Topic Modelling with BERT

Instructor: Xandra Dave Cochran
April 3-10, 2024
Centre for Data, Culture & Society

Topic Modelling:

- Unsupervised Machine Learning

Topic Modelling:

- Unsupervised Machine Learning
- Identifies clusters of related words in text

Topic Modelling:

- Unsupervised Machine Learning
- Identifies clusters of related words in text
- Does not require predefined categories – good for discovery and exploration of a dataset

Introductions!

What is your previous experience with machine learning?

Why are you interested in topic modelling?

Have you used LLMs before?

Is there a dataset you have in mind to use for topic modelling in future?

BERTopic

- Problem: we have more data than we know what to do with! (nice problem to have)

BERTopic

- Problem: we have more data than we know what to do with! (nice problem to have)
- Processing large datasets by hand is very time-consuming

BERTopic

- Problem: we have more data than we know what to do with! (nice problem to have)
- Processing large datasets by hand is very time-consuming
- Modern Large Language Models (LLMs) can help – advanced Neural Network models can process large bodies of text to infer meaning, sentiment, topic, etc

BERTopic

- Problem: we have more data than we know what to do with! (nice problem to have)
- Processing large datasets by hand is very time-consuming
- Modern Large Language Models (LLMs) can help – advanced Neural Network models can process large bodies of text to infer meaning, sentiment, topic, etc
- BERTopic uses state-of-the-art Transformer models to infer clusters of related words in a dataset, thereby identifying key topics

A little bit about transformer models

- The neural network architecture behind the current AI book – ChatGPT, Dall-E, etc
- Encoder-decoder
- Positional encoding
- Self-Attention

