

# **Layman to Layman**

# **Basics of NLP/LLM**

VarCamp 2025

Sept. 7, 2025



Thura Aung

[thuraaung.ai.mdy@gmail.com](mailto:thuraaung.ai.mdy@gmail.com)

Software Engineering (International Program)

King Mongkut's Institute of Technology Ladkrabang

# Intelligence

- Intelligence is a continuum not a dichotomy.
- ဉာဏ်ရည် အဆင့်အတန်း ဟူသည် ပျီးနှံသည့်သဘော၊ ပကတိ အစွမ်းရောက် မဟုတ်
- လူသား သည် ဘာသာစကားကြောင့် တိုးတက်
- လူဘာသာစကားဟာ လူမဟုတ်တဲ့ တြော်စံရွှေစွာတွေရဲ့ ဘာသာစကားနဲ့ မတူပါ
- compositional/productive (“အဓိပ္ပာယ်ရှိတဲ့ အသုံးအနှုန်းတွေက တြော်အဓိပ္ပာယ်ရှိတဲ့ အသုံးအနှုန်းတွေကနေ တည်ဆောက်ထားတာ”) ဖြစ်ပါတယ်
- open-ended → အချိန်ကြာလာတာနဲ့အမျှ စကားလုံးအသစ်တွေနဲ့ စာကြောင်းအသစ်တွေ ထပ်မံဖန်တီးလာနိုင်ပါတယ်
- abstract → စိတ္တဇာ (ဒြပ်ကင်းသဘော) နာမ်တွေကို ဖော်ပြနိုင်

# Learning

- အတွေ့အကြုံများမှ ဉာဏ်ရည်ပေါ်ထွက်လာသည်

ကျို့စာတော်တို့ ဘယ်လို့ "သိ" ကြတာလဲ

- အရာဝတ္ထုတွေနဲ့ အပြန်အလှန်ဆက်ဆံခြင်းအားဖြင့်: မြင်ခြင်း၊ ကြားခြင်း၊ ထံတွေ့ခြင်း၊ အရသာခံခြင်း စသည်ဖြင့်

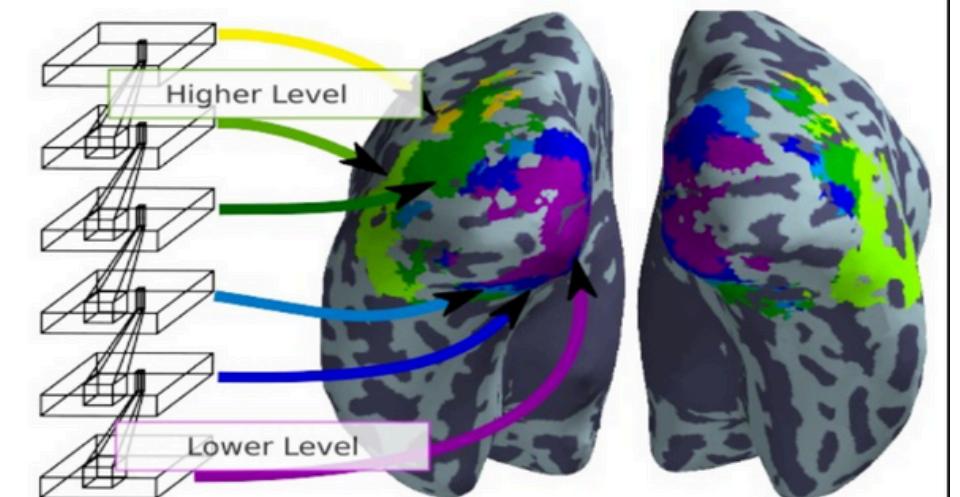
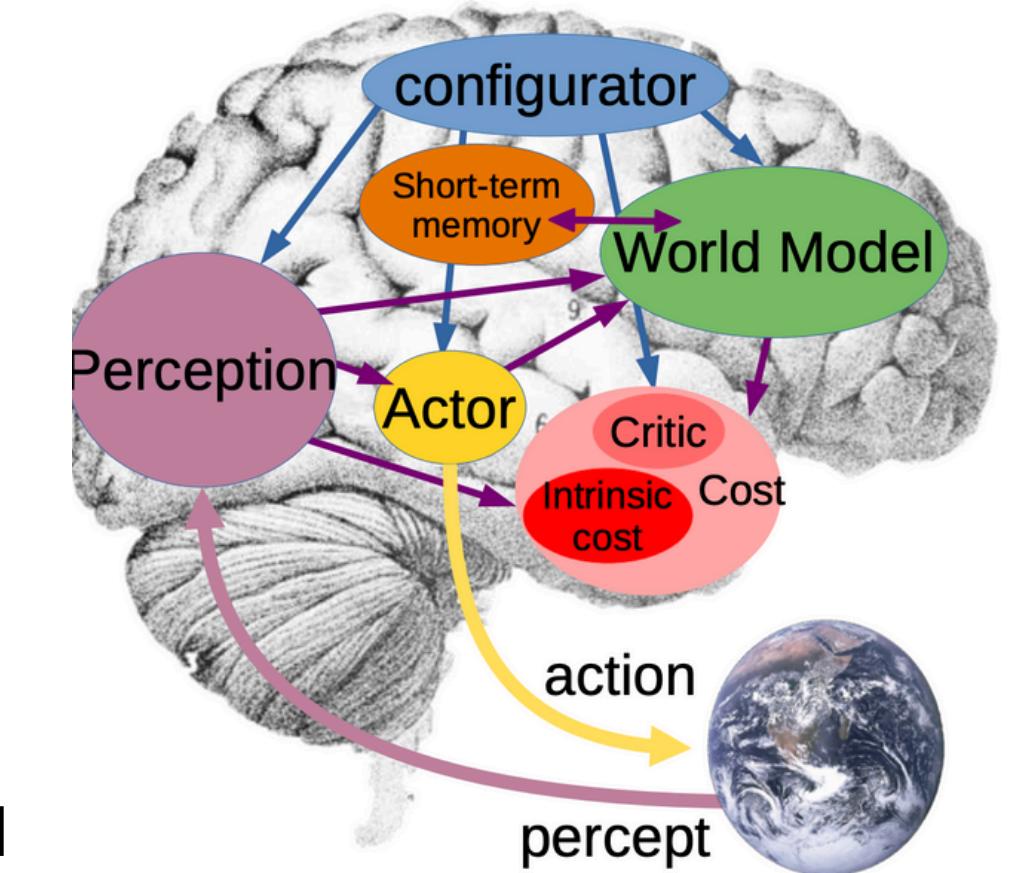
ကျို့စာတော်တို့ဟာ ကျွမ်းကျင်မူတွေကို ဘယ်လို့ "သင်ယူ" ကြတာလဲ။

- လက်တွေ့လုပ်ဆောင်ကြည့်ခြင်းအားဖြင့်: လမ်းလျှောက်ခြင်း၊ ပြေးခြင်း၊ နောက်ပြန်ပစ်ခြင်း (back-flip)၊ ချက်ပြုတ်ခြင်း စသည်ဖြင့်

ဒဏ္ဍာရီထဲက ပျော်စေထိုးကို မြင်ဖူးလား?

- အရေးအသား၊ အပြောအဆိုက တဆင့် သိရတာ (အမှန် အမှား အကုန်ပါ)

ဘာသာစကားတွေလည်း မတူ

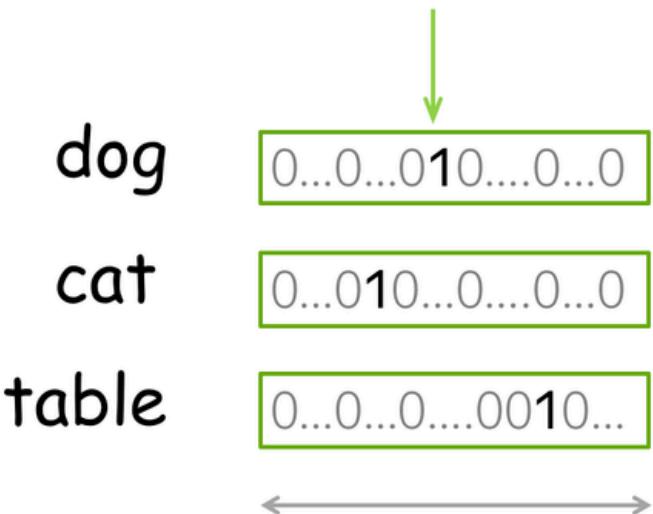


[Eickenberg 2016]

