

VarCamp 2025

An Open Space for Tech, Creativity & Beyond

NLP & LLMs

Ye Kyaw Thu
Lab Leader, LU Lab., Myanmar
Visiting Professor, NECTEC, Thailand

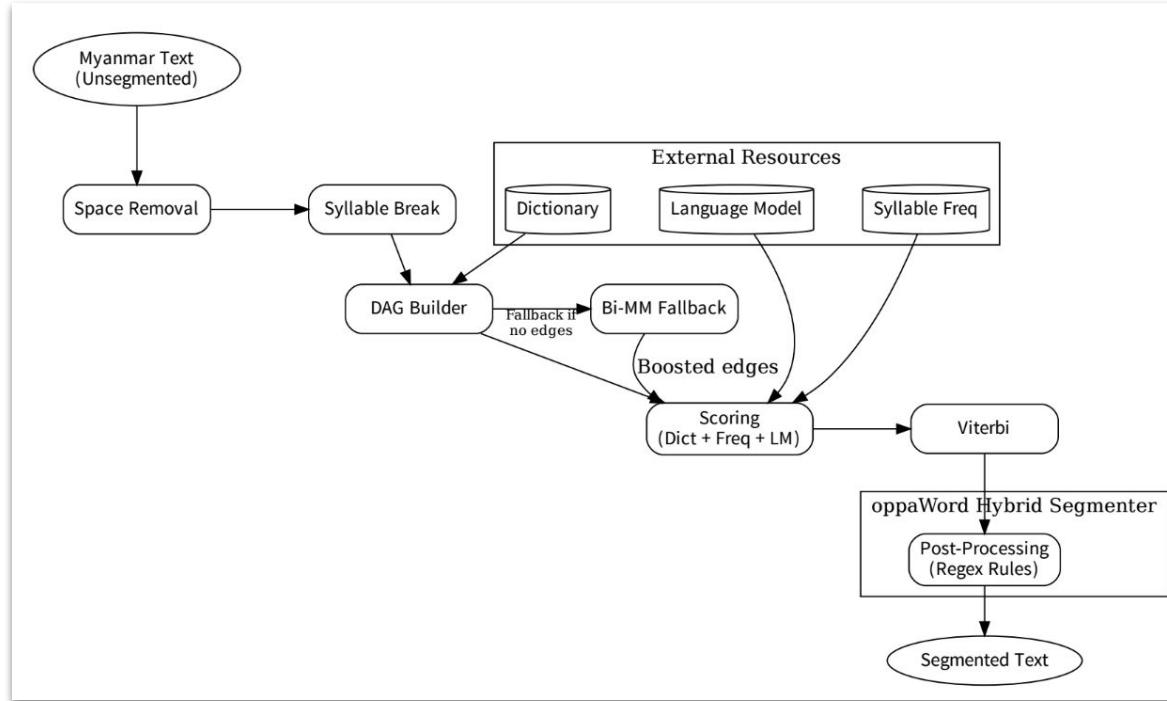
7 Sept 2025 (Sun)

Table of Content

- Differentiation between LLM and NLP
- Word Segmentation
- Statistical Language Model
- Word Frequency
- Word Cloud
- Embedding Techniques
- Transformer
- LLMs
- R&D with LLMs
- Summary

Word Segmentation

oppaWord



- Word segmentation အလုပ်က အရောင်းပါတယ
- Syllable, sub-word, word, BPE etc.

Fig. Overview of oppaWord Myanmar Word Segmenter
Github Link: <https://github.com/ye-kyaw-thu/oppaWord>

