அத்தியாயம் 5: Transformers NLP துறையில் ஒரு புரட்சி

2017-ஆம் ஆண்டு, செயற்கை நுண்ணறிவு உலகம், ஒரு புதிய கண்டுபிடிப்புக்காகக் காத்திருந்தது. அதுவரை, RNN மற்றும் LSTM போன்ற வரிசைக்கிரம மாதிரிகள், NLP துறையை ஆண்டு வந்தன. ஆனால், அவற்றின் உள்ளார்ந்த குறைபாடுகளால், ஒரு தேக்கநிலை உருவாகியிருந்தது. அந்தத் தேக்கநிலையை உடைத்து, AI-யின் அடுத்த கட்ட வளர்ச்சிக்கான வழியைக் காட்ட, ஒரு புதிய சிந்தனை தேவைப்பட்டது.

அந்தப் புதிய சிந்தனை, அந்த ஆண்டின் புகழ்பெற்ற NeurIPS மாநாட்டில், கூகுள் பிரெய்ன் (Google Brain) குழுவின் ஒரு ஆய்வுக் கட்டுரையாக வெளிப்பட்டது. அதன் தலைப்பே ஒரு பிரகடனமாக இருந்தது: "Attention Is All You Need" (உங்களுக்குத் தேவையானது கவனம் மட்டுமே).

அது வெறும் ஆய்வுக் கட்டுரை அல்ல; அது NLP உலகின் பழைய நம்பிக்கைகளைத் தகர்த்தெறிந்த ஒரு புரட்சியின் அறிக்கை. "மொழியைப் புரிந்துகொள்ள, வார்த்தைகளை ஒன்றன் பின் ஒன்றாகப் படிக்க வேண்டிய அவசியமே இல்லை" என்ற ஒரு தைரியமான, புதிய தத்துவத்தை அது உலகுக்கு அறிமுகப்படுத்தியது. இந்த ஒற்றை யோசனை, செயற்கை நுண்ணறிவின் பயணத்தையே திசை திருப்பியது.

அந்தப் புரட்சியின் பெயர்: **டிரான்ஸ்ஃபார்மர் (Transformer)**.

டிரான்ஸ்ஃபார்மர் பிறப்பதற்கு முன்பு, RNN மற்றும் LSTM போன்ற நரம்பியல் வலைப்பின்னல்கள், ஒரு வாக்கியத்தை மனிதனைப் போலவே, வார்த்தைக்கு வார்த்தை, ஒன்றன் பின் ஒன்றாகப் படித்துப் புரிந்துகொள்ள முயன்றன. ஆனால், ஒரு நீண்ட கதையைக் கேட்கும்போது, அதன் தொடக்கத்தை நாம் மறந்துவிடுவது போல, இந்த அணுகுமுறையில் ஒரு உள்ளார்ந்த குறைபாடு இருந்தது: ஒரு நீண்ட வாக்கியத்தின் தொடக்கத்தில் உள்ள முக்கியத் தகவல்கள், வாக்கியத்தின் இறுதிக்கு வரும்போது, கணினியின் நினைவிலிருந்து முற்றிலுமாக மங்கிவிடும்.

டிரான்ஸ்ஃபார்மர், இந்த வரிசை சிறையை உடைத்தது. அது, ஒரு வாக்கியத்தை முழுமையாக, ஒரே நேரத்தில் பார்க்கும் ஒரு புதிய பார்வையை அறிமுகப்படுத்தியது. அதன் ரகசிய ஆயுதம், **தன்னியக்கக் கவனத்தின் நுட்பம் (Self-Attention Mechanism)**. இதன் மூலம், ஒரு வாக்கியத்தில் உள்ள ஒவ்வொரு வார்த்தையும், மற்ற எல்லா வார்த்தைகளுடனும் உள்ள தொடர்பைத் தானே கண்டறிந்து, எதற்கு அதிக கவனம் செலுத்த வேண்டும் என்று தீர்மானிக்கிறது. இது, கணக்கீடுகளைப் பன்மடங்கு வேகப்படுத்தியது **(Parallel Processing)** மட்டுமல்லாமல், மொழியின் மிக நுணுக்கமான உறவுகளையும் புரிந்துகொள்ளும் ஆற்றலை வழங்கியது.

இந்த ஒற்றைப் புரட்சிகரமான யோசனையிலிருந்துதான், இன்று நாம் வியக்கும் **BERT**, **GPT**, **T5** போன்ற மாபெரும் மாதிரிகள் பிறந்தன. அவை, மொழிபெயர்ப்பு, உரை சுருக்கம், வினா-விடை, மற்றும் உரை உருவாக்கம் என மொழியின் அனைத்துப் பரிமாணங்களிலும் ஒரு புதிய தரத்தை நிர்ணயித்தன. டிரான்ஸ்ஃபார்மரின் தாக்கம் NLP-யுடன் நிற்கவில்லை; கணினிப் பார்வை (Computer Vision) போன்ற மற்ற துறைகளிலும் அதன் எதிரொலி கேட்டது.

வாருங்கள், செயற்கை நுண்ணறிவின் எதிர்காலத்தை வடிவமைத்த இந்த மாபெரும் இயந்திரத்தின் உள்ளே சென்று, அதன் கட்டமைப்பு மற்றும் அதன் ஆன்மாவான "கவனம்" என்ற தத்துவம் பற்றி விரிவாக ஆராய்வோம்.