# Bag of Word

- ဝါကျရဲ့အဓိပ္ပာယ်ကို စကားလုံးတွေ တစ်လုံးချင်းစီ ဘယ်နှစ်ခါပါလဲဆိုတာကို ကြည့်ပြီး ဆုံးဖြတ်ခြင်း
- စကားလုံးများကို discrete symbols များအဖြစ်ဖော်ပြခြင်း
- ဆန့်ကျင်ဘက် စကားလုံးတွေ စကားလုံး စီစဉ်ပုံတွေ ဆင်တူတဲ့ စကားလုံးတွေကို ထည့်သွင်းစဉ်းစားမထားပါ

One is 1, the rest are 0



Embedding dimension =  
vocabulary size

- စကားလုံးများကို one-hot vectors များဖြင့် ဖော်ပြနိုင်ပါတယ်။
- Vector dimension = ဝေါဟာရစကားလုံးအရေအတွက်

# Distributional hypothesis

- စကားလုံးတွေဟာ ဆင်တူတဲ့ context တွေမှာ တွေ့ရတတ်ပြီး
- အခါတွေဟာ ဆင်တူတဲ့ အဓိပ္ပာယ်တွေ ရှိတတ်ကြပါတယ်
- You shall know a word by the company it keeps*
- အဓိပ္ပာယ်ဟာ စကားလုံးကိုယ်တိုင်မှာ မရှိဘဲ၊ စကားလုံးပေါ်လာတဲ့ context မှာပဲ ရှိပါတယ်။
- context ကို ကျယ်ကျယ်ပြန့်ပြန့် အနက်ဖွံ့ဖြိုးသင့်တယ်။
- Textual context
- Multimodal context



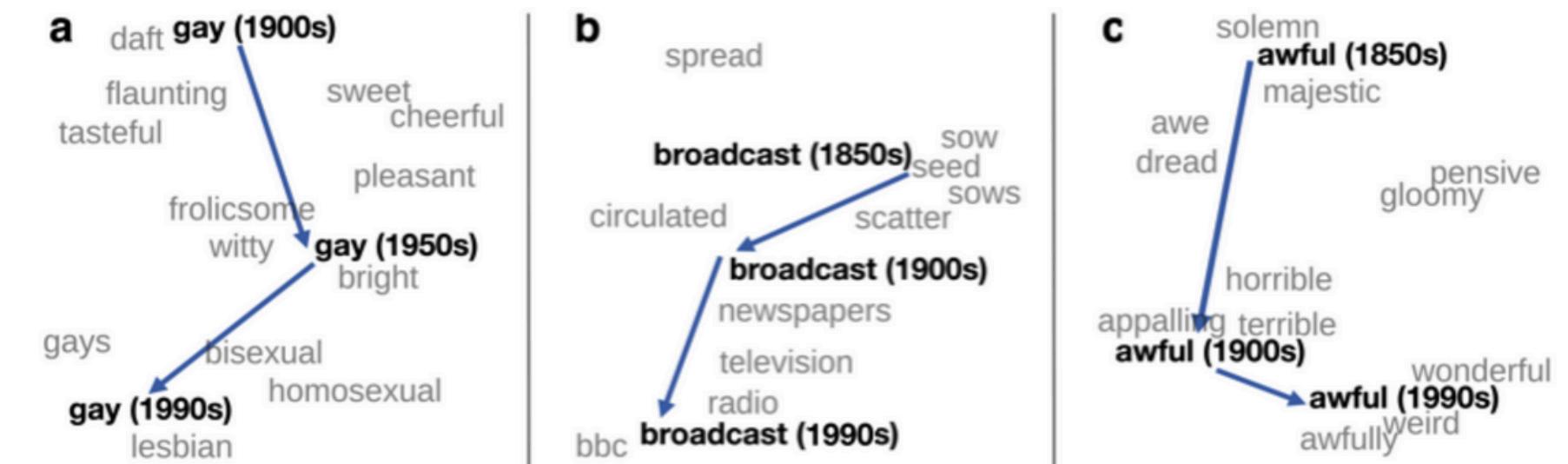
မြေစိမ်း



တောက်



J.R.Firth, 1957



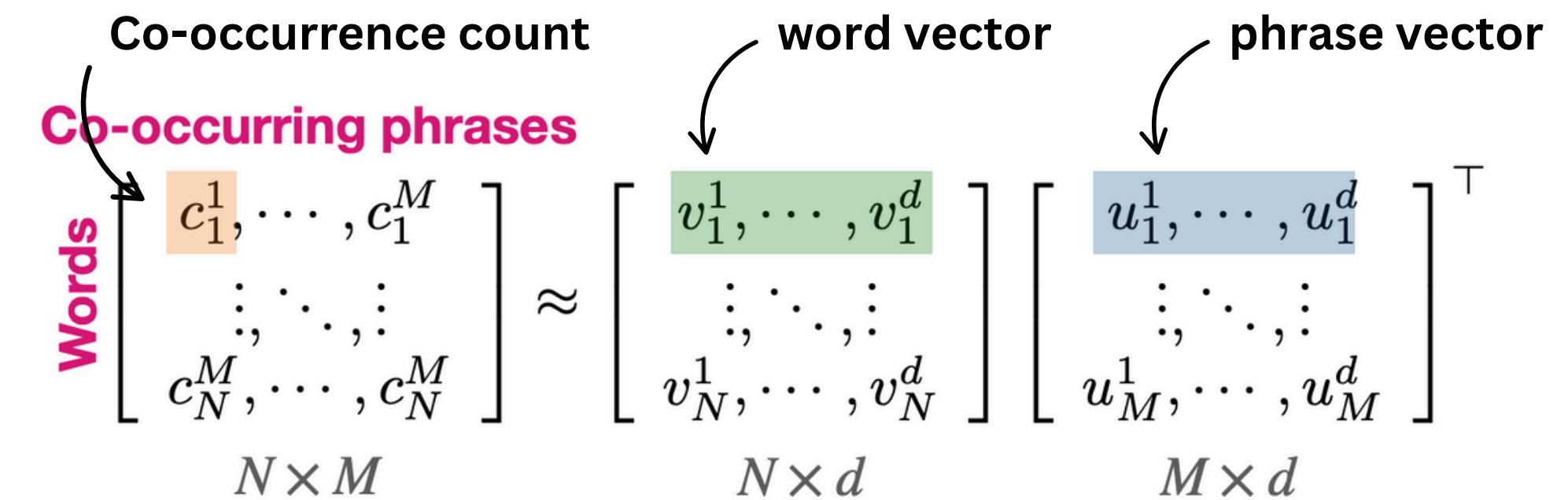
Hamilton WL, Leskovec J, Jurafsky D. 2016. Diachronic word embeddings reveal statistical laws of semantic change, In Proc. of ACL

# Word Vectors

- Word vector ဆုတေသန word co-occurrence matrix ထဲက row တစ်ခုဖြစ်ပါတယ်
  - ဆင်တူတဲ့ word vectors တွေရှိတဲ့ စကားလုံးတွေဟာ အဓိပ္ပာယ်ဆင်တူကြပါတယ်
  - Word vectors တွေဟာ sparse နဲ့ high-dimensional ဖြစ်ကြပါတယ်
  - စကားလုံးအများစုနဲ့ စကားစုတွေဟာ အတူတူမရှိကြတဲ့ အတွက် zeros တွေ အများကြီးပါဝင်လိုပါ
  - Solution: **Word Vector Compression** → ဆင်တူတဲ့ စကားလုံးတွေကို တစ်ခုနဲ့တစ်ခု နှီးကပ်စေ
  - **Co-occurrence count** ကို word နဲ့ phrase vectors က ခန့်မန်းမယ် [Mikolov et al., 2013]

Words		The	a	-	...	0
The	0	1	32	...	0	
a	3	0	...	...	0	
-	0	...	...	...	9	
...	:	:	:	:	:	
Gobbledygook	9	11	0	...	1	

Co-occurrence counts



# Word Vectors (Cont'd)

- Word vectors တွေဟာ compositional ဖြစ်ပါတယ်
- Compositionality ကို vector arithmetics အပြစ် သတ်မှတ်နိုင်ပါတယ်
- Word vector arithmetics (e.g. Russia - Beijing + China = Moscow)



Hinton

Querying analogy for: လက်ရှည် - လက်တို့ + ဘေးသီရှည်  
Analogies result:  
Word: ဝတ်, Score: 0.5567076206207275  
Word: ဖုန်းသီတေသန, Score: 0.5551380515098572  
Word: အကျိုး, Score: 0.5383158326148987  
Word: ထိုင်မသိမ်း, Score: 0.5366981625556946  
Word: ထဘိ, Score: 0.5333731174468994  
Word: အဝါး, Score: 0.5322700142860413  
Word: ဉာဏ်ပိုင်တဲ့, Score: 0.5307892560958862  
Word: ဝတ်ဆင်, Score: 0.5197224020957947  
Word: အကျိုး, Score: 0.519253671169281  
Word: ရော်ရောင်, Score: 0.5191075205802917