Word Segmentation

oppaWord

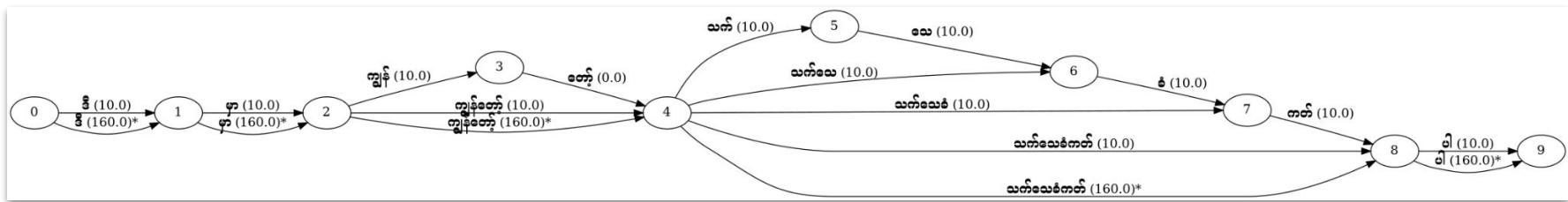


Fig. Word Segmentation Path

Github Link: <https://github.com/ye-kyaw-thu/oppaWord>

- oppaWord ရဲ ကောင်းတဲ အချက်က စာကြားရေ တစ်သိန်းကျော်လည်း ၃ မွန်အထွင်း ဖြတ်လိပ်းပြီးတယ်
- ကယ် domain ပေါ်မှတည်ပြီး အဘိဓာန်ကို update လုပ်လိုရတယ်
- RE သုံးပြီး post-editing လုပ်လို ရတယ်

Natural Language Processing & Large Language Model

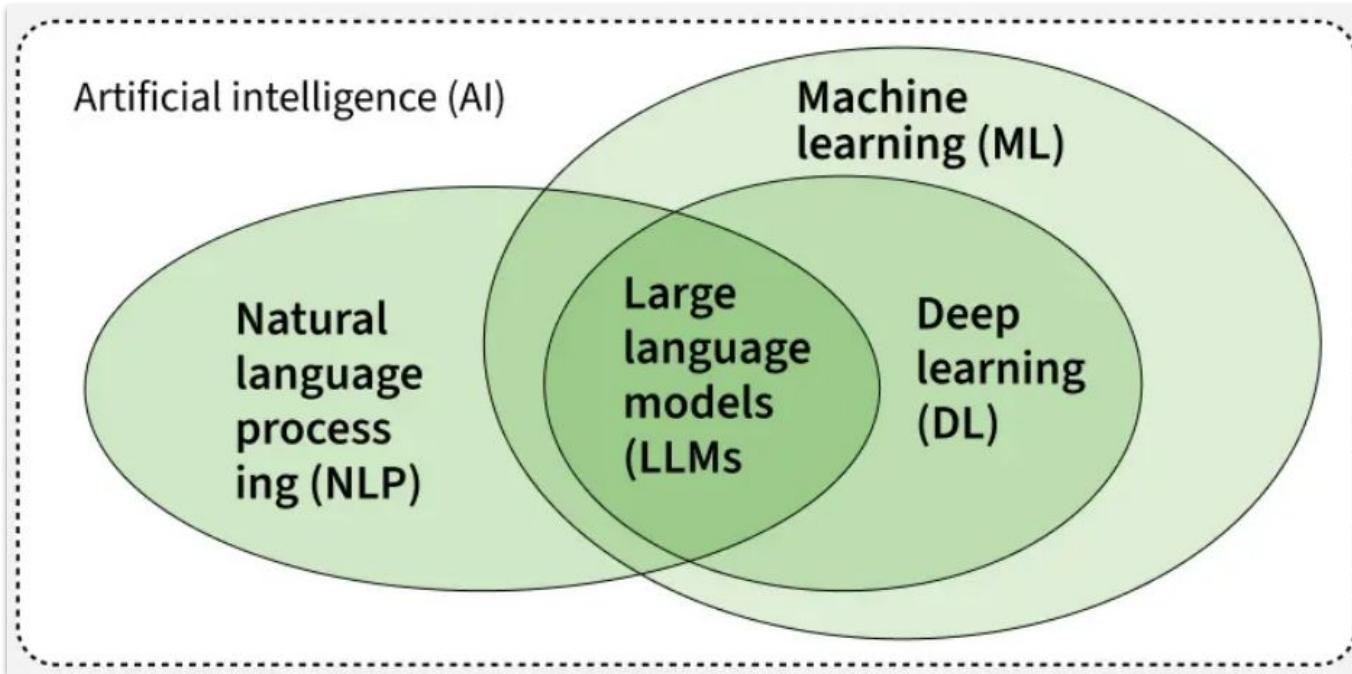
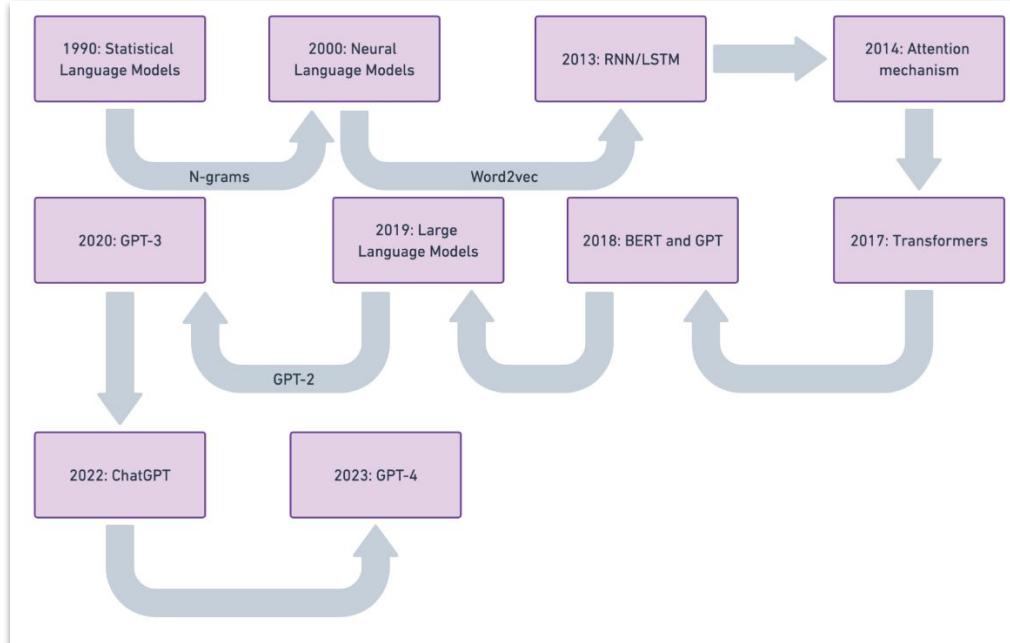