Transformers

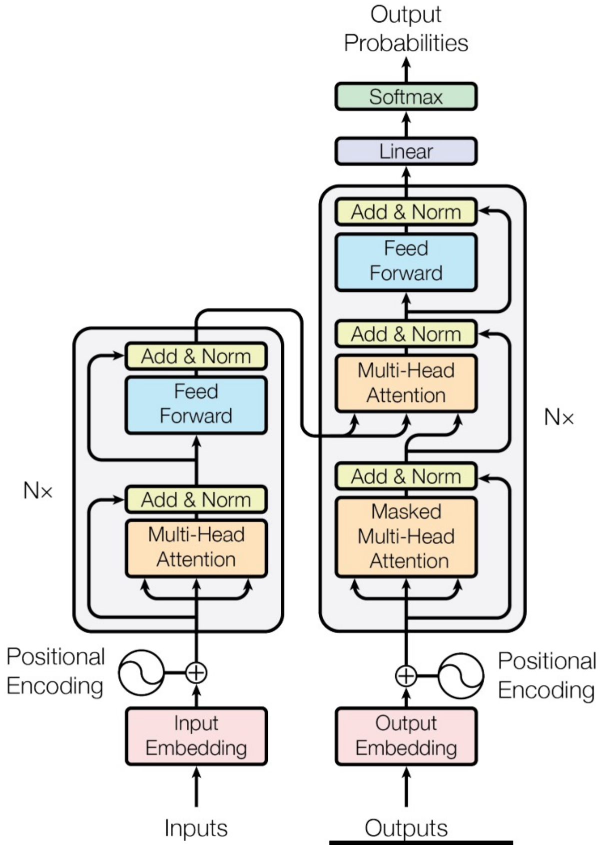
Transformers

Wait, no

Transformers

BERT

Encoder

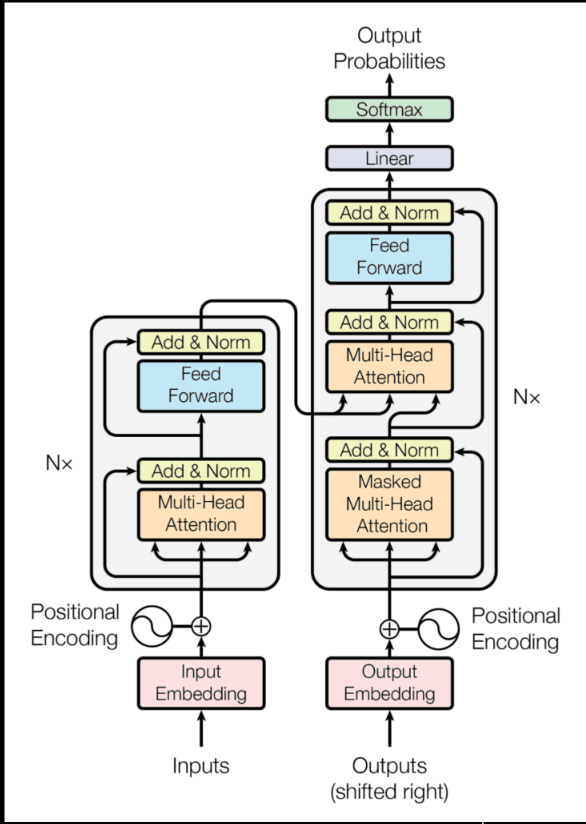


GPT

Decoder

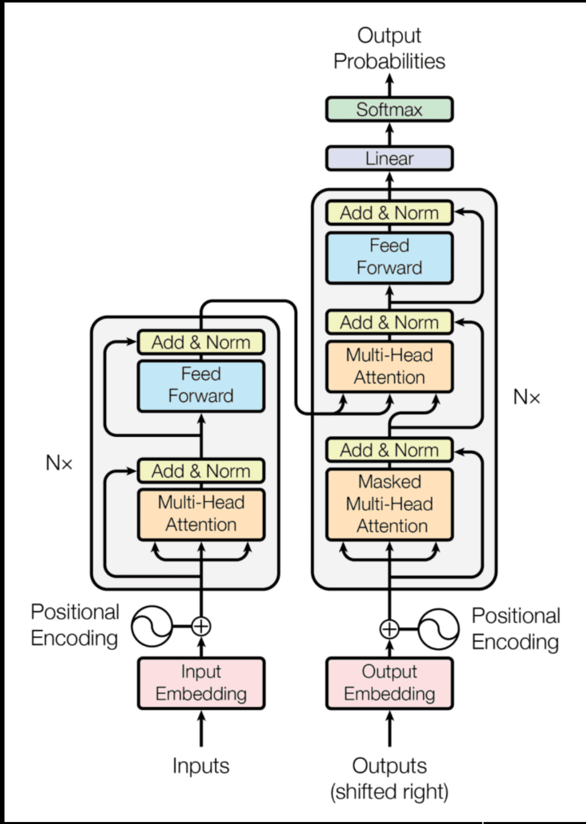
Transformers

- Key insight – computers don't understand text well, they understand large lists of numbers



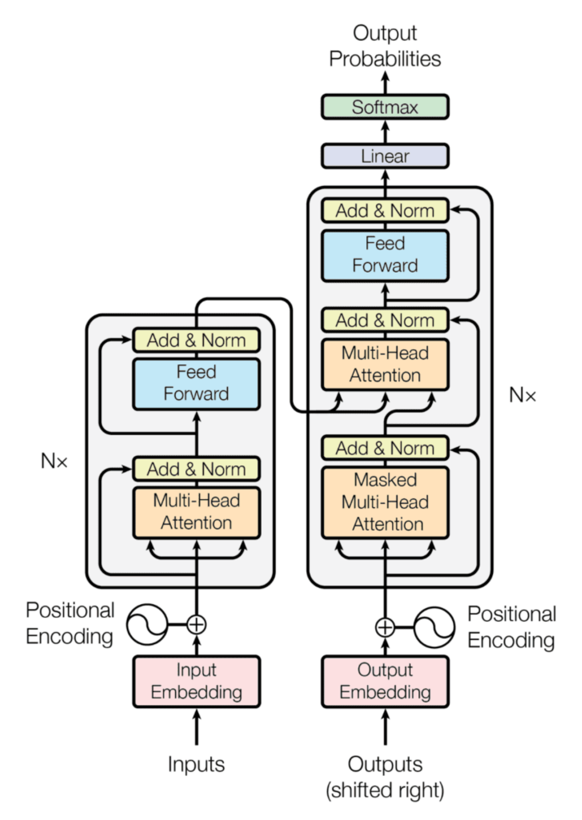
Transformers

- Key insight – computers don't understand text well, they understand large lists of numbers
- Almost everything in machine learning consists of turning a problem that makes sense to humans (but is very time consuming) into a gnarly mess of linear algebra that makes sense to computers, then converting it back to a human-interpretable form



