அத்தியாயம் 4: மொழி மாதிரிகளில் (language models)

நாம் வாழும் இன்றைய டிஜிட்டல் யுகத்தில், இரண்டு உலகங்கள் ஒன்றோடொன்று உரையாட வேண்டிய கட்டாயத்தில் உள்ளன. ஒன்று, நமது உலகம் – கவிதை, அன்பு, கோபம் என எண்ணற்ற உணர்வுகளையும், சிந்தனைகளையும் சுமந்திருக்கும் மனித மொழியின் உலகம். மற்றொன்று, இயந்திரங்களின் உலகம் – தர்க்கம், கணக்கீடுகள், மற்றும் ஆம்/இல்லை (1/0) என்ற இரும மொழியை மட்டுமே அறிந்திருக்கும் உலகம்.

இந்த இரு உலகங்களுக்கும் இடையே ஒரு பாலத்தை அமைப்பது எப்படி? நமது உணர்வுப்பூர்வமான மொழியை, இயந்திரங்களின் தர்க்கரீதியான மொழிக்கு அறிமுகம் செய்து வைப்பது எப்படி?

இந்த மாபெரும் பயணத்தின் முதல் மற்றும் மிக முக்கியமான படி, நமது வார்த்தைகளின் ஆன்மாவை, **எண்களின்** வடிவத்திற்கு மாற்றுவதுதான்.

அதற்கு முதலில் நாம் பயன்படுத்தும் வார்த்தைகளின் தொகுப்பை வரையறுக்க வேண்டும். இதை சொற்களஞ்சியம் (Vocabulary) என்று முன்னரே அறிமுக படுத்தி இருந்தேன். எடுத்துக்காட்டாக, தமிழில் "**நான் தண்ணீர் குடிக்கலாமா**" என்ற வாக்கியத்தை எடுத்துக்கொள்வோம். இந்த வார்த்தைகளை ஒரு சொற்களஞ்சியமாக வரையறுப்போம். இப்போது ஒவ்வொரு வார்த்தைக்கும் ஒரு தனிப்பட்ட எண்ணை ஒதுக்கலாம்.

- தண்ணீர் = 1
- குடிக்கலாமா = 2
- நான் = 3

அந்த வகையில், "நான் தண்ணீர் குடிக்கலாமா" என்ற வாக்கியம், கணினியின் பார்வையில் `[3, 1, 2]` என்ற எண் வரிசையாக மாறுகிறது. இது ஒரு நல்ல முதல் படி.

ஆனால், இதில் ஒரு சிறிய சிக்கல் உள்ளது. "நான்" (எண் 3) என்ற வார்த்தை, "தண்ணீர்" (எண் 1) என்ற வார்த்தையை விடப் பெரியது அல்லது முக்கியமானது என்று கணினி தவறாக நினைக்கக்கூடும். இந்தக் குழப்பத்தைத் தவிர்க்க, ஒரு மிகத் தெளிவான, புத்திசாலித்தனமான முறை பயன்படுத்தப்படுகிறது. அதுதான் **ஒன்-ஹாட் என்கோடிங் (One-Hot Encoding)**.

இதை, நமது அகராதியில் உள்ள ஒவ்வொரு வார்த்தைக்கும் ஒரு பிரத்யேக ஒளிவிளக்கு (Light Switch) கொடுப்பதாகக் கற்பனை செய்யுங்கள். ஒரு நேரத்தில், ஒரே ஒரு வார்த்தைக்குரிய விளக்கு மட்டுமே **ON (1)** நிலையில் இருக்கும்; மற்ற அனைத்தும் **OFF (0)** நிலையில் இருக்கும். இது, ஒவ்வொரு வார்த்தைக்கும் ஒரு தனித்துவமான, கணித ரீதியான கைரேகையை வழங்குகிறது.

இது எப்படி செயல்படுகிறது?

மேலே உள்ள சொற்களஞ்சியத்தை எடுத்துக்கொள்வோம்:

- தண்ணீர் = [1, 0, 0]
- குடிக்கலாமா = [0, 1, 0]
- நான் = [0, 0, 1]

இப்போது, "**நான் தண்ணீர் குடிக்கலாமா**" என்ற வாக்கியத்தை ஒன்-ஹாட் என்கோடிங் மூலம் மாற்றினால், அது பின்வரும் மேட்ரிக்ஸாக (Matrix) மாறும்:

```
[ [0, 0, 1],
  [1, 0, 0],
  [0, 1, 0] ]
```

இங்கே, ஒவ்வொரு வரியும் ஒரு வார்த்தையைக் குறிக்கிறது. இந்த வார்த்தை எண் வடிவங்களைப் பயன்படுத்தி, கணினிகள் பல்வேறு கணித செயல்பாடுகளை செய்ய முடியும். அவற்றில் ஒரு முக்கியமான செயல்பாடு **டாட் ப்ராடக்ட் (Dot Product)** ஆகும். ஒன்-ஹாட் என்கோடிங்கில் டாட் ப்ராடக்ட் எப்படி பயன்படுத்தப்படுகிறது என்பதைப் பார்ப்போம்.