Vector for 'သီ':  
[ 0.03071883 -0.25971985 0.40830743 0.11242703 -0.12839618 -0.00550127  
0.05469747 -0.10745972 -0.15699431 0.3023142 -0.29259813 0.16690251  
0.07785437 -0.58479047 -0.7412099 -0.04547881 -0.03287222 -0.11044414  
-0.08638554 -0.02459145 -0.08973993 -0.02339625 -0.28159484 0.24005242  
0.0358529 -0.6657557 0.27617294 -0.59123087 0.1455806 0.18249877  
0.12091456 -0.4804478 -0.00294562 -0.56233424 0.278252 0.11192498  
0.14295132 0.05967402 -0.12034112 0.05577437 -0.293318 -0.31646758  
0.21822222 -0.40633217 0.17665516 0.2565288 -0.14461318 -0.08262646  
0.4971917 0.3262872 0.09940107 0.18980542 -0.04641127 0.9501322  
0.08271138 0.02752739 -0.17911749 -0.22868471 0.34966385 0.13285655  
-0.06082508 -0.1977889 0.9427577 0.50436085 0.2823574 -0.19498166  
0.1311327 -0.09055237 0.31270525 0.13989182 -0.84258276 0.30009115  
-0.3908675 0.14674334 -0.4403673 -0.3262866 -0.26670656 0.24394107  
0.18048933 -0.23953764 0.02176909 -0.29563728 0.10838298 0.12806904  
0.09947888 0.0433741 -0.11379734 0.20521459 0.01840506 0.24994595  
0.24415538 -0.19377986 -0.1143683 -0.30197594 0.12075205 -0.24896991  
-0.11793793 0.49135238 0.0792672 0.16126719]

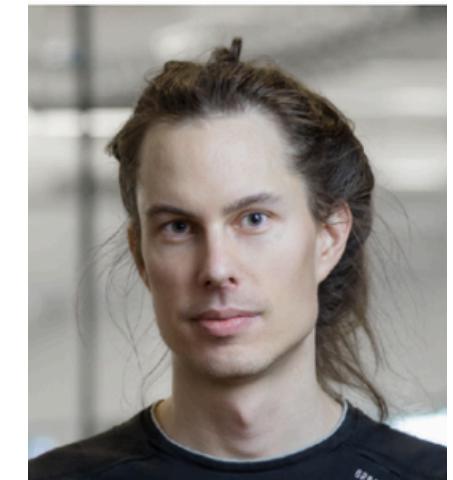
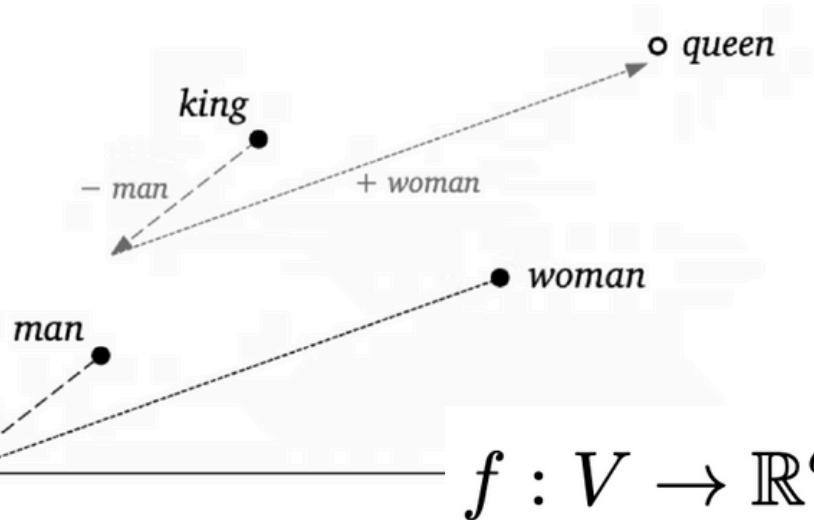
Nearest neighbors for 'အင်တာနှင်း':  
Score: 0.697070837020874, Word: မြန်မာနှင်း  
Score: 0.6930655837059021, Word: တာနှင်း  
Score: 0.6491116881370544, Word: ဘေးနှင်း  
Score: 0.6279654502868652, Word: အင်တာနာ  
Score: 0.6255528926849365, Word: တာဝါတိုင်  
Score: 0.609035849571228, Word: ကွန်ရက်လိုင်း  
Score: 0.598670482635498, Word: ဖွံ့ဖြိုးတွေ  
Score: 0.5967662334442139, Word: မှုဘိုင်းဖုန်း  
Score: 0.5892794132232666, Word: ဒေဝါ  
Score: 0.5861119627952576, Word: လူမှုကွန်ရက်

$$v_{\text{cat}} = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix} \quad v_{\text{dog}} = \begin{pmatrix} -0.124 \\ 0.430 \\ -0.200 \\ 0.329 \end{pmatrix}$$

$$v_{\text{the}} = \begin{pmatrix} 0.234 \\ 0.266 \\ 0.239 \\ -0.199 \end{pmatrix} \quad v_{\text{language}} = \begin{pmatrix} 0.290 \\ -0.441 \\ 0.762 \\ 0.982 \end{pmatrix}$$



[Hinton, Geoffrey E., "Learning Distributed Representations of Concepts."](#) [Proceedings of the Eighth Annual Conference of the Cognitive Science Society, Erlbaum, 1986, pp. 1-12.](#)

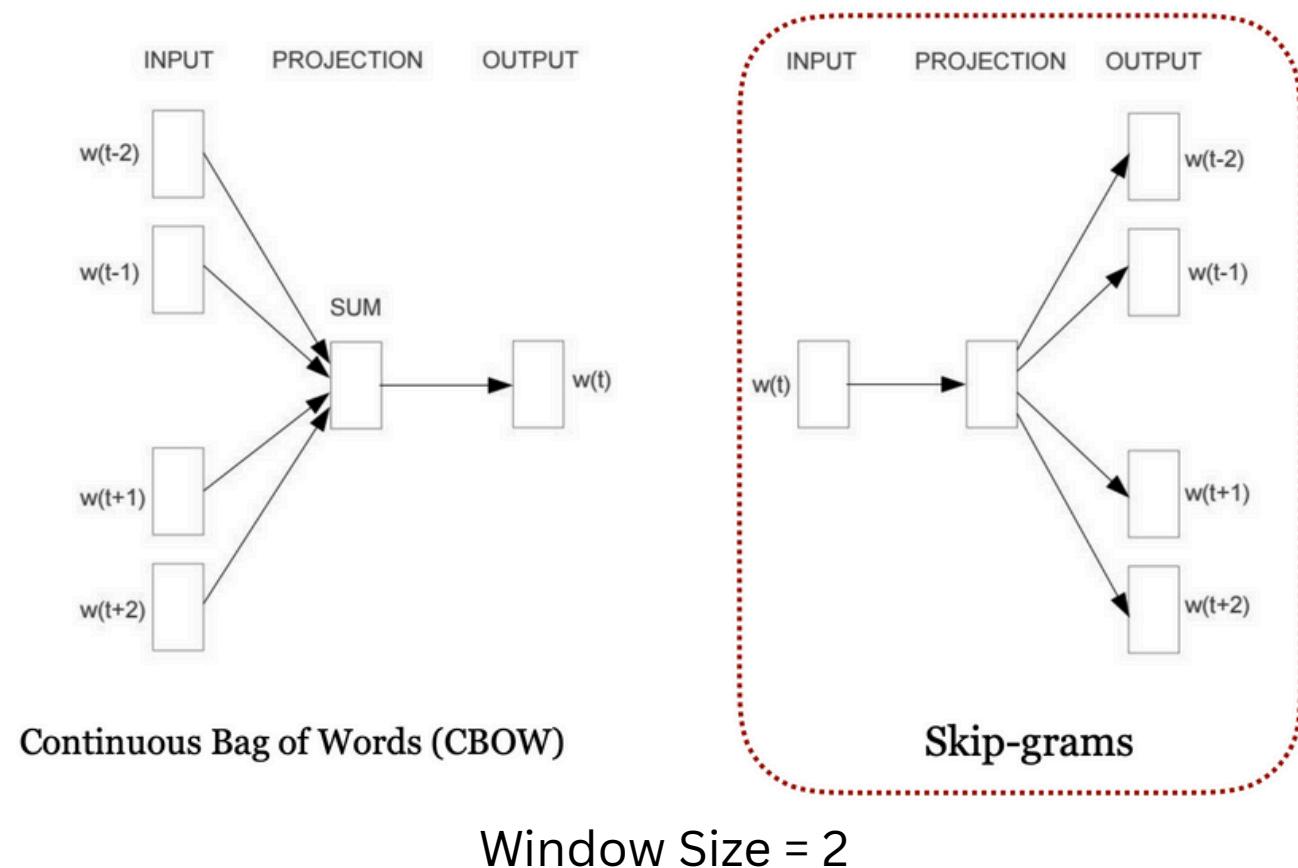


Mikolov

[Mikolov, T., Chen, K., Corrado, G., & Dean, J. \(2013\). Efficient Estimation of Word Representations in Vector Space.](#) In [1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings.](#)