Fig. Differentiation between LLM and NLP

(source: <https://www.geeksforgeeks.org/nlp/nlp-vs-lm/>)

Natural Language Processing & Large Language Model



- ကိုယ်တိုင်က statistical LM ထွေကင် Neural LMs, Pretrained LMs, LLMs တွေအားလုံးနဲ့ NLP အလုပ်ထွေအများကြီး လုပ်ဖို့

Fig. A chronological timeline showcasing the evolution of Large Language Models (LLMs) from 1990 to 2023.

(source: <https://arxiv.org/pdf/2408.13296.pdf>)

Statistical Language Model (Ngram)

| | | |
|----|------|---|
| 1 | 6539 | → ရွှေ သည် ပဲခူး တိုင်း ဒေသ ကြီး အ |
| 2 | 4744 | → သည် ရွှေ နေရာ ကုတ် မှာ ၂ ၀ |
| 3 | 4672 | → သည် ရွှေ နေရာ ကုတ် မှာ ၁ ၈ |
| 4 | 4424 | → ရွှေ သည် ရှမ်း ပြုပုံ နယ် အ ရှေ့ |
| 5 | 4305 | → သည် ရွှေ နေရာ ကုတ် မှာ ၂ ၁ |
| 6 | 4087 | → သည် ရွှေ နေရာ ကုတ် မှာ ၁ ၇ |
| 7 | 3660 | → သည် ပဲခူး တိုင်း ဒေသ ကြီး အ နောက် |
| 8 | 3511 | → သည် ၂ ၀ ၁ ၄ သန်းခေါင်စာရင်း အရ |
| 9 | 3136 | → ဖြစ် သည် ရွှေ နေရာ ကုတ် မှာ ၁ |
| 10 | 3086 | → တည်ရှိ သည် ဘူတာ တစ် ခု ဖြစ် သည် |
| 11 | 3083 | → တွင် တည်ရှိ သည့် ဘူတာ တစ် ခု ဖြစ် |
| 12 | 3033 | → သည် ပဲခူး တိုင်း ဒေသ ကြီး အ ရှေ့ |
| 13 | 2608 | → သည် ရွှေ နေရာ ကုတ် မှာ ၁ ၉ |
| 14 | 2572 | → သည် ရွှေ နေရာ ကုတ် မှာ ၁ ၆ |
| 15 | 2529 | → ပြုပုံ နယ် အ ရှေ့ ကျိုးင်း တဲ့ ခရီး |
| 16 | 2529 | → ရှမ်း ပြုပုံ နယ် အ ရှေ့ ကျိုးင်း တဲ့ |
| 17 | 2529 | → သည် ရှမ်း ပြုပုံ နယ် အ ရှေ့ ကျိုးင်း |
| 18 | 2477 | → မြို့ တွင် တည်ရှိ သည့် ဘူတာ တစ် ခု |
| 19 | 2191 | → ရှမ်း ပြုပုံ နယ် တောင် လွှိုင် လင် ခရီး |
| 20 | 2191 | → သည် ရှမ်း ပြုပုံ နယ် တောင် လွှိုင် လင် |
| 21 | 2185 | → ခဲ့ သည် ရွှေ နေရာ ကုတ် မှာ ၁ |
| 22 | 2176 | → ရွှေ သည် ရှမ်း ပြုပုံ နယ် တောင် လွှိုင် |