டாட் ப்ராடக்ட் என்பது இரண்டு வெக்டர்களை (Vectors) எடுத்துக்கொண்டு, அவற்றின் ஒத்த உறுப்புகளைப் பெருக்கி, பின்னர் அந்த பெருக்கல்களின் கூட்டுத்தொகையைக் கண்டுபிடிக்கும் ஒரு கணித செயல்பாடு. இதை **இன்னர் ப்ராடக்ட் (Inner Product)** அல்லது **ஸ்கேலார் ப்ராடக்ட் (Scalar Product)** என்றும் அழைப்பார்கள்.

டாட் ப்ராடக்ட் கணக்கிடும் முறை:

இரண்டு வெக்டர்கள் **A** மற்றும் **B** கொடுக்கப்பட்டால், அவற்றின் டாட் ப்ராடக்ட் பின்வருமாறு கணக்கிடப்படும்:

```
A = [a1, a2, a3]
B = [b1, b2, b3]
A.B = (a1 * b1) + (a2 * b2) + (a3 * b3)
```

ஒன்-ஹாட் என்கோடிங்கில், ஒவ்வொரு வார்த்தையும் ஒரு வெக்டராக குறிப்பிடப்படுகிறது. இந்த வெக்டர்களுக்கு இடையேயான தொடர்பைப் புரிந்துகொள்ள, டாட் ப்ராடக்ட் பயன்படுத்தப்படுகிறது.

ஒரு வெக்டரை அதே வெக்டருடன் டாட் ப்ராடக்ட் செய்தால், முடிவு **1** கிடைக்கும். இது அந்த வெக்டர் தனித்துவமானது என்பதைக் காட்டுகிறது.

உதாரணம்:
"பூனை" என்ற வார்த்தையின் ஒன்-ஹாட் வெக்டர் **[1, 0, 0]** என்று வைத்துக்கொள்வோம். இதை அதே வெக்டருடன் டாட் ப்ராடக்ட் செய்தால்:

```
[1, 0, 0] . [1, 0, 0] = (1 * 1) + (0 * 0) + (0 * 0) = 1
```

இதன் மூலம், ஒரு குறிப்பிட்ட வார்த்தை ஒரு வாக்கியத்தில் அல்லது தொகுப்பில் இருக்கிறதா என்பதை கண்டறியலாம். ஒரு வெக்டரை வேறு ஒரு வெக்டருடன் டாட் ப்ராடக்ட் செய்தால், முடிவு **0** கிடைக்கும். இது அந்த இரண்டு வெக்டர்களும் வேறுபட்டவை என்பதைக் காட்டுகிறது.

உதாரணம்:
"பூனை" என்ற வார்த்தையின் ஒன்-ஹாட் வெக்டர் **[1, 0, 0]** என்றும், "நாய்" என்ற வார்த்தையின் ஒன்-ஹாட் வெக்டர் **[0, 1, 0]** என்றும் வைத்துக்கொள்வோம்.

இவ்விரண்டையும் டாட் ப்ராடக்ட் செய்தால்:

```
[1, 0, 0] . [0, 1, 0] = (1 * 0) + (0 * 1) + (0 * 0) = 0
```

இதன் மூலம், இரண்டு வார்த்தைகளும் வேறுபட்டவை என்பதை உறுதி செய்யலாம். ஒன்-ஹாட் என்கோடிங் முறையில், ஒத்த வார்த்தைகளுக்கு இடையேயான தொடர்பை நேரடியாக அளவிட முடியாது. ஏனெனில், ஒவ்வொரு வார்த்தையும் தனித்தனி வெக்டராக குறிப்பிடப்படுகிறது.

உதாரணம்:
"மகிழ்ச்சி", "சந்தோஷம்", "நகைச்சுவை" போன்ற வார்த்தைகள் ஒன்றுக்கொன்று தொடர்புடையவை. ஆனால், ஒன்-ஹாட் என்கோடிங்கில், இவை தனித்தனி வெக்டர்களாக குறிப்பிடப்படுவதால், இவற்றுக்கு இடையேயான தொடர்பை டாட் ப்ராடக்ட் மூலம் கண்டறிய முடியாது. ஒத்த வார்த்தைகளின் தொடர்பை அளவிட, வேர்ட் எம்பெடிங்ஸ் போன்ற மேம்பட்ட முறைகள் தேவைப்படுகின்றன. **வேர்ட் எம்பெடிங்ஸ் (Word Embeddings)**, வார்த்தைகள் தொடர்ச்சியான வெக்டர் வெளியில் (Continuous Vector Space) குறிப்பிடப்படுகின்றன. இதன் மூலம், ஒத்த வார்த்தைகள் ஒன்றுக்கொன்று அருகில் இருக்கும்.