# Word Vectors (Cont'd)

- Unsupervised learning ကို နှစ်ခုစလုံးက အသုံးပြုထားတဲ့အတွက် human label တွေ မလိုအပ်ပါဘူး။
- CBOW: context စကားလုံးများပေါ်မှုတည်ပြီး target စကားလုံးကို ခန့်မှန်းပါတယ်။
- CBOW က embeddings တွေကို သင်ယူပြီး ဆင်တူတဲ့ context တွေကနေ စကားလုံးတစ်ခုတည်းကို ခန့်မှန်းနိုင်ပါတယ်။
- Skipgram- target စကားလုံးတစ်ခုပေးထားရင် context စကားလုံးတွေကို ခန့်မှန်းပါတယ်။ (CBOW ခဲ့ ပြောင်းပြန်)
- Skip-gram က target စကားလုံးတစ်ခုစိုးအပေါ်အရုံးစိုးကို အတွက် ရှားပါးတဲ့စကားလုံးတွေအတွက် ပိုကောင်းပါတယ်။
- Word2Vec, GloVe, fastText



context ဟာ စကားလုံး တစ်ခုချင်းစိရဲ့ အဓိပ္ပာယ်ကို  
ဖြစ်ပေါ်စေပါတယ်

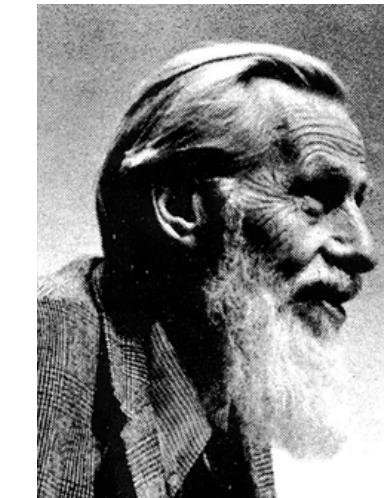
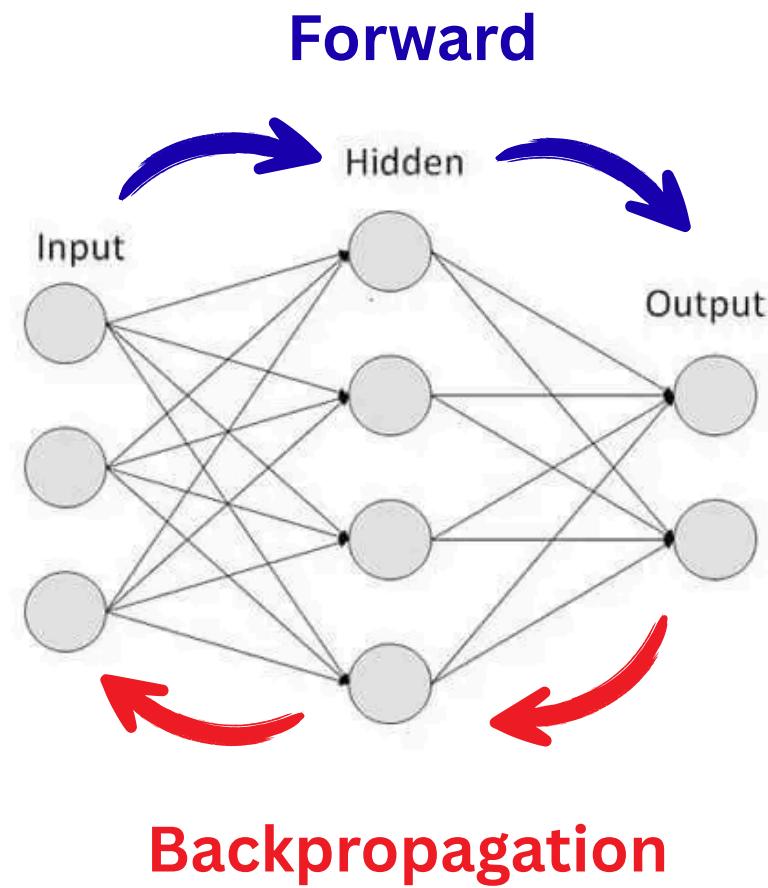
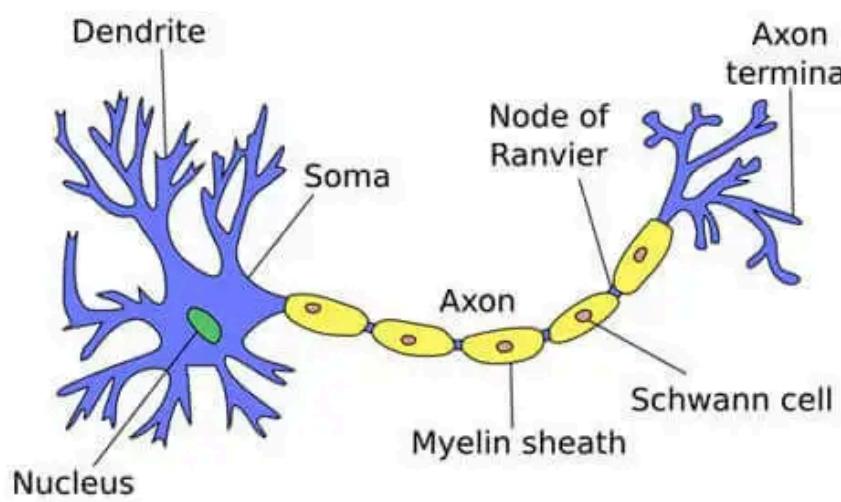
↑ ↓

စကားလုံး တစ်ခုချင်းစိနဲ့ ငြင်းတို့ရဲ့ အထားအသို့ က  
စသားရဲ့ အဓိပ္ပာယ် (context) ကို ဖြစ်ပေါ်စေပါတယ်

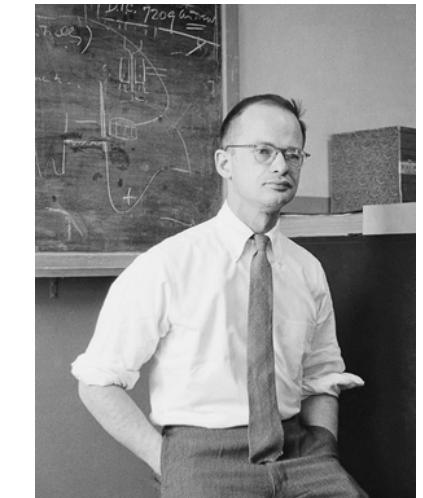
စကားလုံးတစ်ခုဟာ သူရဲ့ ဘေးပတ်ဝန်းကျင်က စကားလုံးတွေပေါ်မှာ အဓိပ္ပာယ်သက်ရောက်ပါတယ်

# Neural Networks

- လူခြီးနှောက်ကို အခြေခံပြီး၊ အချင်းချင်းချိတ်ဆက်ထားတဲ့ nodes (neurons) တွေရဲ့ layer တွေနဲ့ ဖွံ့ဖည်းထားပါတယ်
- Input Layer – Train data တွေကို လက်ခံရယူပါတယ် (ဥပမာ- စာသား၊ ပုံရဲ့ pixels တွေ)
- Hidden Layers – data ကို weights နဲ့ activation functions တွေအသုံးပြုပြီး လုပ်ဆောင်ပါတယ်
- Output Layer – ခန့်မှန်းချက်တွေကို ထုတ်လုပ်ပေးပါတယ် (ဥပမာ- class labels, ကိန်းဂဏ်နဲ့တွေ)