Transformers-ன் கட்டமைப்பு

டிரான்ஸ்ஃபார்மர் என்ற இந்தப் புரட்சிகரமான இயந்திரத்தின் உள்ளே பார்த்தால், நமக்கு ஏற்கெனவே பரிச்சயமான சில பாகங்கள் தெரியும். ஆம், இதுவும் ஒரு **என்கோடர்-டிகோடர்** கட்டமைப்பைக் கொண்டதுதான். ஆனால், இது நாம் முன்பு பார்த்த RNN-ஐ அடிப்படையாகக் கொண்ட கூட்டணி அல்ல. இது, "கவனத்தின் நுட்பம்" (Attention Mechanism) என்ற ஒரு புதிய, சக்திவாய்ந்த தத்துவத்தை மையமாகக் கொண்டு உருவாக்கப்பட்ட ஒரு அடுத்த தலைமுறைப் பதிப்பு. வாருங்கள், இந்த இயந்திரத்தின் ஒவ்வொரு பாகத்தையும் விரிவாகப் பார்ப்போம்.

உள்ளீட்டுப் பிரதிநிதித்துவம் (Input Representation)

இந்த மாபெரும் இயந்திரம், தனது பயணத்தைத் தொடங்கும் முன், அதற்கு எரிபொருள் நிரப்ப வேண்டும். அந்த எரிபொருள், நாம் ஏற்கெனவே சந்தித்த **எம்பெடிங் வெக்டர்கள் (Embedding Vectors)** தான்.

உள்ளே வரும் ஒவ்வொரு வார்த்தையும், அதன் ஆழமான அர்த்தத்தையும், குணத்தையும் தன்னுள் அடக்கிய ஒரு எண் வெக்டராக மாற்றப்படுகிறது. இந்த கணித ஆன்மாக்களைத்தான், டிரான்ஸ்ஃபார்மரின் என்கோடர் தனது முதல் உள்ளீடாக எடுத்துக்கொள்கிறது.

Embeddings

செயற்கை நுண்ணறிவின் உலகில், ஒவ்வொரு வார்த்தைக்கும் ஒரு தனித்துவமான, கணிதரீதியான அடையாளம் கொடுக்கப்படுகிறது. இந்த அடையாளம், **எம்பெடிங் வெக்டர் (Embedding Vector)** என்று அழைக்கப்படுகிறது. இது, அந்த வார்த்தையின் அர்த்தத்தையும், குணத்தையும் தன்னுள் அடக்கிய ஒரு **எண்ணியல் கைரேகை (Numerical Fingerprint)** போன்றது.

ஒரு வார்த்தை **w** என்பது, **d** பரிமாணங்களைக் கொண்ட ஒரு வெக்டராக

$$(\mathbf{e}_w \in \mathbb{R}^d) \tag{1}$$

மாற்றப்படுகிறது.

உதாரணமாக, "cat" என்ற சொல், 512 எண்களைக் கொண்ட ஒரு நீண்ட வரிசையாக (`[0.2, -0.5, 0.7, ..., 1.2]`) மாற்றப்படலாம். இந்த எண்ணியல் கைரேகை, அந்த வார்த்தையின் அர்த்தத்தைக் கணினி புரிந்துகொள்ளும் வகையில் பிரதிநிதித்துவப்படுத்துகிறது.

Positional Encoding(இடக்குறியீடு): வார்த்தைகளுக்கு முகவரி கொடுப்பது

டிரான்ஸ்ஃபார்மர், ஒரு வாக்கியத்தை முழுமையாக, ஒரே நேரத்தில் பார்க்கும் ஒரு மாபெரும் சக்தியைக் கொண்டது என்று பார்த்தோம். ஆனால், இந்த அபாரமான சக்திக்கே ஒரு பலவீனம் இருந்தது. ஒரே நேரத்தில் அனைத்தையும் பார்ப்பதால், எது முதலில் வந்தது, எது இரண்டாவதாக வந்தது என்ற **வரிசை (Sequence)** பற்றிய அறிவை அது இழந்துவிடுகிறது.

மொழியில், வரிசை என்பது மிக முக்கியம்.

- `"குழந்தை பூனையை விரட்டியது"`
- `"பூனை குழந்தையை விரட்டியது"`

இந்த இரண்டு வாக்கியங்களிலும் ஒரே வார்த்தைகள்தான் உள்ளன. ஆனால், வரிசை மாறியதால், அர்த்தம் தலைகீழாக மாறிவிட்டது. இந்தச் சிக்கலை டிரான்ஸ்ஃபார்மர் எப்படிச் சமாளிக்கிறது?

அதற்கான புத்திசாலித்தனமான தீர்வுதான் **இடக்குறியீடு (Positional Encoding)**.

டிரான்ஸ்ஃபார்மர், ஒவ்வொரு வார்த்தையின் இடத்திற்கும் ஒரு தனித்துவமான கணிதக் கைரேகையை (Mathematical Fingerprint) உருவாக்க வேண்டும். இதற்காக, அது **சைன் (Sine)** மற்றும் **கொசைன் (Cosine)** என்ற அலைகளை ஒரு புத்திசாலித்தனமான முறையில் பயன்படுத்துகிறது.