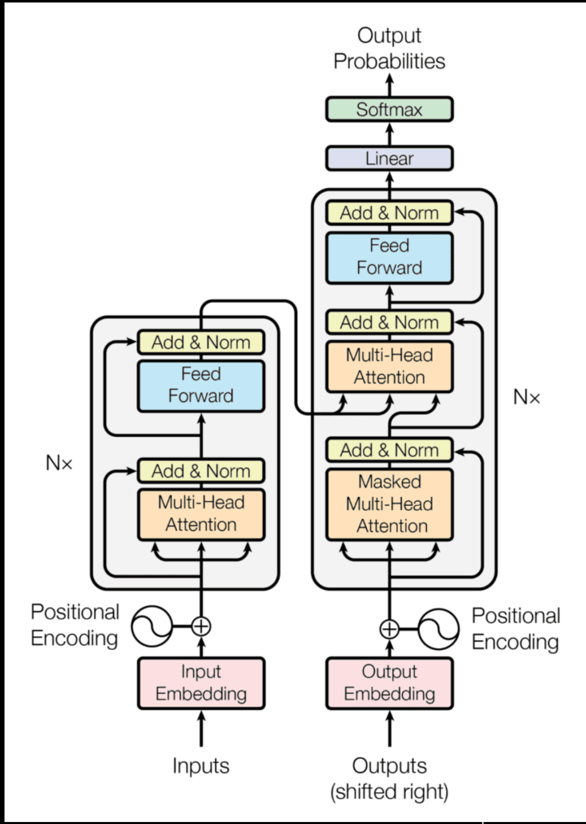
Transformers

- Key insight – computers don't understand text well, they understand large lists of numbers
- Almost everything in machine learning consists of turning a problem that makes sense to humans (but is very time consuming) into a gnarly mess of linear algebra that makes sense to computers, then converting it back to a human-interpretable form
- As such, step 1 is *input embedding*



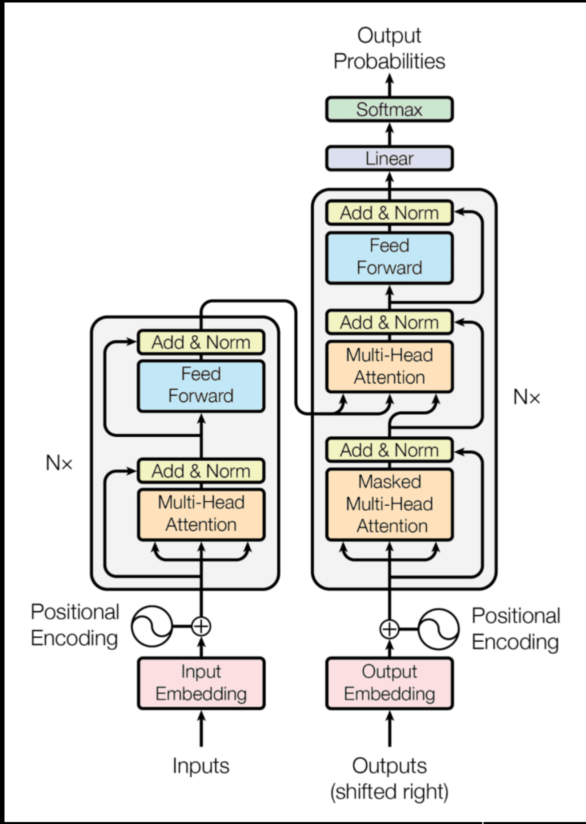
Embeddings

- Each word (more accurately, *token*) is represented as a vector of numbers:
- The = [7.3, -6.1, 8.0 ... -0.2, 3.2]
- cat = [0.1, 3.2, -0.5 ... 3.7, -1.2]