Warren McCulloch



Walter Pitts

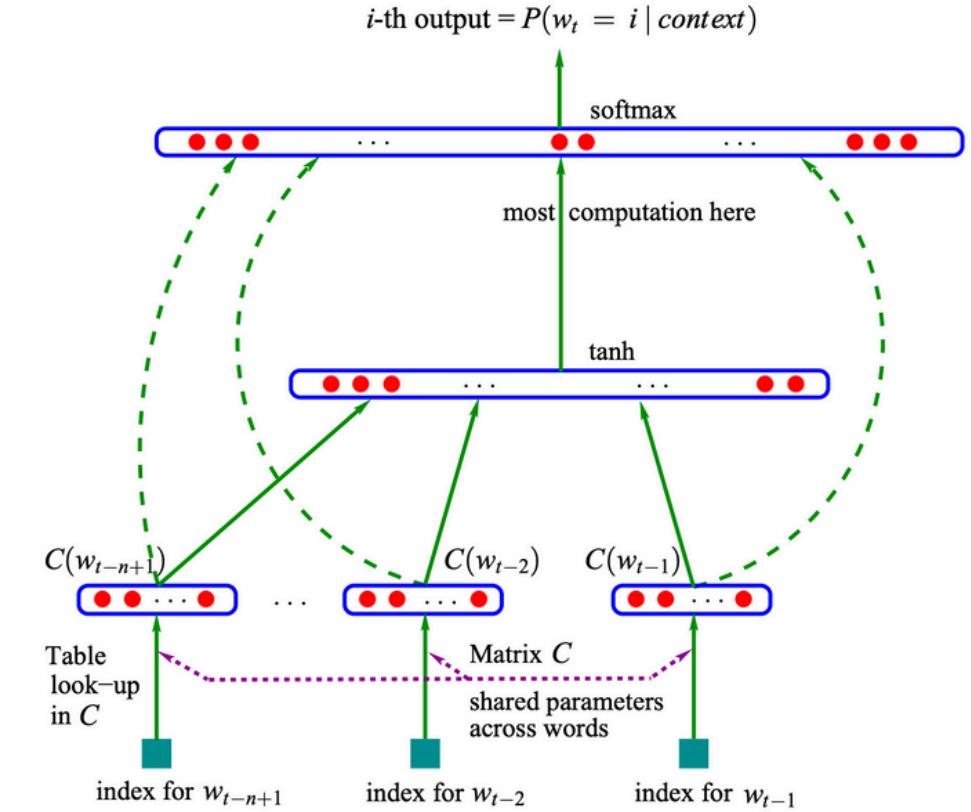
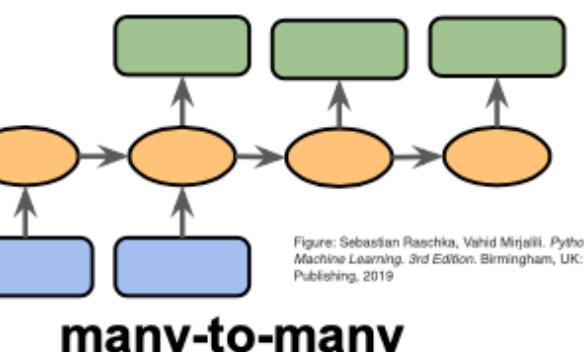
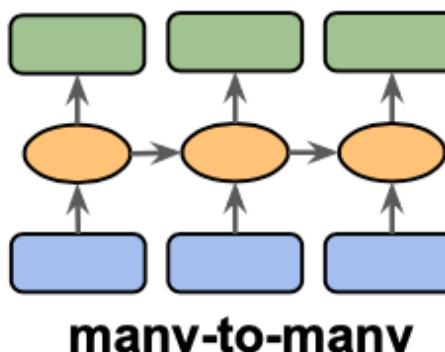
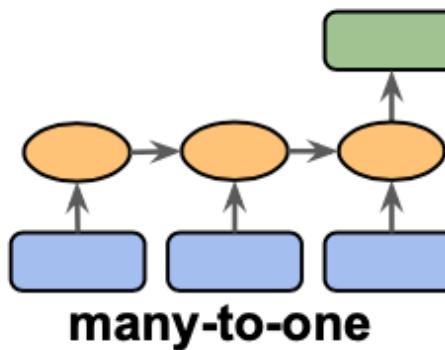
# Language Models

$$P(S) = \prod_{i=1}^n P(w_i | w_{i-1})$$

- ဘာသစကား မော်ဒယ် language model (LM) ဟာ အရင်စကားလုံးတွေ ကို အခြေခံပြီး နောက်ထပ်စကားလုံးကို ခန့်မှန်းပါတယ်
- RNNs (Recurrent Neural Networks) တွေက အရှေ့က အချက်အလက် တွေကို hidden state ကို အသုံးပြုပြီး မှတ်သားထားနိုင်တယ်
- ရိုးရှင်းတဲ့ feed-forward network နဲ့မတူဘဲ RNN က အချက်အလက်တွေ ကို စကားလုံးတစ်လုံးချင်းစိုက ယူပါတယ်။
- သူရဲ့ မှတ်ညက် (hidden state) ကို update လုပ်ပါတယ်။
- အဲဒီမှတ်ညက်ကို အသုံးပြုပြီး နောက်ထပ်စကားလုံးကို ခန့်မှန်းပါတယ်။
- အားနည်းချက် - Context ရှည်ရင် မမှတ်နိုင်



Bengio



# Polysemy



စကားလုံး/token တစ်ခုမှာ အဓိပ္ပာယ်တစ်ခုထက်ပိုရှိနေခြင်း

Homonymy သံတူ့ကြောင်းကွဲမဟုတ်

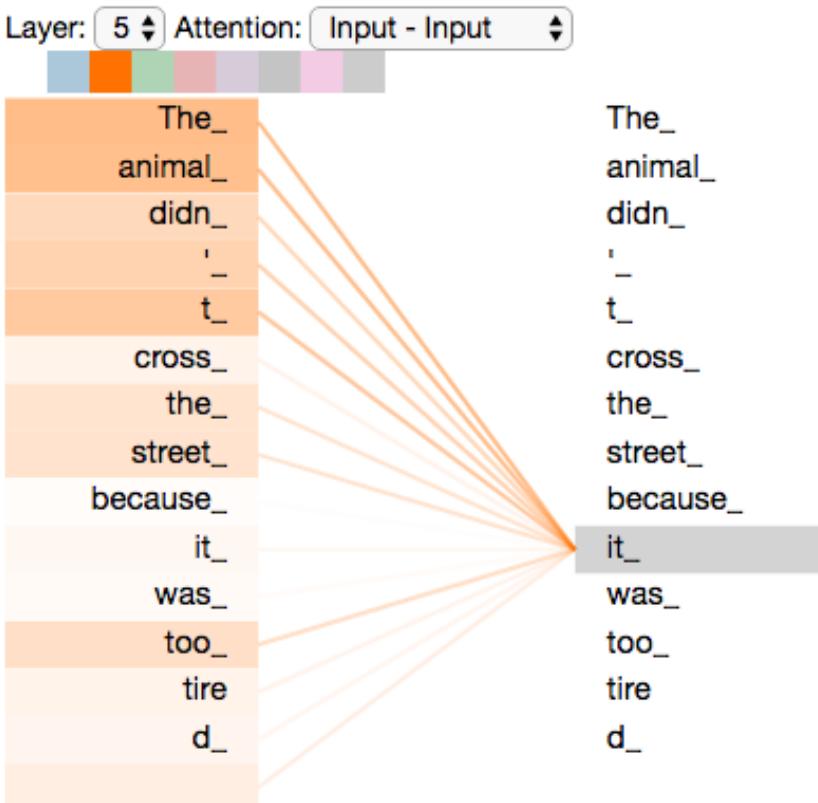
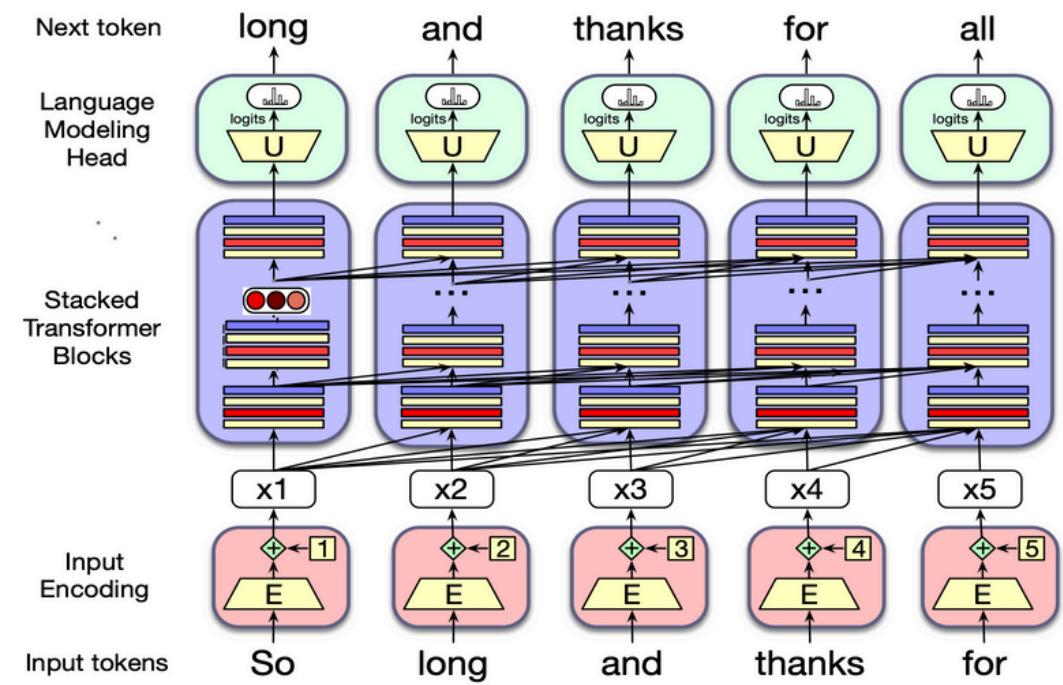
ဥပမာ

