the model to pay attention to different parts of the text based on the position without sacrificing the ability to learn long-range dependencies or generalize to new text. Currently, the models typically do not incorporate any metadata other than positional encoding.

The use of transformers typically involves additional external scaffolding before or after output generation. The scaffolding is employed to augment prompt pre-generation and/or to perform output validation to avoid issues such as hallucination, irrelevance, etc. Augmenting prompt pre-generation can be expensive due to retrieval and limited prompt windows. Additionally, validating each sentence post-generation and discarding unvalidated ones can be expensive due to chunking and validation. As a result, the external scaffolding before or after generation can significantly increase the budget to produce the final results. Moreover, the scaffolding is complex, difficult to manage, with no explicit quality guarantees.

DESCRIPTION

This disclosure describes techniques to facilitate better incorporation of a variety of metadata types beyond positional encoding into the architecture of transformer models. The metadata can be used to change weights during model training by making these additional signals natively available at the time of inference. The additional signals can help avoid hallucinations, improve relevance of the results generated by the transformer, and minimize the need of expensive external scaffolding to validate the quality, accuracy, and relevance of the output.

Generalizing transformer operation to incorporate a diversity of metadata can be achieved in a variety of ways.

Metadata conditioning

Metadata conditioning plays a crucial role in ensuring that developers can guide and control artificial intelligence models with both precision and flexibility. By leveraging rich metadata, model behavior can be tailored to specific contexts and tasks, maximizing their effectiveness and mitigating potential biases. Metadata can be materialized as raw content associated with the input to the model, however the expressiveness of the metadata might bring bottlenecks. Alternatively, the metadata can be encoded externally before materializing it into the transformer metadata by learning compressed, universal representations via highly efficient embeddings. The same piece of metadata can be converted into different types of embeddings, each focusing on different aspects of the representation. This can enable capturing specific properties of the input metadata externally. These embedding-based summaries are then seamlessly integrated into the transformer as external inputs. This enables multiple advantages - effortless scaling and a host of additional benefits, including:

- **Enhanced Security:** LLM leakage, a potential threat of copied model parameters, is mitigated by leveraging external embeddings. Without access to these critical dependencies, a leaked model can suffer from diminished or even degraded performance. Thus, the techniques provide enhanced security.

- **Improved Robustness**: The interpretation of metadata by an LLM can vary considerably due to training data size and diversity. By training external representations on vast datasets, consistent and coherent metadata interpretation by the LLM can be ensured regardless of training data nuances.
- **Efficient Maintainability:** Regularly retraining LLMs to maintain world state incurs significant costs. External embeddings, being inexpensive and easily refreshed, enable seamless integration of updated metadata into existing LLMs. This approach minimizes retraining overhead and ensures a consistent embedding space for metadata.
- **Data synergies:** This approach can be even more powerful by leveraging the synergy between different metadata spaces. Valuable insights can be first extracted from one space, establishing a strong foundation of knowledge. Then, by incorporating additional, targeted learnings from other spaces, even for the same piece of information, additional benefits can be achieved with less data. Essentially, the data synergies enable building upon an initial knowledge based and refining it further with specific explorations.
- **Dynamic Adaptation**: The required level of metadata can be dynamically adjusted based on the task and user intent. For complex tasks, richer metadata may be necessary to achieve results, while simpler tasks might function perfectly well without it. This adaptive approach optimizes resource utilization and ensures the model operates efficiently across different situations.

- **Optimized Servability:** As serving large transformer models becomes increasingly expensive, externally encoded metadata allows for rapid retrieval. This precomputation lowers the serving cost of the LLM, resulting in smooth and efficient performance, scaling seamlessly to meet user demands.
- **Reduced Hallucinations:** Metadata acts as a safety net and can prevent the model from straying into unrealistic or nonsensical territory. By grounding model predictions in real-world data with multiple metadata signals, it can be ensured that the model outputs are not only relevant but also plausible.
- **Personalized Experiences**: By incorporating user-specific metadata (with user permission), artificial intelligence experiences can be personalized to individual preferences and needs. This can be instrumental in applications like personalized platforms, where customized content and recommendations are essential for optimal user engagement.