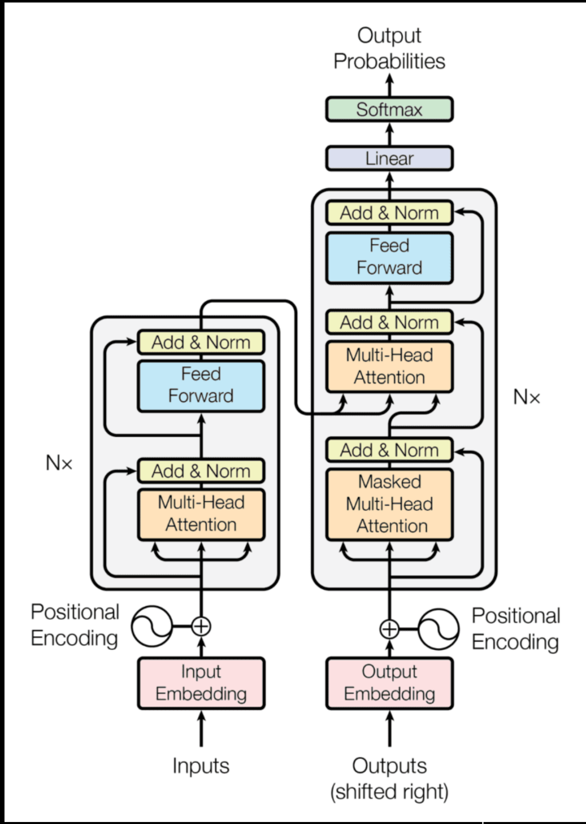
Embeddings

- Each word (more accurately, *token*) is represented as a vector of numbers:
- The = [7.3, -6.1, 8.0 ... -0.2, 3.2]
- cat = [0.1, 3.2, -0.5 ... 3.7, -1.2]
- Seems random, but the idea is to train these embeddings to represent *meaning* as a *high-dimensional space*



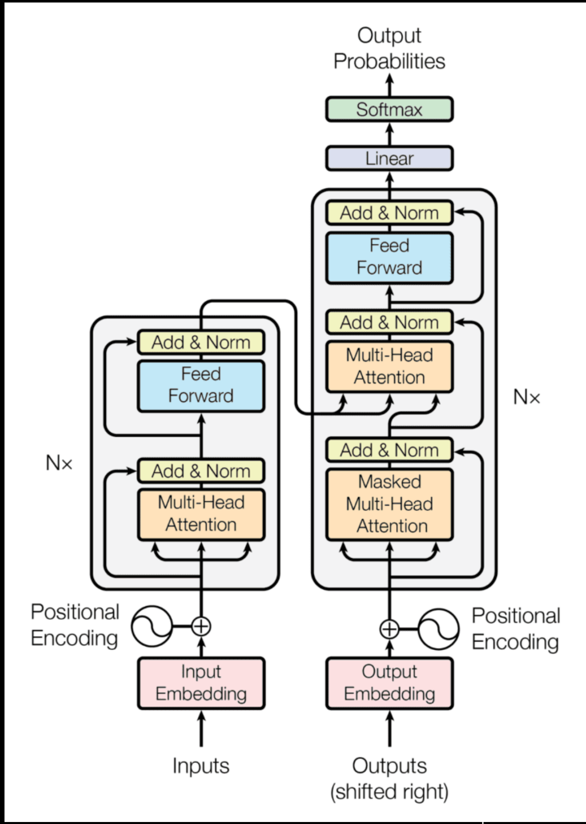
Embeddings

- Each word (more accurately, *token*) is represented as a vector of numbers:
- The = [7.3, -6.1, 8.0 ... -0.2, 3.2]
- cat = [0.1, 3.2, -0.5 ... 3.7, -1.2]
- Seems random, but the idea is to train these embeddings to represent *meaning* as a *high-dimensional space*
- This means semantically related words will be closer together than unrelated words - can be measured with cosine distance