မင်း

- ကြိယာထောက်ပစ္စည်း၊ လွန်ကဲခြင်းအနက်ဖြင့်ကြိယာကိုထောက်ပံ့သောစကားလုံး → အိုမင်း၊ လွန်မင်း၊ ကြောက်မင်း။
- နာမ်စား၊ ရာထူးကြီးမြင့်သူကမိမိအောက်မှုငယ်သားတို့အားရည်ညွှန်းသောစကားလုံး → မင်းသိသမျှ ငါကို ပြော။
- နာမ်စား၊ မိမိကြီးတည်ပြောဆိုသူကိုရည်ညွှန်းသောစကားလုံး → သူငယ်ချင်း၊ မင်းနဲ့ငါ အတူသွားမယ်နော်။
- နာမဝိသေသန၊ ဂုဏ်ပြု ချို့မြင့်ခေါ်သော စကားလုံး → လူကြီးမင်း၊ ဦးမင်း။
- နာမ်၊ အစိုးရအမှုထမ်းအကြီးအကဲ → တပ်မင်း၊ ဗိုလ်မင်း အပေါင်းတို့ သည်လည်း။
- နာမ်၊ ဘုရင့်ဆွေတော်မျိုးတော် → သားတော်အိမ်ရှေ့မင်း။
- နာမ်၊ ဘုရင် → မင်းတို့စကား၊ မိုးကြိုးသွား။

# Transformer Embeddings

$$\text{softmax} \left( \frac{\mathbf{Q} \times \mathbf{K}^T}{\sqrt{d_k}} \right) \mathbf{V}$$



- **Self-attention** - တစ်ခုနဲ့တစ်ခု ပိုပြီး ဆင်တူတဲ့ tokens တွေကို အဓိပ္ပာယ်တစ်ခုတည်းဆီကို ရောက်အောင်ကြံးစားခြင်း
- အထောပိုင်း context tokens တွေက နောက်ပိုင်း tokens တွေရဲ့ embedding representations ကို သက်ရောက်မှုရှိပါတယ်
- သိအိုရီအရ context token တစ်ခုစီဟာ စာကြောင်းတစ်ခုလုံးရဲ့ အဓိပ္ပာယ်ကို ထည့်သွင်းနိုင်ပါတယ်
- Transformers တွေက tokens တွေကို တပြိုင်နောက်တည်းလုပ်ဆောင်ပါတယ် (တစ်လုံးပြီးမှတစ်လုံးဖတ်တဲ့ RNNs တွေနဲ့မတူပါဘူး)
- Problem → Transformers တွေက စကားလုံးအစီအစဉ်ကို မူလကတည်းက မသိပါ
- **Solution → Positional Encoding:** မော်ဒယ်က tokens တွေရဲ့ အစီအစဉ်ကို သိအောင်လို့ input embeddings တွေထဲကို positional signal တစ်ခု ထည့်ပေးရပါမယ်

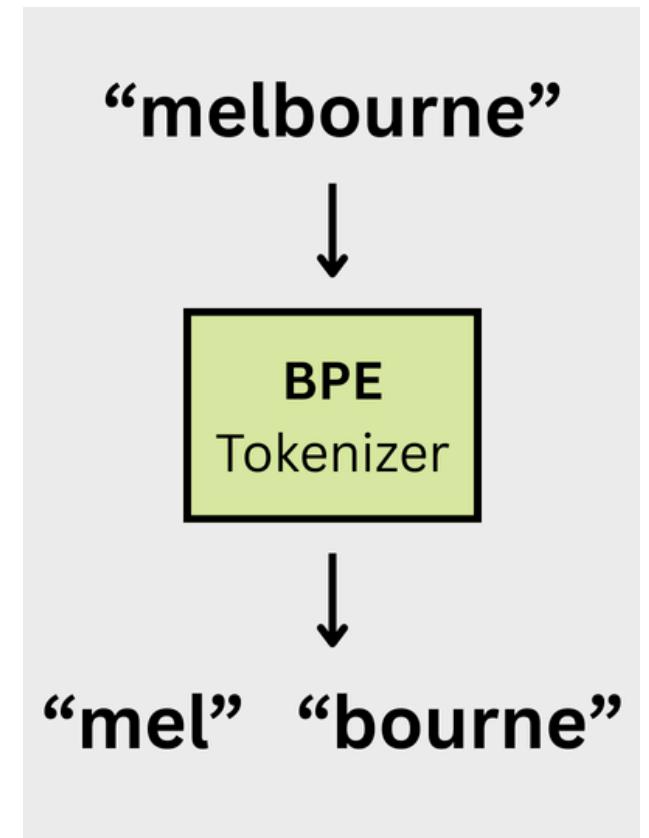
<https://jalammar.github.io/illustrated-transformer/>

<https://web.stanford.edu/~jurafsky/slp3/slides/transformer24aug.pdf>

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention is all you need*. In Advances in Neural Information Processing Systems (pp. 5998–6008).

# Subword tokenization

- အသုံးများတဲ့ algorithm သုံးမျိုးရှိပါတယ်
  - Byte-Pair Encoding (BPE) (Sennrich et al., 2016)
  - Unigram language modeling tokenization (Kudo, 2018)
  - WordPiece (Schuster and Nakajima, 2012)
- Tokenization Algorithm သုံးမျိုးစလုံးမှာ ပါဝင်တဲ့ အစိတ်အပိုင်းနှစ်ခု ရှိပါတယ်
  - A token learner: Raw training corpus တစ်ခုကိုယူပြီး ဝေါဟာရ (tokens အစ္စအဝေး) ကို ထုတ်လုပ်ပေးပါတယ်
  - A token segmenter: Raw test sentence တစ်ခုကိုယူပြီး အဲဒီဝေါဟာရအရ tokenization လုပ်ပေးပါတယ်
- များသောအားဖြင့် အသုံးများတဲ့ စကားလုံးတွေ ပါဝင်ပါတယ်
- အသုံးများတဲ့ subwords တွေလည်း ပါဝင်ပါတယ်
- ငှုံးတို့ဟာ မကြောခဏဆိုသလို -est သို့မဟုတ် -er လိုမျိုး morpheme တွေဖြစ်တတ်ပါတယ်
- Morpheme ဆိုတာ ဘာသာစကားတစ်ခုရဲ့ အသေးငယ်ဆုံး အဓိပ္ပာယ်ရှိတဲ့ အစိတ်အပိုင်းတစ်ခုပါ
- ဥပမာ- "unlikeliest" မှာ "un-", "likely" နဲ့"-est" ဆိုပြီး morpheme သုံးခုရှိပါတယ်