மேட்ரிக்ஸ் என்றால் என்ன?

மேட்ரிக்ஸ் என்பது எண்களை வரிசைகள் (Rows) மற்றும் நெடுவரிசைகளாக (Columns) அடுக்கி வைக்கும் ஒரு அமைப்பு. இதை ஒரு அட்டவணை போல கற்பனை செய்து கொள்ளலாம்.

உதாரணம்:
ஒரு கடையில் இருக்கும் பழங்களின் விலை பட்டியலை ஒரு மேட்ரிக்ஸாக குறிப்பிடலாம்:

பழம்	விலை (கிலோ)
ஆப்பிள்	200
ஆரஞ்சு	100
வாழைப்பழம்	50

இதை மேட்ரிக்ஸ் வடிவில் எழுதினால்:

```
A = [[200, 100, 50]]
```

மேட்ரிக்ஸ் பெருக்கல் எப்படி செயல்படுகிறது?

இரண்டு மேட்ரிக்ஸ்களை பெருக்குவதற்கு, முதல் மேட்ரிக்ஸின் நெடுவரிசைகளின் எண்ணிக்கையும், இரண்டாவது மேட்ரிக்ஸின் வரிசைகளின் எண்ணிக்கையும் சமமாக இருக்க வேண்டும்.

மேட்ரிக்ஸ் பெருக்கல் செய்யும்போது, முதல் மேட்ரிக்ஸின் ஒவ்வொரு வரிசையையும், இரண்டாவது மேட்ரிக்ஸின் ஒவ்வொரு நெடுவரிசையையும் எடுத்துக்கொண்டு, அவற்றின் ஒத்த உறுப்புகளை பெருக்கி, கூட்டுத்தொகையை கண்டுபிடிக்க வேண்டும்.

உதாரணம்: பழங்களின் விலை கணக்கீடு

மேலே குறிப்பிட்ட பழங்களின் விலை பட்டியல் மேட்ரிக்ஸை (**A**) மற்றும் நாம் வாங்க விரும்பும் பழங்களின் அளவை குறிப்பிடும் மற்றொரு மேட்ரிக்ஸை (**B**) பெருக்கினால், நாம் செலுத்த வேண்டிய மொத்த தொகையை கணக்கிடலாம்.

மேட்ரிக்ஸ் A:

```
A = [
  [200, 100, 50]  // ஆப்பிள், ஆரஞ்சு, வாழைப்பழம் விலை
]
```

மேட்ரிக்ஸ் B:

```
B = [
  [2],  // 2 கிலோ ஆப்பிள்
  [3],  // 3 கிலோ ஆரஞ்சு
  [4]   // 4 கிலோ வாழைப்பழம்
]
```

மேட்ரிக்ஸ் பெருக்கல்:

```
A . B = [
  [ (200 * 2) + (100 * 3) + (50 * 4) ]
]
= [
  [ 400 + 300 + 200 ]
]
= [
  [ 900 ]
]
```

எனவே, நாம் செலுத்த வேண்டிய மொத்த தொகை **900 ரூபாய்.**

இயல் மொழி தெளிதல் (Natural Language Processing - NLP) என்பது கணினிகள் மனித மொழியைப் புரிந்துகொள்ளவும், செயல்படவும் உதவும் ஒரு துறையாகும். இந்தத் துறையில், **முதல் வரிசை தொடர் மாதிரி (First Order Sequence Model)** ஒரு முக்கியமான கருவியாகப் பயன்படுத்தப்படுகிறது. இந்த மாதிரி, வார்த்தைகளின் வரிசையை கணிக்க உதவுகிறது.

சொற்களஞ்சியம் (Vocabulary)

முதலில், நாம் பயன்படுத்தும் வார்த்தைகளின் தொகுப்பை வரையறுக்க வேண்டும். இதை **சொற்களஞ்சியம் (Vocabulary)** என்று அழைக்கிறோம்.

உதாரணம்:
நாம் மூன்று கட்டளைகளை மட்டும் கையாள விரும்புகிறோம் என்று வைத்துக்கொள்வோம்:

1. "எனது கோப்புகளைத் திற." ("Open my files.")
2. "எனது இசையை இயக்கு." ("Play my music.")
3. "எனது புகைப்படங்களைக் காட்டு." ("Show my photos.")