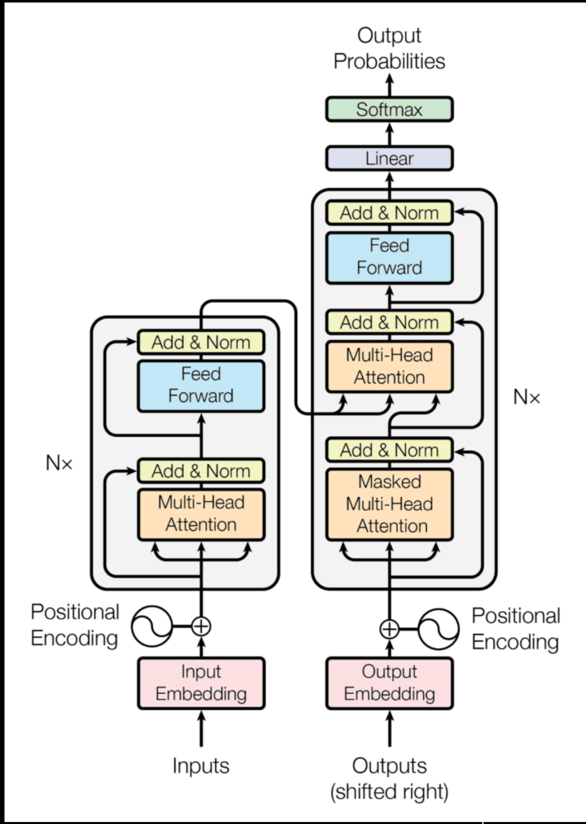
Embeddings

- Each word (more accurately, *token*) is represented as a vector of numbers:
- The = [7.3, -6.1, 8.0 ... -0.2, 3.2]
- cat = [0.1, 3.2, -0.5 ... 3.7, -1.2]
- Seems random, but the idea is to train these embeddings to represent *meaning* as a *high-dimensional space*
- This means semantically related words will be closer together than unrelated words – can be measured with cosine distance
- Displacements in embedding space also have interesting properties, e.g.: (PUPPY – DOG) + CAT ≈ KITTEN



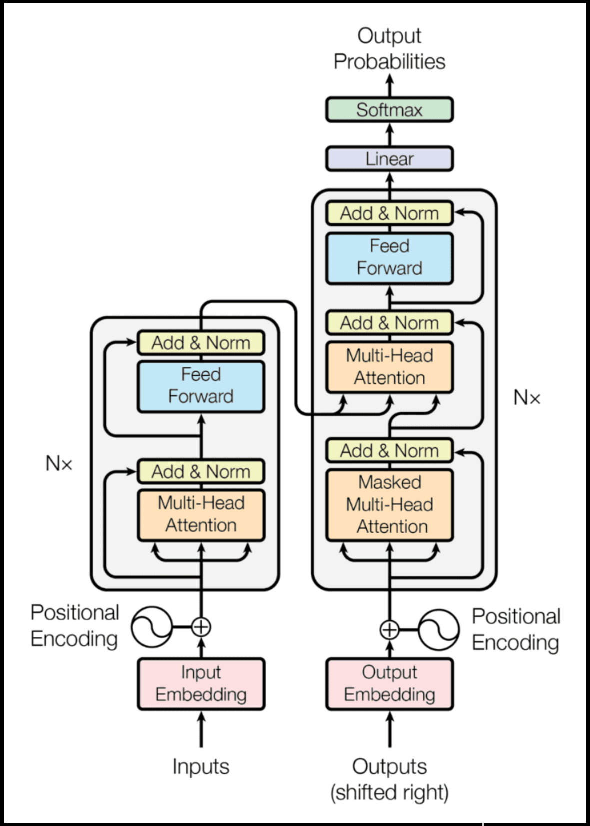
Token Embedding & Positional Encoding

- Each token in a piece of text corresponds to a vector in semantic space



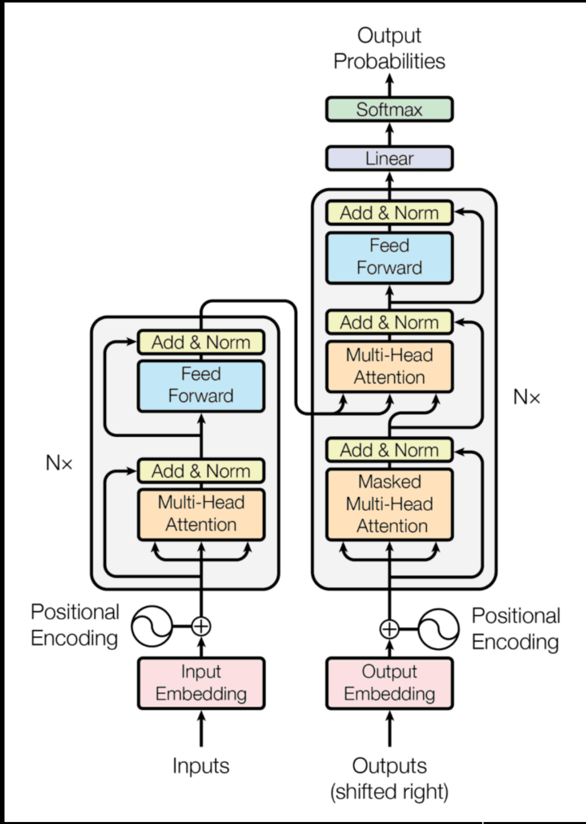
Token Embedding & Positional Encoding

- Each token in a piece of text corresponds to a vector in semantic space
- Combine this with another vector that represents the position of the word in the input – usually the output of a periodic function, e.g., a sum of multiple sine and cosine waves