இதை, பல முட்களைக் கொண்ட ஒரு கடிகாரமாக (Clock) கற்பனை செய்யுங்கள்:

இந்தக் கடிகாரத்தில், எம்பெடிங்கின் ஒவ்வொரு பரிமாணத்திற்கும் (dimension) ஒரு முள் உள்ளது, மேலும் ஒவ்வொரு முள்ளும் **வெவ்வேறு வேகத்தில்** சுழல்கிறது.

- **வேகமான முட்கள் (குறைந்த அலைநீளம்):** சில முட்கள் நொடி முள் போல மிக வேகமாகச் சுழலும். இவை, ஒரு வார்த்தைக்கு மிக அருகில் உள்ள மற்ற வார்த்தைகளின் இடத்தைத் துல்லியமாகக் குறிக்க உதவுகின்றன.
- **மெதுவான முட்கள் (நீண்ட அலைநீளம்):** மற்ற முட்கள் மணி முள் போல மிக மெதுவாகச் சுழலும். இவை, ஒரு நீண்ட வாக்கியத்தில் ஒரு வார்த்தையின் ஒட்டுமொத்த இடத்தைக் குறிக்க உதவுகின்றன.

ஒரு குறிப்பிட்ட நேரத்தில் (அதாவது, ஒரு வார்த்தையின் குறிப்பிட்ட இடத்தில் - `pos`), இந்த அத்தனை நூற்றுக்கணக்கான முட்களும் இருக்கும் நிலைகளின் தொகுப்பு, அந்த இடத்திற்கு ஒரு தனித்துவமான, மீண்டும் வராத ஒரு கைரேகையை உருவாக்குகிறது. சைன் மற்றும் கொசைன் செயல்பாடுகள், இந்த வெவ்வேறு வேகத்திலான சுழற்சியை கணித ரீதியாக உருவாக்குகின்றன.

இந்த முறையின் மிகப்பெரிய சிறப்பு என்னவென்றால், முதல் இடத்திற்கும் இரண்டாம் இடத்திற்கும் உள்ள கணிதரீதியான தொடர்பு, பத்தாவது இடத்திற்கும் பதினொன்றாவது இடத்திற்கும் உள்ள தொடர்பைப் போலவே இருக்கும். இதனால், டிரான்ஸ்ஃபார்மர் மாடல், வார்த்தைகளின் **உறவுநிலையை (relative positions)** மிக எளிதாகக் கற்றுக்கொள்கிறது.

இந்தத் தாளத்தைக் கணக்கிடும் சூத்திரங்கள்:

$$\mathbf{P}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \tag{2}$$

$$\mathbf{P}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \tag{3}$$

மாறி	விளக்கம்
pos	வார்த்தையின் இடம் (உ.ம்: 1, 2, 3...)
i	எம்பெடிங் பரிமாணத்தின் குறியீட்டு எண்
d	எம்பெடிங்கின் மொத்த பரிமாணம் (உ.ம்: 512)

சுருக்கமாக, இடக்குறியீடு என்பது, டிரான்ஸ்ஃபார்மரின் வேகமான, இணைச் செயலாக்கத் திறனை (Parallel Processing) வைத்துக்கொண்டு, அதே நேரத்தில் மொழியின் வரிசை அறிவை இழந்துவிடாமல் இருக்க உதவும் ஒரு நேர்த்தியான கணிதத் தந்திரமாகும்.

Embeddings + Positional Encoding

இடக்குறியீடு என்பதை, ஒவ்வொரு வார்த்தையின் அர்த்தத்தோடு (Embedding Vector), அதன் இருப்பிடத்திற்கான ஒரு **GPS தகவலை** இணைப்பதாகக் கற்பனை செய்யுங்கள்.

1. **எம்பெடிங் வெக்டர்:** "பூனை" என்ற வார்த்தையின் அர்த்தம் என்ன என்பதைக் கூறுகிறது.
2. **இடக்குறியீடு வெக்டர்:** அந்த "பூனை" வாக்கியத்தில் **எந்த இடத்தில்** (முதல், இரண்டாம், அல்லது மூன்றாம் இடத்தில்) வருகிறது என்பதைக் கூறுகிறது.

இந்த இரண்டு வெக்டர்களையும் ஒன்றாக இணைப்பதன் மூலம், டிரான்ஸ்ஃபார்மர் ஒரு வார்த்தையின் அர்த்தத்தை மட்டுமல்ல, வாக்கியத்தில் அதன் இடத்தையும் சேர்த்தே புரிந்துகொள்கிறது.

$$\mathbf{P}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \tag{4}$$

இங்கு:

- ($\mathbf{E}$) என்பது embeddings மேட்ரிக்ஸ்.
- ($\mathbf{P}$) என்பது positional encoding மேட்ரிக்ஸ்.
- ($\mathbf{X}$) என்பது இறுதி உள்ளீட்டு பிரதிநிதித்துவம்.

இந்த உள்ளீட்டு பிரதிநிதித்துவம் Transformers மாதிரிக்கு உரையை புரிந்துகொள்ள உதவுகிறது. இது மாதிரிக்கு உரையில் உள்ள வார்த்தைகளின் அர்த்தம் மற்றும் அவற்றின் வரிசை பற்றிய தகவலை ஒரே நேரத்தில் வழங்குகிறது.