இந்த கட்டளைகளில் உள்ள வார்த்தைகளை வைத்து நமது சொற்களஞ்சியத்தை உருவாக்கலாம்:

```
{எனது, கோப்புகள், திற, இசை, இயக்கு, புகைப்படங்கள், காட்டு}
```

Transition Model

இந்த வார்த்தைகளின் வரிசையை கணிக்க, ஒரு **Transition Model** பயன்படுத்தலாம். இந்த மாதிரி, ஒவ்வொரு வார்த்தைக்கும் அடுத்ததாக வரக்கூடிய வார்த்தையின் நிகழ்தகவை (Probability) காட்டும்.

உதாரணம்:
"எனது" என்ற வார்த்தைக்கு பிறகு, "கோப்புகள்" வரும் நிகழ்தகவு **0.4** என்றும், "இசை" வரும் நிகழ்தகவு **0.3** என்றும், "புகைப்படங்கள்" வரும் நிகழ்தகவு **0.3** என்றும் வைத்துக்கொள்வோம்.

இந்த மாற்றம் மாதிரி, ஒரு **மார்கோவ் சங்கிலி (Markov Chain)** என்று அழைக்கப்படுகிறது. ஏனெனில் இவை சங்கிலி போன்று இணைக்கப்பட்டிருக்கும்; அடுத்த வார்த்தையின் நிகழ்தகவு தற்போதைய வார்த்தையை மட்டுமே சார்ந்ததாக இருக்கும். இதை **முதல் வரிசை மார்கோவ் மாதிரி (First Order Markov Model)** என்றும் அழைக்கிறோம்.

மேட்ரிக்ஸ் வடிவில் குறிப்பிடுதல்

இந்த மார்கோவ் சங்கிலியை, ஒரு மேட்ரிக்ஸ் (Matrix) வடிவில் குறிப்பிடலாம். மேட்ரிக்ஸின் ஒவ்வொரு வரிசையும் (Row) மற்றும் நெடுவரிசையும் (Column) சொற்களஞ்சியத்தில் உள்ள ஒரு வார்த்தையை குறிக்கும். மேட்ரிக்ஸில் உள்ள ஒவ்வொரு உறுப்பும் (Element), அந்த வரிசையில் உள்ள வார்த்தைக்கு அடுத்ததாக நெடுவரிசையில் உள்ள வார்த்தை வரும் நிகழ்தகவை குறிக்கும்.

உதாரணம்:

இப்போது, இந்த வார்த்தைகளுக்கான மாற்றம் மாதிரியை (Transition Model) மேட்ரிக்ஸ் வடிவில் குறிப்பிடலாம். இந்த மேட்ரிக்ஸில், ஒவ்வொரு வரிசையும் ஒரு வார்த்தையைக் குறிக்கும், மற்றும் ஒவ்வொரு நெடுவரிசையும் அந்த வார்த்தைக்கு அடுத்ததாக வரக்கூடிய வார்த்தையின் நிகழ்தகவைக் குறிக்கும்.

மேட்ரிக்ஸ் வடிவில் குறிப்பிடுதல்:

	எனது	கோப்புகள்	திற	இசை	இயக்கு	புகைப்படங்கள்	காட்டு
எனது	0.0	0.4	0.0	0.3	0.0	0.3	0.0
கோப்புகள்	0.0	0.0	1.0	0.0	0.0	0.0	0.0
திற	0.0	0.0	0.0	0.0	0.0	0.0	0.0
இசை	0.0	0.0	0.0	0.0	1.0	0.0	0.0
இயக்கு	0.0	0.0	0.0	0.0	0.0	0.0	0.0
புகைப்படங்கள்	0.0	0.0	0.0	0.0	0.0	0.0	1.0
காட்டு	0.0	0.0	0.0	0.0	0.0	0.0	0.0

விளக்கம்:

- "எனது" என்ற வார்த்தைக்கு பிறகு, "கோப்புகள்" வரும் நிகழ்தகவு **0.4**, "இசை" வரும் நிகழ்தகவு **0.3**, மற்றும் "புகைப்படங்கள்" வரும் நிகழ்தகவு **0.3**.
- "கோப்புகள்" என்ற வார்த்தைக்கு பிறகு, "திற" வரும் நிகழ்தகவு **1.0** (ஏனெனில் "கோப்புகளைத் திற" என்ற கட்டளையில் இது நிகழ்கிறது).
- "இசை" என்ற வார்த்தைக்கு பிறகு, "இயக்கு" வரும் நிகழ்தகவு **1.0** (ஏனெனில் "இசையை இயக்கு" என்ற கட்டளையில் இது நிகழ்கிறது).
- "புகைப்படங்கள்" என்ற வார்த்தைக்கு பிறகு, "காட்டு" வரும் நிகழ்தகவு **1.0** (ஏனெனில் "புகைப்படங்களைக் காட்டு" என்ற கட்டளையில் இது நிகழ்கிறது).