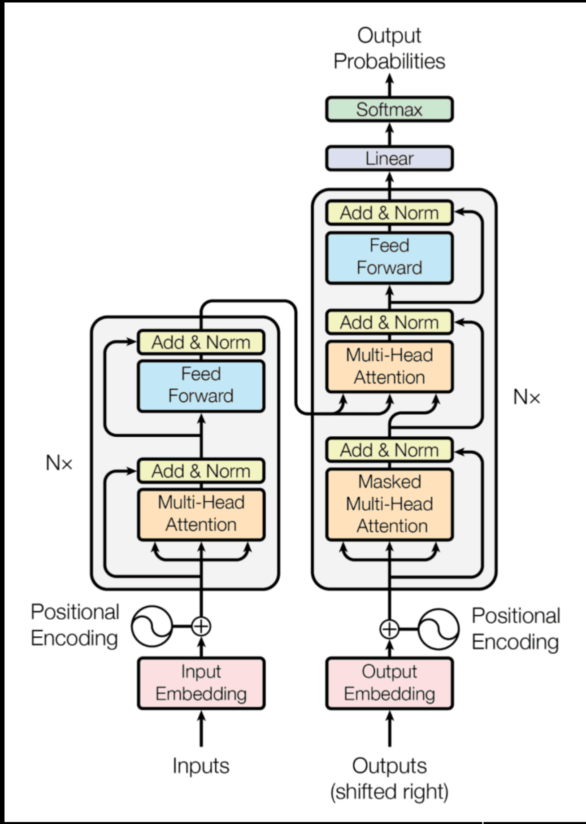
Self-Attention

- Calculate the vector product of each word in the input with all words in the input, sum and normalise the result



Self-Attention

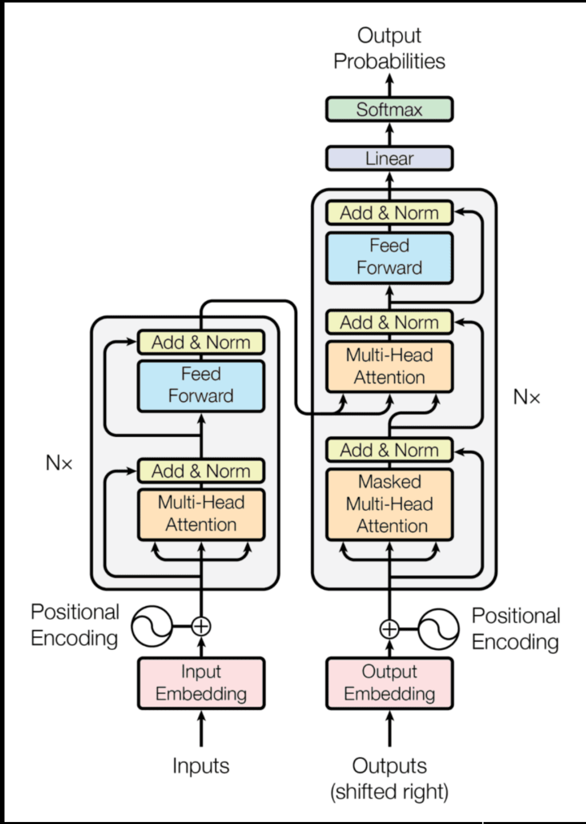
- Calculate the vector product of each word in the input with all words in the input, sum and normalise the result
- *Insight*: words that are related (closer in semantic space) in the input will