Adding a separate metadata embedding layer

This approach can enable the model to learn separate representations for the input sequence and the metadata. The added metadata embedding layer can learn a separate embedding for each piece of metadata. The metadata is first embedded into a vector space compatible with the input and output representations of the model. These embeddings can then be concatenated with the input sequence embeddings provided to the transformer encoder. The transformer can then learn to attend to the metadata features when computing the attention weights for each input token. The hidden state of the model is passed to the decoder which generates the output sequence by taking into account the hidden state as well as the metadata embeddings. The operation is depicted in Fig. 1

```
Input sequence:          [token1, token2, ..., tokenN]
Metadata:                [metadata1, metadata2, ..., metadataM]
Metadata embedding layer: [embedding1, embedding2, ..., embeddingM]
Output sequence:         [output1, output2, ..., outputP]
```

Fig. 1: Metadata embedding layer added to the Transformer operation.

Conditioning the self-attention mechanism on the metadata

This approach allows the model to attend to different parts of the input sequence based on the metadata. The self-attention mechanism can be conditioned on the metadata by concatenating the metadata embeddings with the query and key vectors. When computing the attention weights for each input and output token, the concatenation can enable the model to learn to attend to the metadata features directly without needing to embed them into a different vector space first. The operation is depicted in Fig. 2.

```
Input sequence:          [token1, token2, ..., tokenN]
Metadata:                [metadata1, metadata2, ..., metadataM]
Self-attention mechanism: [query, key, value]
Output sequence:         [output1, output2, ..., outputN]

Condition the self-attention mechanism on the metadata
Inputs:                  Query Q, Keys K, Values V, Metadata M
Outputs:                 Context vector C

Compute the self-attention scores:           S = $QK^T$
Compute the metadata-aware attention weights: A = $M^TS$
Compute the context vector:                  C = V*A
      Where:      Q: Query vector
                  K: Key vector
                  V: Value vector
                  M: Metadata vector
                  S: Self-attention scores
                  A: Metadata-aware attention weights
                  C: Context vector
```

Fig. 2: Using the metadata to condition the self-attention mechanism.

Conditioning with a gated self-attention mechanism

This approach can enable the model to control the parts of the metadata to attend and to focus tasks when they require specific types of metadata. A gated attention mechanism can be employed to compute a set of attention weights for each part of the metadata using a simple attention technique, such as dot-product attention, scaled dot-product attention, etc. The result can be used to gate the flow of information from the metadata to the model via a suitable gating function, such as a sigmoid, tanh, etc. The gated information forms the metadata used to update the internal state of the model. The operation is illustrated in Fig. 3.

```
Inputs:                  Query Q, Keys K, Values V, Metadata M
Outputs:                 Context vector C

Compute the self-attention scores:            $S = QK^T$
Gate the self-attention scores:               G = sigmoid(W_gS)
Compute the gated self-attention weights:     A = G*S
Compute the context vector:                   C = V*A
      Where:      Q:    Query vector
                  K:    Key vector
                  V:    Value vector
                  M:    Metadata vector
                  W_g:  Weight matrix for the gating function
                  S:    Self-attention scores
                  G:    Gated self-attention scores
                  A:    Gated self-attention weights
                  C:    Context vector
```

Fig. 3: Using the metadata to condition the self-attention mechanism in a gated manner.

Conditioning with dropout

Dropout metadata conditioning introduces a layer of controlled randomness into the training process of LLMs. By strategically removing either portions of metadata, or some of the metadata totally during training, dropout techniques can offer improved robustness to the absence of the metadata that the model was trained on. For example, during each training iteration, a portion of the metadata associated with the training data can be removed, e.g., randomly.