இந்த மேட்ரிக்ஸ், மார்கோவ் சங்கிலியின் மாற்றம் மாதிரியை குறிக்கிறது மற்றும் இது ஒரு முதல் வரிசை மார்கோவ் மாதிரி (First Order Markov Model) ஆகும். இந்த மாதிரியைப் பயன்படுத்தி, வார்த்தைகளின் வரிசையை கணிக்க முடியும்.

இரண்டாம் வரிசை மார்கோவ் மாதிரி (Second Order Markov Model)

முந்தைய வார்த்தையை மட்டும் பார்த்து அடுத்த வார்த்தையை கணிப்பது போதுமானதாக இல்லை. இது ஒரு பாடலின் முதல் சுரத்தை (note) மட்டும் கேட்டு, முழு பாடலையும் ஊகிப்பது போன்றது. ஆனால், குறைந்தது இரண்டு சுரங்கள் (notes) இருந்தால், நம்முடைய ஊகம் மிகவும் துல்லியமாக இருக்கும்.

இதைப் புரிந்துகொள்ள, தமிழில் உள்ள ஒரு எளிய மொழி மாதிரியை (toy language model) பார்க்கலாம். இந்த மாதிரியில் இரண்டு வாக்கியங்கள் மட்டுமே உள்ளன, மேலும் அவை 40/60 விகிதத்தில் உள்ளன என்று வைத்துக்கொள்வோம்:

1. "எனது புத்தகத்தை எடு."
2. "எனது பேனாவை எடு."

இந்த வாக்கியங்களுக்கு ஒரு முதல் வரிசை மார்கோவ் சங்கிலி (First Order Markov Chain) உருவாக்கினால், அது ஒவ்வொரு வார்த்தையையும் அதற்கு முந்தைய ஒரு வார்த்தையை மட்டும் பார்த்து கணிக்கும். ஆனால், இந்த மாதிரியில் சில நிச்சயமற்ற தருணங்கள் (uncertainty) இருக்கும். எடுத்துக்காட்டாக, "எனது" என்ற வார்த்தைக்குப் பிறகு "புத்தகத்தை" அல்லது "பேனாவை" வரலாம்.

இதைச் சரிசெய்ய, நமது மாதிரி ஒரு வார்த்தையை மட்டும் பார்க்காமல், இரண்டு வார்த்தைகளைப் பார்த்தால், அது மிகவும் துல்லியமாக செயல்படும். எடுத்துக்காட்டாக, "எனது புத்தகத்தை" என்ற இரண்டு வார்த்தைகளைப் பார்த்தால், அடுத்த வார்த்தை "எடு" என்பது தெளிவாகத் தெரியும். அதேபோல், "எனது பேனாவை" என்ற இரண்டு வார்த்தைகளைப் பார்த்தால், அடுத்த வார்த்தை "எடு" என்பது தெளிவாகத் தெரியும். இதன் மூலம், மாதிரியில் உள்ள கிளைத்தல் (branching) குறைந்து, நிச்சயமற்ற தன்மை (uncertainty) குறைகிறது.

இரண்டு வார்த்தைகளைப் பார்ப்பதன் மூலம், இது ஒரு **இரண்டாம் வரிசை மார்கோவ் மாதிரி (Second Order Markov Model)** ஆக மாறுகிறது. இது அடுத்த வார்த்தையை கணிக்க அதிகமான சூழலை (context) வழங்குகிறது.

முதல் வரிசை vs இரண்டாம் வரிசை மாற்றம் மேட்ரிக்ஸ்

முதல் வரிசை மார்கோவ் மாதிரியில், ஒவ்வொரு வார்த்தைக்கும் அடுத்து வரக்கூடிய வார்த்தைகளின் நிகழ்தகவுகள் மட்டுமே உள்ளன. ஆனால், இரண்டாம் வரிசை மார்கோவ் மாதிரியில், ஒவ்வொரு **இரண்டு வார்த்தைகளின் கலவைக்கும்** (combination) அடுத்து வரக்கூடிய வார்த்தைகளின் நிகழ்தகவுகள் உள்ளன.

முதல் வரிசை மாற்றம் மேட்ரிக்ஸ்:
இந்த மேட்ரிக்ஸில், ஒவ்வொரு வரிசையும் ஒரு வார்த்தையைக் குறிக்கும், மற்றும் ஒவ்வொரு நெடுவரிசையும் அந்த வார்த்தைக்கு அடுத்து வரக்கூடிய வார்த்தையின் நிகழ்தகவைக் குறிக்கும்.