Self-Attention

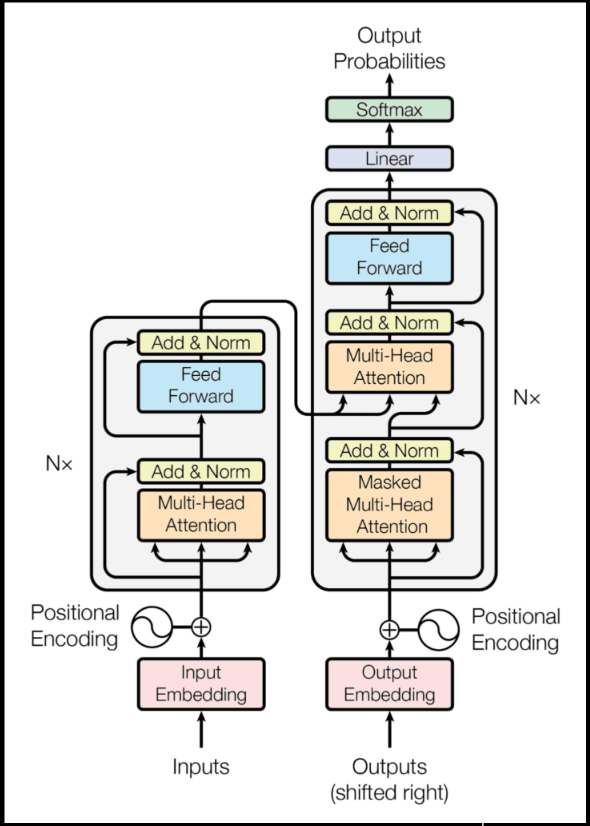
- Calculate the vector product of each word in the input with all words in the input, sum and normalise the result
- *Insight*: words that are related (closer in semantic space) in the input will

- ---be more influential on the result



Self-Attention

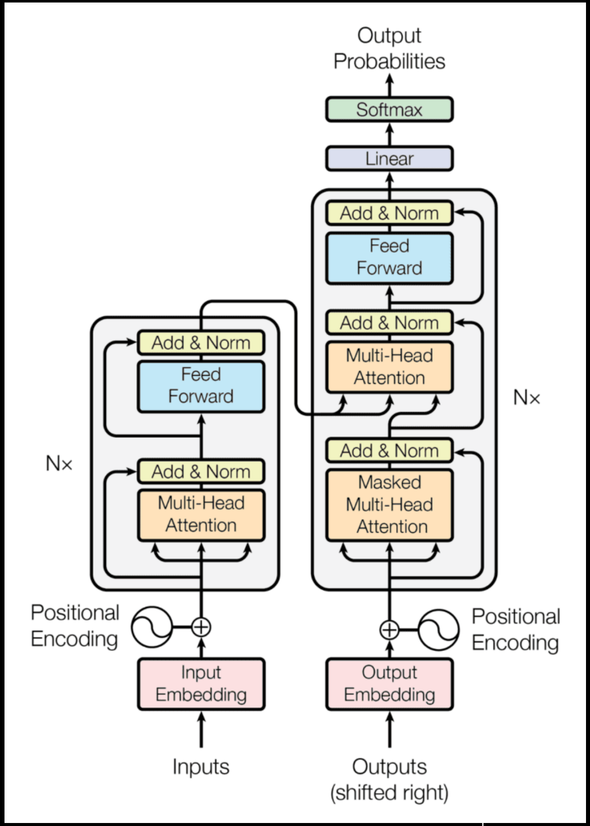
- Calculate the vector product of each word in the input with all words in the input, sum and normalise the result
- *Insight*: words that are related (closer in semantic space) in the input will
- ---be more influential on the result
- ---be more likely to be relevant to understanding the context of the focal word (e.g., the presence of 'river' disambiguates 'bank'



Self-Attention

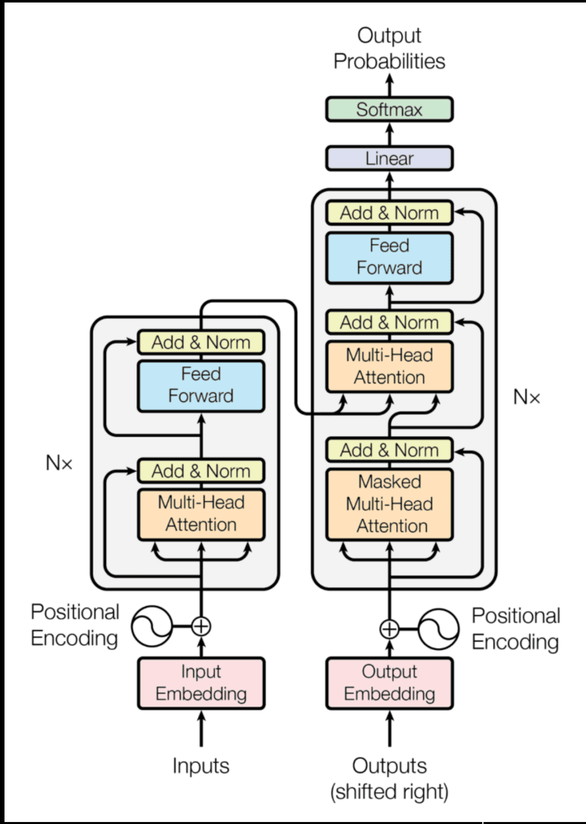
- Calculate the vector product of each word in the input with all words in the input, sum and normalise the result
- *Insight*: words that are related (closer in semantic space) in the input will