குறியாக்கி (Encoder)

டிரான்ஸ்ஃபார்மர் என்ற மாபெரும் இயந்திரத்தின் முதல் அறைக்குள், வார்த்தைகளின் அர்த்தமும் (`Embedding`), அதன் இடமும் (`Positional Encoding`) கலந்த ஒரு சக்திவாய்ந்த உள்ளீடு இப்போது நுழைகிறது. இந்த முதல் அறையின் பெயர்தான் **என்கோடர் (Encoder)**.

என்கோடரின் ஒரே நோக்கம், உள்ளே வரும் வாக்கியத்தின் ஒவ்வொரு வார்த்தையையும் மிக ஆழமாகப் புரிந்துகொள்வதுதான். இந்த ஆழமான புரிதல், ஒரே படியில் நிகழ்ந்துவிடுவதில்லை. என்கோடர், பல அடுக்குகளைக் (Layers) கொண்டது. ஒரு அறிஞர், ஒரு சிக்கலான கவிதையை மீண்டும் மீண்டும் படித்து, ஒவ்வொரு முறையும் ஒரு புதிய, ஆழமான அர்த்தத்தைக் கண்டறிவதைப் போல, நமது உள்ளீடும் ஒவ்வொரு அடுக்கைக் கடக்கும்போதும், மேலும் மேலும்

செறிவூட்டப்பட்ட ஒரு புரிதலைப் பெறுகிறது.

ஒவ்வொரு அடுக்கின் உள்ளேயும், என்கோடர் இரண்டு முக்கியமான கருவிகளைப் பயன்படுத்துகிறது:

1. **Multi-Head Self-Attention (பன்முகக் கவனக் குவிப்பு)**
2. **Feedforward Neural Network (முன்னோக்கிய நரம்பியல் வலைப்பின்னல்)**

இந்த இரு கருவிகள் கொண்ட அடுக்குகளைப் பலமுறை கடந்து வந்த பிறகு, வார்த்தைகள் தங்கள் ஆரம்ப நிலையில் இருந்து, ஒரு முழுமையான, சூழல் அறிந்த, ஞானம் பெற்ற ஒரு புதிய நிலையை அடைகின்றன..

Multi-Head Self-Attention: Transformers-இன் மூளை!

ஒரு வாக்கியத்தில் உள்ள ஒவ்வொரு வார்த்தையும் மற்ற வார்த்தைகளுடன் எப்படி தொடர்பு கொள்கிறது என்பதைப் புரிந்துகொள்வதே Self-Attention-ன் வேலை!

> ***எடுத்துக்காட்டு:***
> *"குழந்தை பூனையை விரட்டியது"*
>
> - *"**விரட்டியது**" என்பது "**குழந்தை**" மற்றும் "**பூனை**" இரண்டையும் சார்ந்துள்ளது.*
> - ***Self-Attention** இந்த **தொடர்புகளை** தானாகவே கண்டுபிடிக்கும்!*

கவனத்தின் நடனம்: Self-Attention எப்படி வேலை செய்கிறது?

என்கோடரின் ஒவ்வொரு அடுக்கின் உள்ளேயும், ஒரு ஆழமான, அறிவுபூர்வமான உரையாடல் நிகழ்கிறது. அதுதான் **Self-Attention**. ஒரு வாக்கியத்தில் உள்ள ஒவ்வொரு வார்த்தையும், மற்ற எல்லா வார்த்தைகளுடனும் உரையாடி, தனது அர்த்தத்தை ஆழப்படுத்திக் கொள்ளும் ஒரு மேஜிக் இது.

இந்த உரையாடல், நான்கு நேர்த்தியான படிகளைக் கொண்ட ஒரு நடனம் போல அரங்கேறுகிறது:

படி 1: மூன்று பாத்திரங்கள் (Query, Key, Value)

இந்த உரையாடல் மேடையில், ஒவ்வொரு வார்த்தையும் தனக்குத்தானே மூன்று வெவ்வேறு பாத்திரங்களை ஏற்றுக்கொள்கிறது.

பாத்திரம்	அதன் வேலை	விளக்கம்
Query (Q)	கேள்வி கேட்பவர்	"இந்த வாக்கியத்தில் எனது பங்கு என்ன? நான் யாருடன் தொடர்புடையவன்?" என்று கேட்கும்.
Key (K)	தன் அடையாளத்தைக் கூறுபவர்	"நான் தான் இந்த வாக்கியத்தின் கர்த்தா (subject)" அல்லது "செயல்" என தனது முக்கியத்துவத்தை வெளிப்படுத்தும்.
Value (V)	உண்மையான தகவலைத் தருபவர்	தனது முழுமையான, செறிவூட்டப்பட்ட அர்த்தத்தை வழங்கும்.

ஒவ்வொரு வார்த்தையின் எம்பெடிங்கும் `(X)`, பயிற்சி செய்யப்பட்ட எடை `(W)` மூலம் பெருக்கப்பட்டு, இந்த மூன்று தனித்துவமான வெக்டர்களும் உருவாக்கப்படுகின்றன.

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V \tag{5}$$

இங்கு

$$(\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V) \tag{6}$$

என்பது trainable weights (பயிற்சி மூலம் கற்றுக்கொள்ளப்படும் எடைகள்).