இரண்டாம் வரிசை மாற்றம் மேட்ரிக்ஸ்:
இந்த மேட்ரிக்ஸில், ஒவ்வொரு வரிசையும் **இரண்டு வார்த்தைகளின் கலவையை** (combination) குறிக்கும், மற்றும் ஒவ்வொரு நெடுவரிசையும் அந்த இரண்டு வார்த்தைகளுக்கு அடுத்து வரக்கூடிய வார்த்தையின் நிகழ்தகவைக் குறிக்கும்.

எடுத்துக்காட்டாக, "எனது புத்தகத்தை" என்ற இரண்டு வார்த்தைகளுக்கு அடுத்து "எடு" வரும் நிகழ்தகவு **1.0** ஆகும். அதேபோல், "எனது பேனாவை" என்ற இரண்டு வார்த்தைகளுக்கு அடுத்து "எடு" வரும் நிகழ்தகவு **1.0** ஆகும்.

```
{எனது, புத்தகத்தை, எடு, பேனாவை}
```

இரண்டாம் வரிசை மார்கோவ் மாதிரியில், ஒவ்வொரு இரண்டு வார்த்தைகளின் கலவைக்கும் அடுத்து வரக்கூடிய வார்த்தையின் நிகழ்தகவுகள் கணக்கிடப்படுகின்றன. இதை ஒரு மேட்ரிக்ஸ் வடிவில் குறிப்பிடலாம்.

இரண்டு வார்த்தைகள் (COMBINATION)	அடுத்த வார்த்தை	நிகழ்தகவு (PROBABILITY)
எனது புத்தகத்தை	எடு	1.0
எனது பேனாவை	எடு	1.0
புத்தகத்தை எடு	-	0.0
பேனாவை எடு	-	0.0

விளக்கம்:

1. "**எனது புத்தகத்தை**" என்ற இரண்டு வார்த்தைகளுக்கு அடுத்து "**எடு**" வரும் நிகழ்தகவு **1.0** ஆகும்.
2. "**எனது பேனாவை**" என்ற இரண்டு வார்த்தைகளுக்கு அடுத்து "**எடு**" வரும் நிகழ்தகவு **1.0** ஆகும்.

3. "**புத்தகத்தை எடு**" மற்றும் "**பேனாவை எடு**" போன்ற கலவைகளுக்கு அடுத்து எந்த வார்த்தையும் வராது, எனவே நிகழ்தகவு **0.0** ஆகும்.

இரண்டாம் வரிசை மாற்றம் மேட்ரிக்ஸில், வரிசைகளின் எண்ணிக்கை மிகவும் அதிகமாக இருக்கும். ஏனெனில், ஒவ்வொரு இரண்டு வார்த்தைகளின் கலவைக்கும் ஒரு வரிசை தேவைப்படுகிறது. எனவே, சொற்களஞ்சியத்தில் (vocabulary) (N) வார்த்தைகள் இருந்தால், மேட்ரிக்ஸில் (N^2) வரிசைகள் இருக்கும்.

இரண்டாம் வரிசை மாதிரியின் நன்மைகள்

இரண்டாம் வரிசை மார்கோவ் மாதிரியின் முக்கிய நன்மை என்னவென்றால், அது அதிக நிச்சயத்தன்மையை (confidence) வழங்குகிறது. இந்த மாதிரியில், மேட்ரிக்ஸில் அதிகமான ஒன்றுகள் (1) மற்றும் குறைவான பின்னங்கள் (fractions) இருக்கும். எடுத்துக்காட்டாக, மேலே உள்ள எடுத்துக்காட்டில், ஒரே ஒரு வரிசையில் மட்டுமே பின்னங்கள் உள்ளன (அதாவது, "எனது"க்குப் பிறகு "புத்தகத்தை" அல்லது "பேனாவை" வரும் நிகழ்தகவுகள்).

இரண்டு வார்த்தைகளைப் பார்ப்பதன் மூலம், அடுத்த வார்த்தையை கணிக்க அதிகமான சூழல் (context) கிடைக்கிறது. இது மொழி மாதிரிகளில் (language models) மிகவும் பயனுள்ளதாக இருக்கிறது, ஏனெனில் இது மொழியின் கட்டமைப்பை (structure) மிகவும் துல்லியமாக பிரதிபலிக்கிறது.

இரண்டாம் வரிசை மார்கோவ் மாதிரிகள் சவாலானவையாக இருக்கலாம், ஆனால் அவை மொழி மாதிரிகளின் துல்லியத்தை மேம்படுத்துவதற்கு ஒரு சக்திவாய்ந்த கருவியாகும்.