# Subword tokenization (Cont'd)

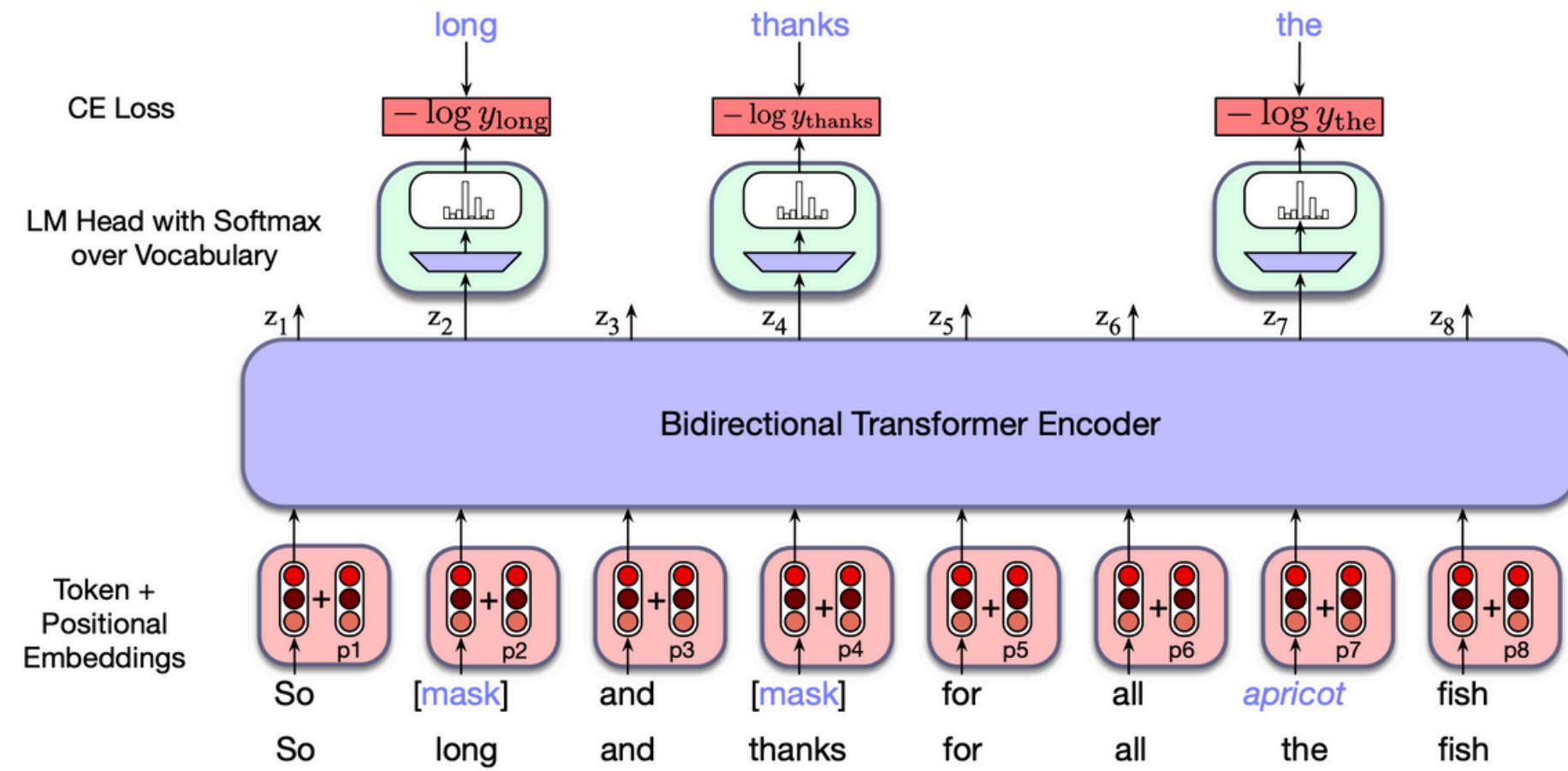
## Byte Pair Encoding (BPE) token learner

- ဝါဟာရကို တစ်ဦးချင်းစီ အကွဲရာအားလုံးရဲ့ အစုအဝေးအဖြစ် သတ်မှတ်လိုက်ပါ။ ဥပမာ- {က, ခ, ဂ, ယ...}
- အောက်ပါအဆင့်တွေကို ထပ်ခါတလဲလဲ လုပ်ဆောင်ပါ။
  - training corpus ထဲမှာ အများဆုံး တွဲဖက်တွေ့ရတဲ့ သက်တန်စုံ (ဥပမာ- 'က'၊ 'ခ') ကို ရွေးချယ်ပါ။
  - ပေါင်းစည်းထားတဲ့ သက်တအသစ် 'ကခ' ကို ဝါဟာရထဲကို ထည့်သွင်းပါ။
  - corpus ထဲမှာ တွဲနေတဲ့ 'ကခ' တွေကို 'ကခ' နဲ့ အစားထိုးပါ။
- သတ်မှတ်ထားတဲ့ ပေါင်းစည်းမှု (k merges) ပြည့်တဲ့ အထိ လုပ်ဆောင်ပါ။

## Byte Pair Encoding (BPE) token segmentar

- Test data ပေါ်မှာ training data ကနေ သင်ယူခဲ့တဲ့ ပေါင်းစည်းမှုတွေကို အောက်ပါအတိုင်း run ပါ
  - Greedily
  - သင်ယူခဲ့တဲ့ အစဉ်လိုက်အတိုင်း
  - (test frequencies တွေက အခန်းကဏ္ဍကနေ မပါဝင်ပါ)
- ဒါကြာ့၊ ပထမဆုံး 'ဝေရ' ကို 'ရေ' အဖြစ် ပေါင်းစည်းပါ၊ ပြီးတော့ 'ရွက်ပြာ' ကို 'ရွက်ပြာ' အဖြစ် ပေါင်းစည်းပါ။ ရလဒ်-
- Test set က "က နီ" ကို tokenized လုပ်ရင် တစ်လုံးတည်းဖြစ်တဲ့ "ကနီ" ဖြစ်ပါမယ်။
- Test set က "က ယ် ဆယ် ရေး" ကို tokenized လုပ်ရင် "ကယ်" နဲ့ "ဆယ်ရေး" ဆိုပြီး tokens နှစ်ခုဖြစ်လာနိုင်ပါတယ်။

# Masked Language Modeling



- ဘာသာစကား နားလည်မှုအတွက် Transformer-based ဖော်ဆိပ်တစ်ခုပါတယ်
  - စာသားကို နှစ်လမ်းသွား လုပ်ဆောင်ပါတယ် (ဘယ်ဘက်နဲ့ ညာဘက် context ကို ထည့်သွင်းစဉ်းစားပါတယ်)
  - contextual word embeddings တွေကို ထုတ်လုပ်ပေးပါတယ်
  - အသုံးများတဲ့ subwords တွေ သုံးမယ်
- ကွက်လပ်ဖြည့်ပါ
- မနက်ခင်းမှာ ကျွန်တော် \_\_\_\_\_ သောက်တယ်။
  - ကျောင်းမှာ ကျွန်တော်တို့ \_\_\_\_\_ လွှဲလာတယ်။
  - ကျွန်မ မိခင်ကို \_\_\_\_\_ လို့ ခေါ်တယ်။ (အမော မေမေ...)



Devlin et al., 2019 – BERT: Pre-training of Deep Bidirectional Transformers. arXiv:1810.04805  
<https://jalammar.github.io/illustrated-bert/>

# Masked Language Modeling (Cont'd)

## Pretraining

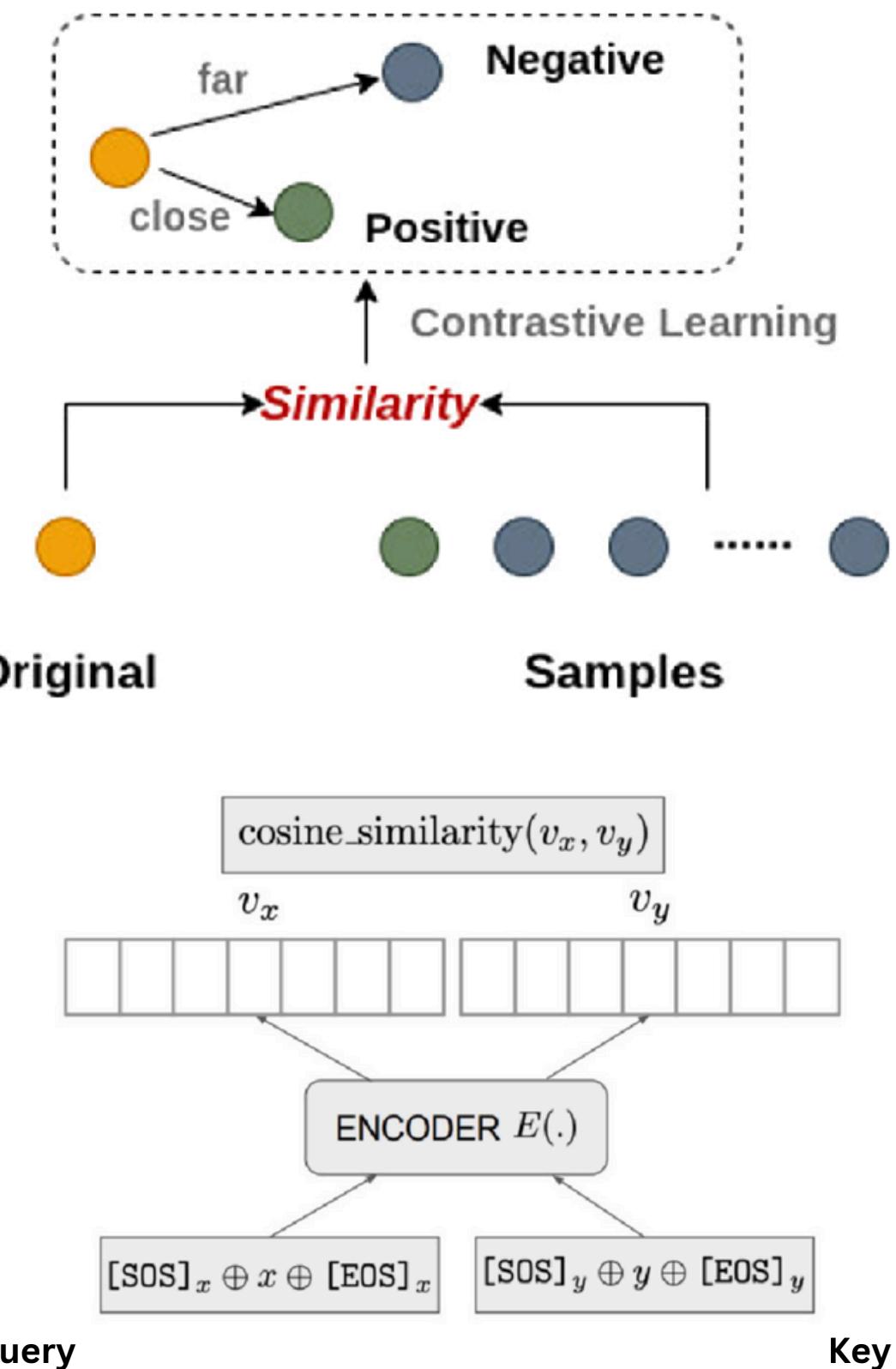
- ဘာသစကား representations တွေကို သင်ယူဖို့
- corpus ကြီး တွေပေါ်မှာ တစ်ကိုမဲ့ လုပ်ဆောင်ပါတယ် (unsupervised)
- Masked Language Modeling (MLM)- စာကြောင်းထဲက ဖုံးကွယ်ထားတဲ့ စကားလုံးတွေကို ခန့်မှန်းခြင်း
- Next Sentence Prediction (NSP)- စာကြောင်း B က စာကြောင်း A ရဲ့ နောက်မှာ ပါမပါ ခန့်မှန်းခြင်း