| | |
|-------|---|
| 37582 | 9 → သထံ ချိုင် သထံ မြို့နယ် သိမ်ဆိပ် ကျေးရွှေ အပ် |
| 37583 | 9 → သထံ မြို့နယ် နောင် ဘို့ ကျေးရွှေ အပ် စုံ၏ |
| 37584 | 9 → သထံ မြို့နယ် သိမ်ဆိပ် ကျေးရွှေ အပ် စုံ၏ တည်ရှိ |
| 37585 | 9 → သဆ္မာ စ အ မဟာ ဓမ္မရာဇာ စီ ရာဇ် |
| 37586 | 9 → သနုတ်း ရွှေ နေရာ ကုတ် မှာ ၂ ၁ |
| 37587 | 9 → သန္တာသား အနေအထား မှန် အောင် အပိုင် ဖက် မှ |
| 37588 | 9 → သန်းခေါင်စာရင်း အရ ကျောင်း ကုန်း ကျေးရွှေအပ်စုံ တွင် ကျား |
| 37589 | 9 → သန်းခေါင်စာရင်း အရ သရက် ကုန်း ကျေးရွှေအပ်စုံ တွင် ကျား |
| 37590 | 9 → သပြု သာ ရွှေ သည် မန္တလေး တိုင်း ဒေသ |
| 37591 | 9 → သဖုန်း ပင် ဆိပ် ကျေးရွှေ အပ် စုံ၏ တည်ရှိ |
| 37592 | 9 → သဘာဝ ဓာတ်းခေါင်းပစ္စာရိုး များ အားလုံး ပါ ရ |
| 37593 | 9 → သဘာဝ ပစ္စာရိုး တွေ ကို အကြော် ပြီး ထုတ် |
| 37594 | 9 → သဘာဝ ပတ်ဝန်းကျင် ထိန်းသိမ်း ရေး ဝန်ကြီးဌာန သစ် တော်း |
| 37595 | 9 → သဘာဝ ပါ ပဲ ရွှေ နေရာ ကုတ် မှာ |
| 37596 | 9 → သမထ ဘာဝနှင့် စိတ် ပိုပိသနာ ဘာဝနာ စိတ် ထို့ |
| 37597 | 9 → သမ္မတ မြို့မာနိုင်ငံ တော် ပြုပုံထောင်စု အစိုးရအဖွဲ့ အစဉ်းအဝေး အမှတ် |
| 37598 | 9 → သမ္မတ ပါ စာ သမ္မတ ကမ္မန် သမ္မတ အာ |
| 37599 | 9 → သမ္မတ သ မာ စီ သမ္မတ ဘု အ |
| 37600 | 9 → သရက် ချောင်း မြို့နယ် အစ် အစ် ကရင် ပ |
| 37601 | 9 → သရက် တေားရွှေ သည်မ ကျေး တိုင်း ဒေသ ကြီး |
| 37602 | 9 → သရက် ပင် ကျေးရွှေ အပ် စုံ၏ တည်ရှိ သည် |
| 37603 | 9 → သရေ လ အ မြို့မြို့ ဘုရား သ ခင့် |