இரண்டாம் வரிசை மாதிரி மற்றும் ஸ்கிப்ஸ் (Second Order Model with Skips)

இரண்டாம் வரிசை மார்கோவ் மாதிரி (Second Order Markov Model) இரண்டு முந்தைய வார்த்தைகளைப் பார்த்து அடுத்த வார்த்தையை கணிக்கிறது. ஆனால், சில சந்தர்ப்பங்களில், அடுத்த வார்த்தையை கணிக்க நாம் இரண்டுக்கும் மேற்பட்ட முந்தைய வார்த்தைகளைப் பார்க்க வேண்டியிருக்கும்.

எடுத்துக்காட்டாக, இரண்டு வாக்கியங்களை எடுத்துக்கொள்வோம்:

1. "நிமலன் பள்ளிக்குச் சென்றான், பாடங்களைக் கற்றான், வீட்டிற்குத் திரும்பினான்."
2. "நிமலன் பூங்காவிற்குச் சென்றான், பூக்களைப் பார்த்தான், வீட்டிற்குத் திரும்பினான்."

இந்த வாக்கியங்களில், "வீட்டிற்குத்" என்ற வார்த்தைக்குப் பிறகு " திரும்பினான்" வரும் என்பதை தீர்மானிக்க, நாம் முந்தைய பல வார்த்தைகளைப் பார்க்க வேண்டும்.

இதுபோன்ற நீண்ட தூர சார்புகளை (long-range dependencies) கையாள, நாம் மூன்றாம் அல்லது அதற்கு மேற்பட்ட வரிசை மாதிரிகளை (higher-order models) பயன்படுத்தலாம். ஆனால், சொற்களஞ்சியம் (vocabulary) பெரியதாக இருந்தால், இது கணக்கிட மிகவும் சிக்கலானதாக இருக்கும். எடுத்துக்காட்டாக, எட்டாம் வரிசை மாதிரியில் (N^8) வரிசைகள் இருக்கும், இது மிகவும் அதிகமான எண்ணிக்கையாகும்.

இதற்கு பதிலாக, நாம் ஒரு சாமார்த்தியமான முறையைப் பயன்படுத்தலாம்: **இரண்டாம் வரிசை மாதிரியைப் பயன்படுத்தி, முந்தைய வார்த்தைகளின் கலவைகளைக் கருத்தில் கொள்ளலாம்.** இந்த மாதிரியில், நாம் இரண்டு வார்த்தைகளை மட்டும் பார்க்கிறோம், ஆனால் அவற்றில் ஒன்று மிக சமீபத்திய வார்த்தையாகவும், மற்றொன்று முந்தைய எந்தவொரு வார்த்தையாகவும் இருக்கலாம். இது நீண்ட தூர சார்புகளைப் பிடிக்க உதவுகிறது.

இதை ஒரு மேட்ரிக்ஸ் வடிவில் குறிப்பிடலாம். இந்த எடுத்துக்காட்டில், "வீட்டிற்குத்" என்ற வார்த்தைக்குப் பிறகு "திரும்பினான்" வரும் என்பதை தீர்மானிக்க, நாம் முந்தைய இரண்டு வார்த்தைகளின் கலவையைப் பார்க்கிறோம்.

இரண்டு வார்த்தைகள் (COMBINATION)	அடுத்த வார்த்தை	நிகழ்தகவு (PROBABILITY)
பள்ளிக்குச் சென்றான், வீட்டிற்குத்	திரும்பினான்	0.0
பூங்காவிற்குச் சென்றான், வீட்டிற்குத்	திரும்பினான்	0.0
பாடங்களைக் கற்றான், வீட்டிற்குத்	திரும்பினான்	1.0
பூக்களைப் பார்த்தான், வீட்டிற்குத்	திரும்பினான்	1.0

விளக்கம்:

1. "**பாடங்களைக் கற்றான், வீட்டிற்குத்**" என்ற இரண்டு வார்த்தைகளுக்கு அடுத்து "**திரும்பினான்**" வரும் நிகழ்தகவு **1.0** ஆகும்.
2. "**பூக்களைப் பார்த்தான், வீட்டிற்குத்**" என்ற இரண்டு வார்த்தைகளுக்கு அடுத்து "**திரும்பினான்**" வரும் நிகழ்தகவு **1.0** ஆகும்.
3. "**பள்ளிக்குச் சென்றான், வீட்டிற்குத்**" மற்றும் "**பூங்காவிற்குச் சென்றான், வீட்டிற்குத்**" போன்ற கலவைகளுக்கு அடுத்து எந்த வார்த்தையும் வராது, எனவே நிகழ்தகவு **0.0** ஆகும்.