This dropout mechanism ensures that the model is not overly reliant on any single piece of information and can ensure that the model training results in the model developing broader, more generalizable learning patterns. The approach is effective because the training data inherently possess a certain level of redundancy. This means that different pieces of metadata convey similar or overlapping information. This redundancy enables inferring the missing information even when some metadata is dropped, mitigating potential overfitting and generalizing the model better. By leveraging data redundancy and the ability of the model to learn from incomplete information, dropout metadata conditioning can make the model suitable for deployment in scenarios where only part of the metadata exists and other metadata that the model was trained on is unavailable. While a small amount of accuracy loss may occur, this is easily outweighed by the substantial robustness offered by dropout metadata conditioning.

Using a different type of encoder-decoder architecture

This approach involves employing a different type of model architecture, such as a hierarchical encoder-decoder architecture. In such an architecture, the encoder is a stack of self-attention layers that can first encode the input sequence and the metadata into a latent representation. Each self-attention layer can learn to attend to different parts of the input sequence and the metadata and combine the attended representations to produce a new representation. The decoder can decode the output sequence conditioned on the encoded metadata. The operation is illustrated in Fig. 4.

```
Input sequence:     [token1, token2, ..., tokenN]
Metadata:           [metadata1, metadata2, ..., metadataM]
Encoder:            [encoder_output1, encoder_output2, ..., encoder_outputN]
Decoder:            [decoder_input1, decoder_input2, ..., decoder_inputN]
Output sequence:    [output1, output2, ..., outputN]
```

Fig. 4: Incorporating metadata in the input via a hierarchical encoder-decoder architecture

Applications of metadata conditioning

A vanilla transformer typically leverages at least one piece of metadata (position encoding that indicates the position of tokens in a sequence). The approaches described herein can generalize the approach in a systematic way to incorporate various other types of metadata such as those listed below. In various applications, user permission is obtained prior to use of such metadata. The different types of metadata can help in different ways to improve the quality of output generated by the transformer and reduce hallucinations:

- **Author:** The name of the person or organization that created the resource.
  - Improve quality: Generate more credible and informative text based on the author’s reputation and expertise.
  - Reduce hallucinations: Avoid generating text inconsistent with the author's writing style and domain knowledge.
- **Link:** A uniform resource locator (URL) that points to the resource.
  - Improve quality: Access full text of the resource for additional context and information.
  - Reduce hallucinations: Verify accuracy of the generated text via the URL.
- **Temporal information**: The time when the resource was created, modified, etc.
  - Improve quality: Generate text more relevant to the relevant time and trends.
  - Reduce hallucinations: Avoid generating anachronistic text.
- **Usage information:** The number of times the resource is accessed, used, etc.

  - Improve quality: Generate text more likely to be engaging and popular.
  - Reduce hallucinations: Avoid generating text too similar to existing text.
- **Category:** The topics connected to the resource.
  - Improve quality: Generate text connected to the topic.
  - Reduce hallucinations: Avoid generating off-topic and irrelevant text.
- **Provenance:** The history of the resource, including creation, ownership, distribution, etc.
  - Improve quality: Generate accurate and reliable text.
  - Reduce hallucinations: Avoid generating fabricated or misleading text.
- **Usage rights:** Information about permissible uses of the resource.
  - Improve quality: Generate text more likely to be used and shared.
  - Reduce hallucinations: Avoid generating copyrighted or restricted text.
- **Real-time information:** The current state of the resource that can evolve over time.
  - Improve quality: Generate text containing up-to-date information.
  - Reduce hallucinations: Avoid generating text with outdated or inaccurate information.
- **Geolocation:** The coordinates of the location where the resource was created or is located.

  For example, the cities in the world, their monuments, properties, highlights, etc.
  - Improve quality: Generate text more relevant to a given locale.
  - Reduce hallucinations: Avoid generating geographically irrelevant or incorrect text.
- **Static ranking:** A static ranking associated with the resource.
  - Improve quality: Generate text more likely to be authoritative and informative.
  - Reduce hallucinations: Avoid generating unreliable and biased text.
- **Metrics data**: Data about resource performance, such as views, downloads, likes, etc.
  - Improve quality: Generate text that is more likely to be popular and shared.