## Finetuning

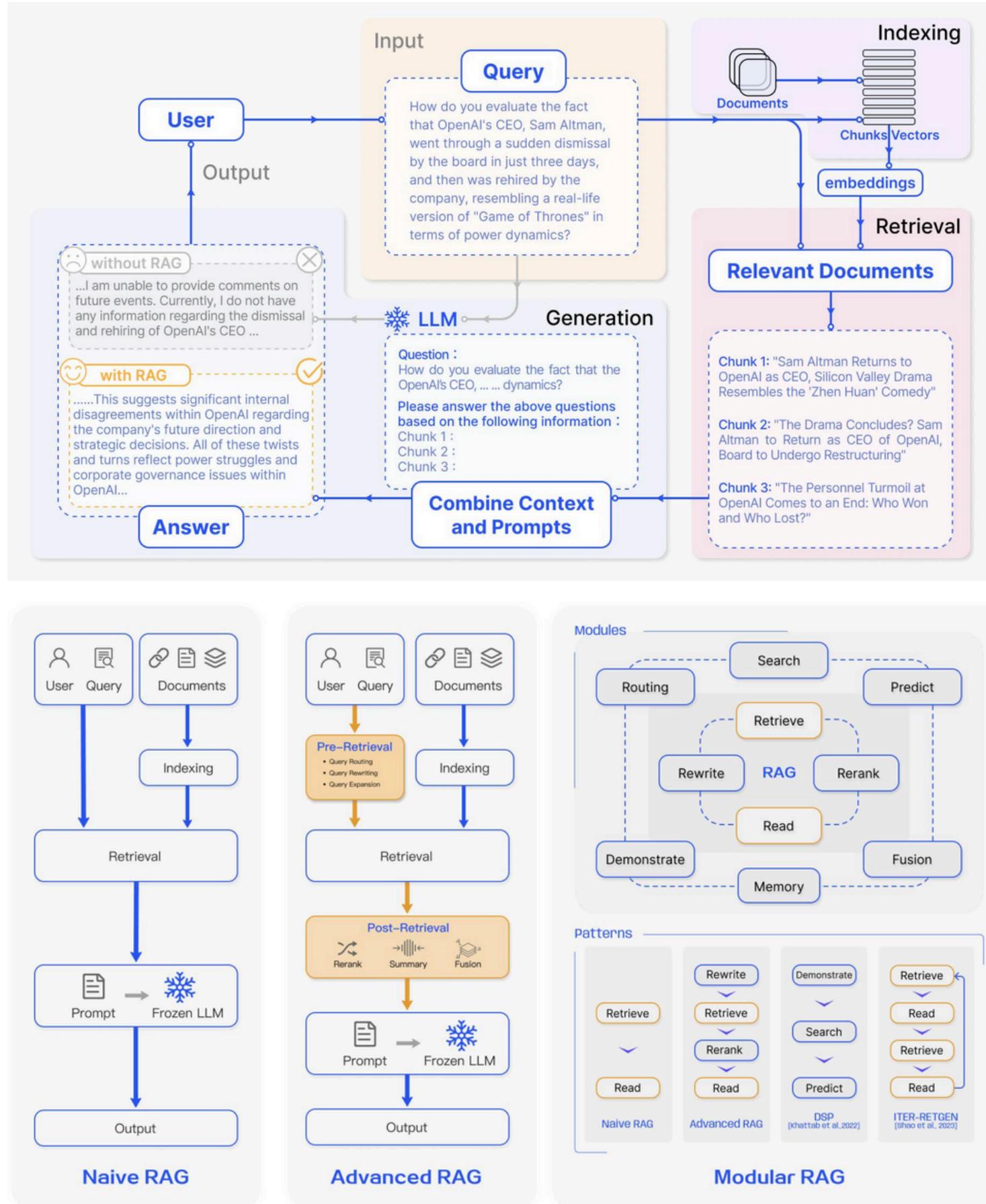
- သီးခြား downstream task တစ်ခုချင်းစီအတွက် လုပ်ဆောင်ပါတယ် (supervised)
- သီးခြား downstream task အတွက် သီးခြား layer တစ်ခုကို ထပ်ထည့်ရပါတယ် (ဥပမာ- classification, QA)
- သီးခြား downstream task ရဲ့ data နဲ့ train ပြီး BERT ရဲ့ weights အားလုံး ဒါမှုမဟုတ် အချို့ကို update လုပ်ပါတယ်
- Text Classification, POS Tagging, NER Tagging, Question Answering.

# Sentence Embeddings

- Vector Space မှာ ဘေးချင်းကပ်နေတဲ့ စာကြောင်းတွေဟာ ဆင်တူကြပါတယ်
- Vector Space မှာ ဘေးချင်းမကပ်တဲ့ စာကြောင်းတွေကတော့ မဆင်တူကြပါဘူး
- Transformer ရဲ့ pre-trained embeddings တွေနဲ့ initialize ပြီး contrastive learning ကို အသုံးပြုပြီး စာကြောင်းတွေကြားက embeddings တွေကို သင်ယူပါတယ်



# RAG



အမေးအဖြတ် ဖြတ်အခါ (သို့) စာသားတွေကို ထုတ်ပေးတဲ့အခါမှာ၊

- RAG က ပထမဆုံးအနေနဲ့ သက်ဆိုင်ရာ အချက်အလက်တွေကို Database/Documents တွေထဲက နေရာဖွေယူပါတယ်
- ပြီးမှ အဲဒီအချက်အလက်တွေပေါ်အခြေခံပြီး LLM တွေက အဖြတ်တွေကို ထုတ်ပေးပါတယ်။
- RAG ဟာ LLM တွေနဲ့ အပြင်ဘက်က ဗဟိုသုတေသန အရှင်းအမြစ် (ဥပမာ- စာအုပ်တွေ၊ ဆောင်းပါးတွေ) ကို ပေါင်းစပ်လိုက်တာဖြစ်ပါတယ်။
  - Retrieval (ရာဖွေခြင်း) - ပထမအဆင့်မှာ မေးခွန်းနဲ့ သက်ဆိုင်တဲ့ အချက်အလက်တွေကို သီးခြားအတော့သွေ့ (knowledge base) ထဲကနေ ရာဖွေစုဆောင်းပါတယ်။
  - Generation (ထုတ်ပေးခြင်း) - ဒုတိယအဆင့်မှာတော့ ရာတွေထားတဲ့ အချက်အလက်တွေကို LLM ကို ထည့်သွင်းပေးပြီး စနစ်တကျနဲ့ ယုတ္တိရှိတဲ့ အဖြတ်တွေဖို့ ညွှန်ကြားပါတယ်။

**Thank You**

**Q and A**