- ---be more influential on the result
- ---be more likely to be relevant to understanding the context of the focal word (e.g., the presence of 'river' disambiguates 'bank'
- ---will act to shift the embedding of the original word to its specific meaning *in context*



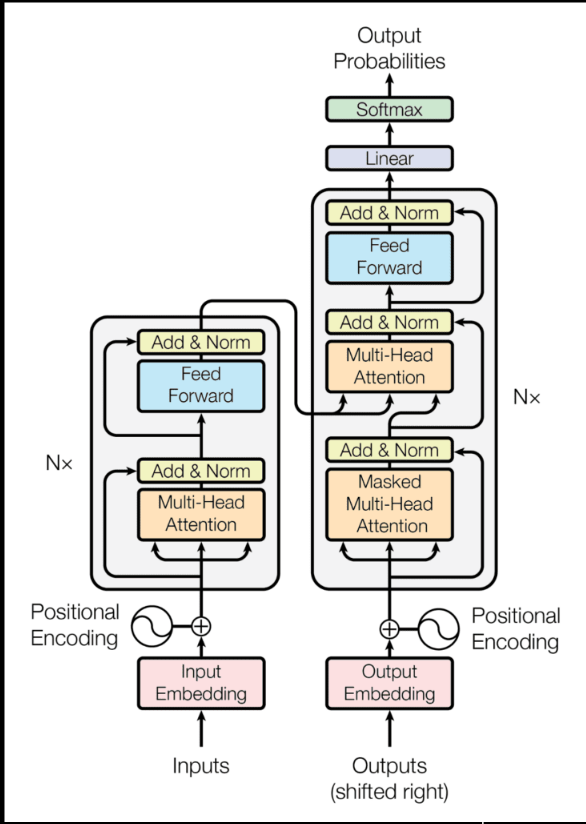
Add & Norm

- The output of attention is added to the original embedding vectors, and normalised so each vector sums to 1.0
- The result is a reweighted version of the embedding which accounts for context



Feed Forward

- Feed the result through a simple feed forward neural network
- Add to original values and normalise again
- repeat



BERTopic

- This can be used to solve next-word problems (chat GPT), missing-word problems, and calculate embeddings for larger blocks of text (BERT)

BERTopic

- This can be used to solve next-word problems (chat GPT), missing-word problems, and calculate embeddings for larger blocks of text (BERT)
- In BERTopic, this is combined with a custom Term Frequency-Inverse Document Frequency model to induce clusters of related words, and this discover topics in a dataset

DEMO

Attributes

- There are a number of attributes that you can access after having trained your BERTopic model:

Attribute	Description
topics_	The topics that are generated for each document after training or updating the topic model.
probabilities_	The probabilities that are generated for each document if HDBSCAN is used.
topic_sizes_	The size of each topic
topic_mapper_	A class for tracking topics and their mappings anytime they are merged/reduced.
topic_representations_	The top *n* terms per topic and their respective c-TF-IDF values.
c_tf_idf_	The topic-term matrix as calculated through c-TF-IDF.
topic_labels_	The default labels for each topic.
custom_labels_	Custom labels for each topic as generated through `.set_topic_labels`.
topic_embeddings_	The embeddings for each topic if `embedding_model` was used.
representative_docs_	The representative documents for each topic if HDBSCAN is used.

We can visualise

- Topics
- Topic Probabilities
- Topic Hierarchies
- Terms
- Topic Similarity

DEMO

Also, we can…

- Search topics
- Reduce topics

DEMO

Thanks Everyone!

Next class: Wednesday 10th, 2-4PM

Please message me on Teams for office hours!