- ○ Reduce hallucinations: Avoid generating text likely to be uninteresting or unpopular.

- **Social media data:** Data from social media platforms, such as likes, shares, comments, etc.
    - ○ Improve quality: Generate text more relevant to current conversations and trends.
    - ○ Reduce hallucinations: Avoid generating text that is likely to be offensive or dull.
- **Financial data:** Data about resource-related financial transactions, such as price, quantity, payment method, etc.
    - ○ Improve quality: Generate text with accurate information on markets and finances.
    - ○ Reduce hallucinations: Avoid generating text with inaccurate or misleading financial information.
- **Customer data:** Data about resource-related customers, such as name, email address, phone number, etc.
    - ○ Improve quality: Generate text personalized to a customer's needs and interests.
    - ○ Reduce hallucinations: Avoid generating text irrelevant or offensive to the customer
- **Shipping data:** Data about resource-related shipping information, such as tracking number, shipping address, delivery date, etc.
    - ○ Improve quality: Generate text with accurate information on the shipping process and delivery.
    - ○ Reduce hallucinations: Avoid text with inaccurate or misleading shipping information.
- **User-generated content:** Content created by users of an application, service, platform or site, such as, forum posts, comments, reviews, ratings, etc.
    - ○ Improve quality: Generate text more relevant to the interests and needs of a target audience.
    - ○ Reduce hallucinations: Avoid generating text that is too similar to existing user posts.

- **Feedback:** Input obtained from users about their experience with an application, service, platform, or site, such as bug reports, survey responses, etc.
  - Improve quality: Generate text that accurately reflects user sentiment.
  - Reduce hallucinations: Avoid generating text that mischaracterizes or fabricates user experiences.
- **Database information:** This includes information about the database schema, such as the names of the tables, the names of the columns, the relationships between the tables, etc.
  - Improve quality: Generate text that is consistent with data stored in the database.
  - Reduce hallucinations: Avoid generating text mismatched with the underlying data.
- **Security-related data:** Data about security-related events connected to an application, service, platform, or site, such as authentication attempts, failed password entries, intrusion detection, etc.
  - Improve quality: Generate text that provides accurate security-related information.
  - Reduce hallucinations: Avoid generating text that can create vulnerabilities and reduce security.
- **Performance data:** Data about the performance of an application, service, platform, or site, such as page load times, server response latency, etc.
  - Improve quality: Generate text that is consistent with performance metrics.
  - Reduce hallucinations: Avoid generating text with mischaracterized, misleading, or inaccurate performance information.
- **Whole Document data:** Data about the whole document, represented by its salient terms and other metadata, like publisher, authorship, last modified date, or whole document embeddings etc.

- ○ Improve quality: Generate text grounded on the semantic content in the document.
- ○ Reduce hallucinations: Avoids generating text not grounded in the document or its metadata.

- **Image Metadata:** Images processed by a separate image processing system, and either compressed as a single embedding or a sequence of multiple patch embeddings.
  - ○ Improve quality: This provides extra information to fulfill a user request by conditioning on the image.
  - ○ Reduce hallucinations: By conditioning on the provided image metadata changes of hallucination go down.
- **Video Metadata**: Similar to images, the extracted features can be compressed into a single embedding for each scene. This embedding can capture the overall gist of the scene, including the dominant objects, actions, and audio characteristics. Alternatively, the scene can be divided into smaller patches, and each patch encoded into its own embedding. This provides more granular information, highlighting specific details within the scene.
  - Improve quality: When generating video based on a user request, the generation process can be conditioned on the scene embeddings or patch embeddings. This provides the model with additional context about the desired content, leading to more accurate and relevant results.
  - Reduce hallucination: Conditioning on the video metadata can also help reduce hallucinations. By focusing the generation on the encoded information, the model is less likely to introduce extraneous details that do not align with the scene context.
- **Audio Metadata:** Similar to video scenes, each audio segment can be compressed into a single embedding, capturing its overall sonic characteristics. Alternatively, the audio features

can be encoded as a sequence of embeddings over time. This provides fine-grained information about how the sound evolves within the segment.