Fig. ngram Counting

Statistical Language Model (ARPA Format)

| | | | | | |
|----|------------------|---------|-------------|----------|------------------------------|
| 1 | \data\ | 3927799 | \3-grams: | 11017987 | -0.85703826 |
| 2 | ngram·1=207187 | 3927800 | -2.0410547 | 11017988 | လင်း ပတ် ဟူသော |
| 3 | ngram·2=3720602 | 3927801 | -0.9211863 | 11017989 | က သိုးဖေ ဟူသော |
| 4 | ngram·3=12365250 | 3927802 | -1.9860684 | 11017990 | အော့ ဗားမား ဟူသော |
| 5 | | 3927803 | -1.740716 | 11017991 | ရှင်း ရဲ့ ဟူသော |
| 6 | \1-grams: | 3927804 | -0.12082286 | 11017992 | ဟိ တ္ထု ဟူသော |
| 7 | -6.608784 | 3927805 | -2.0240984 | 11017993 | -0.62770605 |
| 8 | ><unk> | 3927806 | -1.9391153 | 11017994 | စန္ဒရား ဟူသော |
| 9 | 0 | 3927807 | -2.4646177 | 11017995 | -1.4885107 |
| 10 | -1.4534098 | 3927808 | -1.1716466 | 11017996 | -1.3004675 |
| 11 | -2.9836152 | 3927809 | -1.8926232 | 11017997 | ခေတ်သစ် အက္ခဏသံ့ဌာ ဟူသော |
| 12 | -2.9946578 | 3927810 | -2.0728502 | 11017998 | -1.4840224 |
| 13 | -3.8384993 | 3927811 | -1.6670866 | 11017999 | -0.82077515 |
| 14 | လျှောက့်ခန်း | 3927812 | -2.3193204 | 11018000 | ဂော်ကာ ဟူသော |
| 15 | -2.9895 | 3927813 | -2.5437999 | 11018001 | -0.82077515 |
| 16 | -1.0622561 | 3927814 | -1.1367344 | 11018002 | အထိ ဂော်ကာ ဟူသော |
| 17 | -2.7952466 | 3927815 | -1.508018 | 11018003 | -0.61718965 |
| 18 | ရှင်း | 3927816 | -2.6039376 | 11018004 | မြတ်စိုးရီး ဟူသော |
| 19 | -1.1160072 | 3927817 | -2.255387 | 11018005 | -0.34802505 |
| 20 | -2.1484232 | 3927818 | -1.9319197 | 11018006 | ခါ်ကာ မြတ်စိုးရီး ဟူသော |
| 21 | -0.91495144 | 3927819 | -1.8834062 | 11018007 | -0.35268822 |
| 22 | -4.6132345 | 3927820 | -1.8680185 | 11018008 | -0.7618375 |
| 23 | အလေးမ | 3927821 | -2.0713744 | 11018009 | ဖြစ် တတ်” ဟူသော |
| | | | | | -1.1923101 |
| | | | | | <S> ရောင်းရှင်း ဟူသော |
| | | | | | -0.6090044 |
| | | | | | တရား နှစ်ပါးစုံ ဟူသော |
| | | | | | -1.7485399 |
| | | | | | -0.6221347 |
| | | | | | sympathetic system ဟူသော |
| | | | | | -0.6221347 |
| | | | | | Parasympathetic system ဟူသော |
| | | | | | -1.0945097 |
| | | | | | <S> လက်တိ ဟော |

Fig. 3-gram LM with ARPA format

Statistical Language Model (Next Word Prediction)

```
=====
KenLM Text Generation Experiments
Model: ./myMono.v0.1.6gram.arpa
Prompt: ကျွန်တော်
Log file: generate_6gram.log
Start time: Sat Sep 6 12:13:12 PM +07 2025
=====
```

```
=====
Running: Default Generation (5 sentences)
=====
```

1. ကျွန်တော် သည် လည်း၊ ထို မိန့်မ သည် ဒုစရိုက် ကို မ ပြု
2. ကျွန်တော် မ နေ နဲ့ တော့ မ လုခြေမှ ကို ကာကွယ် ပေး ရေးရ
3. ကျွန်တော် သည် ။) သည် မဟု ကဲသို့ ပင် ဖြစ် ပြီး ။
4. ကျွန်တော် မ နေ ပါ နှင့် ။ ထို အမူ ထို သည် လည်း
5. ကျွန်တော် သည် သူ တို့၏ ခန္ဓာကိုယ် အရွယ်အစား နှင့် ပန်းပွင့် ကို ဆေးဖော် စပ်