படி 2: கவனத்தின் மதிப்பெண்கள் (Attention Scores)

இப்போது, ஒரு வார்த்தையின் **கேள்வி (Query)** வெக்டர், மற்ற எல்லா வார்த்தைகளின் **அடையாள (Key)** வெக்டர்களுடனும் உரையாடுகிறது. இந்த உரையாடலின் மூலம், அவற்றுக்கு இடையிலான தொடர்பு எவ்வளவு வலுவானது என்று ஒரு **கவன மதிப்பெண் (Attention Score)** கணக்கிடப்படுகிறது.

- `"விரட்டியது"` (கேள்வி) என்ற வார்த்தை, `"குழந்தை"` (அடையாளம்) என்ற வார்த்தையைப் பார்க்கும்போது, "ஆகா, இந்தச் செயலைச் செய்தது இவர்தான்!" என்று ஒரு வலுவான தொடர்பு உருவாகிறது. எனவே, அதன் மதிப்பெண் **அதிகமாக** இருக்கும்.

$$\text{Attention Score} = \frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} \tag{7}$$

(d_{k}) என்பது Key-ன் பரிமாணம் (Dimension). இந்த பிரிவு $(\sqrt{d_k})$ attention scores-ஐ நிலைப்படுத்த (Stabilize) உதவுகிறது.

படி 3: சாஃப்ட்மேக்ஸ் - கவனத்தைப் பகிர்தல் (Softmax)

அடுத்து, கணக்கிடப்பட்ட மதிப்பெண்கள், **சாஃப்ட்மேக்ஸ் (Softmax)** என்ற செயல்பாடு மூலம், 0 முதல் 1 வரையிலான நிகழ்தகவுகளாக (probabilities) மாற்றப்படுகின்றன. இது, ஒரு வார்த்தை, மற்ற எந்தெந்த வார்த்தைகளின் மீது தனது கவனத்தைப் பதிக்க வேண்டும் என்பதைத் தீர்மானிக்கிறது.

$$\text{Attention Weights} = \text{Softmax}\left(\frac{\mathbf{QK}^T}{\sqrt{d_k}}\right) \tag{8}$$

> *`"விரட்டியது"` என்ற வார்த்தை, தனது மொத்த கவனத்தில் **90%**-ஐ `"குழந்தை"` மீதும், **10%**-ஐ `"பூனை"` மீதும் செலுத்தலாம்.*

படி 4: இறுதி அர்த்தம் - ஒரு கூட்டுக் கலவை (Weighted Sum)

இறுதியாக, ஒவ்வொரு வார்த்தையும், தனக்குக் கிடைத்த கவன எடைகளின் (Attention Weights) அடிப்படையில், மற்ற எல்லா வார்த்தைகளின் **உண்மையான தகவலை (Value)** உள்வாங்கிக் கொள்கிறது.

- `"விரட்டியது"` என்ற வார்த்தையின் புதிய, செறிவூட்டப்பட்ட அர்த்தம், 90% `"குழந்தை"`-யின் தகவலையும், 10% `"பூனை"`-யின் தகவலையும் கலந்த ஒரு கூட்டுக் கலவையாக உருவாகிறது.

$$[\text{Attention Output} = \text{Attention Weights} \cdot \mathbf{V}] \tag{9}$$

Multi-Head Attention: ஒரு சிந்தனைக் குழு

ஒருவர் மட்டுமே ஒரு விஷயத்தைப் பார்ப்பதற்கும், ஒரு நிபுணர் குழு அதே விஷயத்தைப் பார்ப்பதற்கும் வித்தியாசம் உண்டு அல்லவா? அதுபோலத்தான், **Multi-Head Attention**.

> 1. *ஒவ்வொரு Head-உம் தனி (Q, K, V) வெக்டர்களைக் கொண்டுள்ளது.*
> 2. *ஒவ்வொரு Head-உம் வெவ்வேறு attention patterns-ஐ கற்றுக்கொள்கிறது.*
> 3. *அனைத்து Heads-ன் வெளியீடுகளும் இணைக்கப்பட்டு, ஒரு Linear Transformation மூலம் இறுதி வெளியீடு கணக்கிடப்படுகிறது.*

ஒரே ஒரு Self-Attention அடுக்கு, ஒரு கோணத்தில் மட்டுமே உறவுகளைப் பார்க்கும். ஆனால், Multi-Head Attention, ஒரே நேரத்தில் 8 அல்லது 12 வெவ்வேறு "தலைகளை" (Heads) உருவாக்கி, பல கோணங்களில் உறவுகளை ஆராய்கிறது.

- **தலை 1:** "யார், என்ன செய்தார்?" (Actor-Action) என்ற உறவை ஆராயலாம்.

- **தலை 2:** "செயல், யாரை பாதித்தது?" (Action-Target) என்ற உறவை ஆராயலாம்.
- **தலை 3:** இலக்கணத் தொடர்புகளை ஆராயலாம்.

இந்த எல்லாத் தலைகளின் பார்வைகளையும் ஒன்றிணைக்கும்போது, டிரான்ஸ்ஃபார்மருக்கு அந்த வாக்கியத்தைப் பற்றிய ஒரு மிக ஆழமான, பன்முகப் புரிதல் கிடைக்கிறது. இதுவே அதன் அபாரமான ஆற்றலின் ரகசியம்.