- Improve quality: When generating audio based on a user request, the generation process can be conditioned on the segment embeddings or time-series embeddings. This provides the model with additional context about the desired soundscape, leading to more realistic and relevant results.
- Reduce hallucination: Conditioning on audio metadata can also help reduce hallucinations, which can manifest as unnatural sound transitions, unrealistic instrument combinations, or nonsensical speech. By focusing the generation on the encoded information, the model is less likely to introduce discordant elements that do not align with the segment context.

The above list is not exhaustive. Additional metadata can be incorporated for specific domains and purposes as appropriate. Further, not all types of metadata may be relevant to a given situation and can be excluded if irrelevant.

Unembedded metadata can require a significant amount of information to be learned. To encode this information in LLMs (transformers) through examples can require significant training data and subsequently, significant compute cost to process the tokens.

Furthermore, metadata that is provided in terms of embeddings can change from one version of the embedding to the next e.g., when it is machine learning based. In such a case, the embedding connector (as shown in Fig, 1) is the only part that needs to be adapted to the newly changed embedding, while the rest of the model can stay fixed.

The techniques described in this disclosure can additionally support multimodal data, such as images, audio, video, etc. In such cases, the metadata used represents the specific mode that is analyzed. For instance, metadata for images can include EXIF data, color histograms, edges, faces, objects, scenes, image quality, etc.; metadata for audio includes ID3 tags, speech transcription, musical structure, etc.; and metadata for video includes frame image metadata, text within video frame, dialog transcripts, people and objects in the video, scene, sentiment, etc. Such metadata can be extracted from the underlying multimodal data via one or more of a variety of suitable tools or techniques, such as computer vision, image processing, speech recognition, music analysis, audio fingerprinting, optical character recognition (OCR), NLP, machine learning, etc.

Furthermore, since many types of metadata can be added (e.g., geolocation, images, videos, etc.) and since there can be multiple metadata even for a single input (e.g., when there are different types of image encoders for the same image that focus on different aspects of the image), metadata dropout can provide robustness against some metadata not being present at model training time versus serving time. By dropping certain metadata at the time of training the model, even if some metadata are not available at serving time, the model can still make use of the remaining metadata and be robust to the absence of the metadata that was available at training time. This allows graceful improvement of quality when more and more metadata is made available for a piece of input, while also providing increased robustness to metadata being missing.

The techniques described in this disclosure can be implemented within any type of transformer model architecture, including Large Language Models (LLMs). Implementation of the techniques can boost trust in model output by incorporating multiple additional signals beyond affinity and attention. By incorporating relevant meta information in the model during training, the techniques can make the information available during generation without needing expensive pre-expansion with prompt augmentation or post-validation after generation. As a result, the techniques can simplify the external scaffolding and reduce the model serving budget while boosting the statistical confidence in the probabilistic word.

CONCLUSION

This disclosure describes techniques to incorporate a variety of metadata types into the native architecture of transformer models. The additional signals can help avoid hallucinations, improve relevance, add robustness, and minimize the need of expensive external scaffolding. Generalizing transformer operation to condition on a diversity of metadata can be achieved in various ways such as adding a metadata embedding layer, conditioning self-attention on the metadata, conditioning with gated self-attention, metadata dropout, employing a different encoder-decoder architecture, etc. Different types of metadata can help in different ways to improve the quality of the output generated by the transformer and reduce hallucinations. The techniques described in this disclosure can also support multimodal data, such as images, audio, video, etc., with the metadata used representing the specific mode.