இந்த மேட்ரிக்ஸ், இரண்டாம் வரிசை மார்கோவ் மாதிரியின் அமைப்பைக் காட்டுகிறது. இது முந்தைய இரண்டு வார்த்தைகளைப் பார்த்து, அடுத்த வார்த்தையை கணிக்க உதவுகிறது.

மாஸ்கிங் (Masking)

முந்தைய மாதிரியில், "வீட்டிற்குத்" என்ற வார்த்தைக்குப் பிறகு "திரும்பினான்" வரும் என்பதை கணிக்க, நாம் பல வார்த்தைகளின் கலவைகளைப் பார்த்தோம். ஆனால், இந்த மாதிரியில் பெரும்பாலான கலவைகள் பயனற்றவையாக இருந்தன. எடுத்துக்காட்டாக, "பள்ளிக்குச் சென்றான், வீட்டிற்குத்" மற்றும் "பூங்காவிற்குச் சென்றான், வீட்டிற்குத்" போன்ற கலவைகள் எந்த தகவலையும் வழங்கவில்லை. இதை மேம்படுத்த, நாம் **மாஸ்கிங் (Masking)** என்ற முறையைப் பயன்படுத்தலாம். இந்த முறையில், பயனற்ற கலவைகளை மறைத்து, முக்கியமான கலவைகளை மட்டும் கணக்கில் எடுத்துக்கொள்கிறோம்.

மாஸ்கிங் முறை:

மாஸ்கிங் முறையில், நாம் ஒரு **மாஸ்க் வெக்டர் (Mask Vector)** உருவாக்குகிறோம். இந்த வெக்டரில், முக்கியமான கலவைகளுக்கு **1** மதிப்பும், பயனற்ற கலவைகளுக்கு **0** மதிப்பும் இருக்கும்.

மாஸ்க் வெக்டர்:

```
[பள்ளிக்குச் சென்றான், வீட்டிற்குத்: 0,
 பூங்காவிற்குச் சென்றான், வீட்டிற்குத்: 0,
 பாடங்களைக் கற்றான், வீட்டிற்குத்: 1,
 பூக்களைப் பார்த்தான், வீட்டிற்குத்: 1]
```

இந்த மாஸ்க் வெக்டரை, மாற்றம் மேட்ரிக்ஸுடன் (Transition Matrix) பெருக்கினால், பயனற்ற கலவைகள் மறைக்கப்படும்.

மாஸ்க்டு மாற்றம் மேட்ரிக்ஸ் (Masked Transition Matrix)

இரண்டு வார்த்தைகள் (COMBINATION)	அடுத்த வார்த்தை	நிகழ்தகவு (PROBABILITY)
பள்ளிக்குச் சென்றான், வீட்டிற்குத்	திரும்பினான்	0.0
பூங்காவிற்குச் சென்றான், வீட்டிற்குத்	திரும்பினான்	0.0
பாடங்களைக் கற்றான், வீட்டிற்குத்	திரும்பினான்	1.0
பூக்களைப் பார்த்தான், வீட்டிற்குத்	திரும்பினான்	1.0

விளக்கம்:

1. **"பாடங்களைக் கற்றான், வீட்டிற்குத்"** என்ற இரண்டு வார்த்தைகளுக்கு அடுத்து **"திரும்பினான்"** வரும் நிகழ்தகவு **1.0** ஆகும்.

2. "**பூக்களைப் பார்த்தான், வீட்டிற்குத்**" என்ற இரண்டு வார்த்தைகளுக்கு அடுத்து "**திரும்பினான்**" வரும் நிகழ்தகவு 1.0 ஆகும்.
3. "**பள்ளிக்குச் சென்றான், வீட்டிற்குத்**" மற்றும் "**பூங்காவிற்குச் சென்றான், வீட்டிற்குத்**" போன்ற கலவைகள் மாஸ்க் செய்யப்பட்டு, அவற்றின் நிகழ்தகவு 0.0 ஆகும்.

மாஸ்கிங் முறை, மொழி மாதிரிகளில் (language models) மிகவும் பயனுள்ளதாக இருக்கிறது. இது முக்கியமான தகவல்களை மட்டும் கணக்கில் எடுத்துக்கொண்டு, பயனற்ற தகவல்களை வடிகட்டுகிறது. இது மாதிரியின் செயல்திறனை மேம்படுத்துகிறது மற்றும் துல்லியமான கணிப்புகளை வழங்குகிறது. மாஸ்கிங் முறையின் முக்கிய நன்மை என்னவென்றால், அது மாதிரியின் நம்பகத்தன்மையை (confidence) அதிகரிக்கிறது. பயனற்ற கலவைகளை மறைப்பதன் மூலம், மாதிரி மிகவும் துல்லியமான கணிப்புகளைச் செய்கிறது.