2.2. Feedforward Neural Network: அர்த்தத்தைச் செதுக்கும் பட்டறை

Self-Attention-ன் வெளியீட்டை மேலும் செயலாக்க, ஒரு Feedforward Neural Network (FFN) பயன்படுத்தப்படுகிறது. இது ஒரு எளிய நரம்பியல் வலைப்பின்னல் (Neural Network) ஆகும், இது Self-Attention-ன் வெளியீட்டை மேம்படுத்துகிறது.

FFN என்பது ஒரு எளிய ஆனால் திறன்மிக்க நரம்பியல் வலைப்பின்னல் கட்டமைப்பாகும். இது ஒவ்வொரு சொல்லின் பிரதிநிதித்துவத்தையும் தனித்தனியாக செயலாக்குகிறது.

FFN பொதுவாக இரண்டு layers-ஐ கொண்டிருக்கிறது:

1. **முதல் Layer:** Linear transformation மற்றும் activation function (ReLU போன்றது).
    - உள்ளீட்டு பரிமாணத்தை பெரிதாக்குகிறது (பொதுவாக 4 மடங்கு)
    - எடுத்துக்காட்டாக, 512 பரிமாண உள்ளீடு 2048 பரிமாணமாக விரிவாக்கப்படுகிறது
    - ReLU செயல்பாட்டைப் பயன்படுத்தி எதிர்ம மதிப்புகளை நீக்குகிறது
        - உதாரணம்: [0.5, -1.2, 0.8] → [0.5, 0, 0.8]
2. **இரண்டாவது Layer:** Linear transformation.
    - விரிவாக்கப்பட்ட பரிமாணத்தை மீண்டும் அசல் பரிமாணத்திற்கு குறைக்கிறது (2048 → 512)
    - இந்த செயல்முறை மாதிரிக்கு மிகவும் சிக்கலான பிரதிநிதித்துவங்களைக் கற்றுக்கொள்ள உதவுகிறது

உதாரணமாக, "மழை பெய்தது" என்ற வாக்கியத்தில்:

1. "மழை" என்ற சொல்லின் பிரதிநிதித்துவம் FFN-க்கு உள்ளீடாக வழங்கப்படுகிறது
2. "பெய்தது" என்ற சொல்லின் பிரதிநிதித்துவம் தனியாக FFN-க்கு வழங்கப்படுகிறது

FFN-ன் வெளியீடு Encoder layer-ன் இறுதி வெளியீடாகும். Encoder-ன் முக்கிய பணி உரையில் உள்ள வார்த்தைகளுக்கிடையேயான உறவுகளை புரிந்துகொள்வது மற்றும் அவற்றின் அர்த்தத்தை பிரதிநிதித்துவப்படுத்துவது. இது பல Encoder layers-ஐ கொண்டிருக்கலாம், ஒவ்வொரு layer-உம் Self-Attention மற்றும் FFN-ஐ பயன்படுத்தி உரையை மேம்படுத்துகிறது.

> *"Multi-Head Attention" என்ற குழு உரையாடலில், ஒவ்வொரு வார்த்தையும் மற்ற வார்த்தைகளுடன் தனது உறவைப் புரிந்துகொண்டது. இப்போது, ஒவ்வொரு வார்த்தையும் அந்தப் புதிய அறிவுடன், ஒரு திறமையான கொல்லனின் (Blacksmith) பட்டறைக்குத் தனியாகச் செல்கிறது.*
>
> *அந்தப் பட்டறைதான் **Feedforward Neural Network (FFN)**.*
>
> *இங்கே, ஒவ்வொரு வார்த்தையும் ஒரு தனிப்பட்ட உலோகத் துண்டாகக் கையாளப்பட்டு, அதன் அர்த்தத்தைச் செதுக்கி, அதை மேலும் வலிமையாக்க இரண்டு படிகள் நிகழ்கின்றன:*
>
> 1. ***விரித்து அடித்தல் (Expansion):*** *முதலில், அந்தக் கொல்லன், வார்த்தை என்ற உலோகத்தை நன்கு சூடாக்கி, அதை அடித்து, மிக விரிவான, மெல்லிய தகடாக மாற்றுகிறார். அப்போது, அதில் உள்ள தேவையற்ற கசடுகளை (ReLU-வின் செயல்பாடு) நீக்கிவிடுகிறார். இது, அதன் 512-பரிமாண வடிவத்தை, 2048-பரிமாண பரந்த வெளிக்கு விரிவாக்குவதைப் போன்றது.*
> 2. ***மடித்து உருவாக்குதல் (Contraction):*** *பிறகு, அந்தக் கொல்லன், அந்த சுத்திகரிக்கப்பட்ட விரிந்த தகட்டை மீண்டும் மடித்து, அடித்து, அதன் பழைய அளவிற்கே சுருக்குகிறார். ஆனால் இப்போது, அது முன்பை விட மிகவும் வலிமையான, செறிவூட்டப்பட்ட ஒரு புதிய வடிவமாக மாறியுள்ளது.*
>
> *இந்தச் செயல்முறைக்குப் பிறகு, ஒவ்வொரு வார்த்தையும், சூழலை மட்டும் புரிந்த ஒன்றாக இல்லாமல், தனது தனிப்பட்ட அர்த்தத்திலும் ஒரு புதிய ஆழத்தைப் பெறுகிறது. இந்த ஞானம் பெற்ற வார்த்தைகளே, அடுத்த என்கோடர் அடுக்கிற்குள் நுழையத் தயாராகின்றன.*

Decoder

Decoder என்பது Transformers மாதிரியின் இரண்டாவது முக்கிய பகுதியாகும். இது Encoder-ன் embeddings-ஐ பயன்படுத்தி, வெளியீட்டை (Output) உருவாக்குகிறது. Decoder-ன் பணி என்னவென்றால், Encoder-ல் இருந்து பெறப்பட்ட தகவல்களை பயன்படுத்தி, இலக்கு மொழியில் (Target Language) உரையை உருவாக்குவது அல்லது மொழிபெயர்ப்பது. ஒவ்வொரு Decoder layer-உம் மூன்று முக்கிய பகுதிகளை கொண்டிருக்கிறது:

1. **Masked Multi-Head Self-Attention**
2. **Encoder-Decoder Attention**
3. **Feedforward Neural Network**

3.1 Masked Multi-Head Self-Attention:

Decoder-ல், Self-Attention-ஐ பயன்படுத்தும் போது, ஒரு முக்கியமான வித்தியாசம் உள்ளது. Decoder-ல், **future tokens** (எதிர்கால வார்த்தைகள்) கணக்கில் எடுத்துக்கொள்ளப்படுவதில்லை. இதற்கு **Masking** என்ற ஒரு முறை பயன்படுத்தப்படுகிறது.

நிச்சயமாக, அந்தப் பகுதியை நமது கதை நடையில், மேலும் செறிவூட்டி இங்கே வழங்குகிறேன்.

டிகோடரின் உள்ளே நடைபெறும் Self-Attention, என்கோடரில் நடந்த உரையாடலைப் போன்றதுதான். ஆனால், இங்கே ஒரு மிக முக்கியமான, கடுமையான விதி பின்பற்றப்படுகிறது.

அந்த விதி: **எதிர்காலத்தைப் பார்க்கத் தடை!**

டிகோடர், ஒரு வாக்கியத்தை உருவாக்கும்போது, அது ஒரு தேர்வை எழுதுவதைப் போன்றது. ஒவ்வொரு அடுத்த வார்த்தையையும் அது சரியாக யூகிக்க வேண்டும். தேர்வில் அடுத்த கேள்விக்கான விடையை முன்கூட்டியே பார்ப்பது தவறு அல்லவா? அதேபோல, டிகோடர் தான் அடுத்து உருவாக்கப் போகும் வார்த்தைகளை எட்டிப் பார்ப்பதை, **மாஸ்கிங் (Masking)** என்ற ஒரு தொழில்நுட்பம் தடுக்கிறது.

உதாரணமாக, அது "I love cats" என்ற வாக்கியத்தை உருவாக்கும்போது, "love" என்ற வார்த்தையை உருவாக்கும் நேரத்தில், அதற்கு "I" என்ற முந்தைய வார்த்தை மட்டுமே தெரியும். அதன் பார்வையிலிருந்து , "cats" என்ற எதிர்கால வார்த்தை ஒரு முகமூடியை (Mask) கொண்டு மறைக்கப்பட்டிருக்கும்.

இந்த masking செயல்முறை, டிகோடர் ஒவ்வொரு வார்த்தையையும், தனக்கு முன்னால் உள்ள தகவல்களை மட்டுமே வைத்து, தர்க்கரீதியாக யூகிக்கப் பழக்குகிறது. இதுவே, அது உருவாக்கும் மொழி இயல்பாகவும், மனிதனைப் போலவும் இருப்பதன் ரகசியம்.

> *Encoder-ல் உள்ளதைப் போல, Decoder-லும் Multi-Head Attention பயன்படுத்தப்படுகிறது. ஆனால், இங்கு masking செயல்முறை சேர்க்கப்படுகிறது. இது Decoder-க்கு தற்போதைய வார்த்தையை மட்டுமே பயன்படுத்தி, எதிர்கால வார்த்தைகளை ஊகிக்க உதவுகிறது.*

3.2 Encoder-Decoder Attention:

டிகோடர், முகமூடியணிந்த கவனத்தின் (Masked Self-Attention) மூலம், தான் இதுவரை எழுதியதைத் திரும்பிப் பார்த்தது. இப்போது, அது தனது பயணத்தின் மிக முக்கியமான கட்டத்திற்கு வருகிறது: மூல வாக்கியத்தின் ஞானத்தை உள்வாங்குவது.

இது, டிகோடர் என்ற எழுத்தாளர், என்கோடர் என்ற அறிஞரின் குறிப்புகளைப் புரட்டிப் பார்க்கும் தருணம். இந்த உரையாடல்தான் **என்கோடர்-டிகோடர் கவனம் (Encoder-Decoder Attention)**.

உரையாடல் எப்படி நிகழ்கிறது?

இந்த அறிவுப் பரிமாற்றம், நாம் ஏற்கெனவே சந்தித்த கேள்வி (Query), அடையாளம் (Key) மற்றும் தகவல் (Value) என்ற மூன்று பாத்திரங்களின் மூலம் நிகழ்கிறது. ஆனால், இங்கே ஒரு முக்கியமான வித்தியாசம் உள்ளது.

1. **கேள்வி (Query - Q):** டிகோடர், தனது தற்போதைய நிலையில் இருந்து ஒரு கேள்வியை உருவாக்குகிறது. "நான் இப்போது எழுதப்போகும் அடுத்த வார்த்தைக்கு, மூல வாக்கியத்தில் உள்ள எந்த வார்த்தையின் மீது நான் கவனம் செலுத்த வேண்டும்?" என்பதுதான் அந்தக் கேள்வி. இந்தக் கேள்வி, டிகோடரின் **(Q)** வெக்டராகும்.
2. **அடையாளம் (Key - K) & தகவல் (Value - V):** இந்தக் கேள்வியுடன், டிகோடர் என்கோடரின் இறுதி வெளியீட்டை அணுகுகிறது. என்கோடர், மூல வாக்கியத்தின் ஒவ்வொரு வார்த்தையின் அடையாளத்தையும் **(K)**, அதன் முழுமையான, செறிவூட்டப்பட்ட தகவலையும் **(V)** தயாராக வைத்திருக்கிறது.

டிகோடரின் கேள்வி **(Q)**, என்கோடரின் ஒவ்வொரு அடையாளத்துடனும் **(K)** ஒப்பிடப்பட்டு, மிகவும் பொருத்தமான வார்த்தையின் மீது கவனம் செலுத்தப்படுகிறது. பிறகு, அந்த வார்த்தையின் முழுமையான தகவல் **(V)** டிகோடரால் எடுத்துக்கொள்ளப்படுகிறது.

சுருக்கமாக, இந்த நிலையில், டிகோடர் என்கோருடன் ஒரு நேரடித் தொடர்பை ஏற்படுத்துகிறது. இது, மொழிபெயர்ப்பு துல்லியமாகவும், மூல வாக்கியத்தின் அர்த்தத்திற்கு நேர்மையாகவும் இருப்பதை உறுதி செய்கிறது.

3.3 Feedforward Neural Network:

டிகோடர், தான் எழுதியதைத் திரும்பிப் பார்த்து (Masked Self-Attention), மூலத்தின் அறிவை ஆலோசித்து (Encoder-Decoder Attention), இப்போது ஒரு தெளிவான சிந்தனையைப் பெற்றுவிட்டது. அந்தச் சிந்தனையின் இறுதி வடிவத்தை முடிவு செய்ய, அது கடைசி அறைக்குள் நுழைகிறது.

அதுதான், நாம் ஏற்கெனவே என்கோடரில் சந்தித்த, **Feedforward Neural Network (FFN)** என்ற கொல்லனின் பட்டறை.

இங்கே, அந்த செறிவூட்டப்பட்ட தகவல், ஒரு இறுதி முறை செதுக்கப்பட்டு, அடுத்த வார்த்தையை உருவாக்குவதற்கான மிகத் தெளிவான, சக்திவாய்ந்த ஒரு பிரதிநிதித்துவமாக (representation) மாற்றப்படுகிறது.

ஒரு காலத்தில், செயற்கை நுண்ணறிவு, மொழியை ஒரு நீண்ட, குறுகிய சுரங்கப் பாதை வழியாகப் பயணிப்பதைப் போல, வார்த்தைக்கு வார்த்தை, மெதுவாகப் புரிந்துகொள்ள முயன்றது.

ஆனால், **டிரான்ஸ்ஃபார்மர்** அந்தச் சுரங்கப்பாதையை உடைத்து, ஒரு பரந்த, திறந்தவெளியை உருவாக்கியது.

"கவனம்" (Attention) என்ற தனது ஒற்றைச் சக்தியின் மூலம், அது ஒரு வாக்கியத்தில் உள்ள ஒவ்வொரு வார்த்தையும், மற்ற எல்லா வார்த்தைகளுடனும் கொண்டுள்ள உறவின் முழுமையான சித்திரத்தை ஒரே நேரத்தில் பார்க்கும் திறனைப் பெற்றது. இடக்குறியீடு (Positional Encoding) என்ற கணித மந்திரத்தின் மூலம், அது வேகத்தையும், வரிசைக்கிரமத்தையும் ஒருங்கே பெற்றது.

இதனால், டிரான்ஸ்ஃபார்மர் என்பது வெறும் ஒரு மேம்பட்ட மாதிரி (model) அல்ல. அது, இன்றைய மாபெரும் மொழி மாதிரிகள் (LLMs) என்ற பிரம்மாண்டமான கப்பல்கள் பயணிக்க உதவும் ஒரு சக்திவாய்ந்த **இயந்திரம் (Engine)**.

GPT-யின் படைப்பாற்றல் முதல், BERT-இன் ஆழமான புரிதல் வரை, இன்று நாம் காணும் AI-யின் அத்தனை மாயாஜாலங்களுக்கும் பின்னணியில் இந்த டிரான்ஸ்ஃபார்மர் இயந்திரத்தின் இதயத்துடிப்பே ஒலிக்கிறது. அது மொழியைப் பற்றிய நமது புரிதலை மட்டுமல்ல, இயந்திரங்களின் சாத்தியக்கூறுகள் பற்றிய நமது கற்பனையையும் மாற்றியமைத்த ஒரு தொழில்நுட்பப் புரட்சியாகும்.