- မြန်မာစာငြေကြောင်း
တော့ဟူတ်တယ်
သံသော် ပံ့မျန်
မဟုတ်ဘူး
- Ngram LM ရဲ့
အခန်းကုလ္ပာက
ကြီးတယ်
- လက်ရုံချိန်ထိုး
သံးနေကုတ္တန်းပဲ
ဥပမာ ASR, TTS

Fig. Text generation with 6-gram LM

Statistical Language Model (Next Word Prediction)

Running: Verbose Generation

Step 1:

- 1. တို့ logP=-5.0752
- 2. က logP=-5.4820
- 3. မ logP=-5.6468
- 4. သည် logP=-5.6537
- 5. မျိုး logP=-5.7508

Step 2:

- 1. တို့ logP=-6.4285
- 2. သည် logP=-6.8439
- 3. တို့၏ logP=-6.9923
- 4. အား logP=-7.1947
- 5. များ logP=-7.2584

Step 3:

- 1. စိတ်နှလံး logP=-8.3649
- 2. အရှင် logP=-8.3659
- 3. အမည် logP=-8.9700
- 4. (..... logP=-9.0386
- 5. အပြုံ logP=-9.0406

Step 4:

- 1. များ logP=-9.5121
- 2. ကို logP=-9.5839
- 3. ငြှေ logP=-10.3921
- 4. ငြောင် logP=-10.6149
- 5. မရှိ logP=-10.7894

Step 5:

- 1. ကို logP=-9.6648
- 2. ငြောင် logP=-10.8011
- 3. အထွက် logP=-10.8488
- 4. မ logP=-10.9633
- 5. မ logP=-11.6035

Step 6:

- 1. ခွင့်လွှတ် logP=-11.6006
- 2. ယေ logP=-12.0958
- 3. ယူကျိုးမရ logP=-12.0986
- 4. လည်း logP=-12.5540
- 5. အသုံးပြု logP=-12.6973

Step 7:

Fig. Text generation with 6-gram LM

Statistical Language Model (Next Word Prediction)

```
=====  
Running: Fixed Random Seed = 42
```

1. ကျွန်တော် မ သိ ဘူး။ ၎င်္ခုဗား ကို ကူညီ နှင့် ပါ လိမ့်
 2. ကျွန်တော် တို့ က လည်း । သင် တို့ သည် ထာဝရ ဘူး၏
 3. ကျွန်တော် သည် လည်း။ ထို နေ့၌ မွေ့ သီချင်း များ နှင့် ဆက်စပ်
 4. ကျွန်တော် သည် လည်း ဘာ မျှား မ သိ ရာ သည် အရာ ဖြစ်
 5. ကျွန်တော် သည်။ (က) မူပိုင်ခွင့် တရားရုံး သည် ယာယို အရေးယူ
- ```
=====
```

```
Running: Temperature = 1.5 (More Randomness)
```

1. ကျွန်တော် က မ အိမ် ကံစ ကား မ ဆက် မ နေ ကြ
  2. ကျွန်တော် မ သိ ခဲ့ ကြ သည် । । । လစ် များ ပြည့်တွင်စစ်
  3. ကျွန်တော် မ သိ။ အဘယ် သို့ နည်း။ အကျွန်းပို့ တို့၏ သား
  4. ကျွန်တော် သည် သူ ကိုယ်တို့ အတွက် ပါ ကောင်းကို။ ချမ်းသာ များ ရရှိ နိုင်
  5. ကျွန်တော် က သူ နဲ့ စကားပြော ရ တာ အဆင်မပြော ဘူး။။။
- ```
=====
```

All experiments completed!

End time: Sat Sep 6 12:15:34 PM +07 2025

Results saved in: generate_6gram.log

- နောက်တစ်ခေါက် generate လုပ်တဲ့ အခါမှာ လုပ်သူး
ထဲကြောင်းတဲ့ ချင်ရင်
Random Seed
သတ်မှတ်တယ်
- ထဲကြောင်းတွေ generate လုပ်တဲ့ အခါမှာ
သယ်လောကထိ
ကျပ်န်းလုပ်မှာ လဲ ဆိတ္တကို
Temperature နဲ့ ညိုတယ်

Fig. Applying “random seed” and “temperature”

Statistical Language Model (Formula)

- N-gram Language Model ဟာ စာကြောင်းတစ်ခုရဲ့ ဖြစ်နိုင်ခြေ (probability) ကို တွက်ချက်ဖို့ အသုံးပြုတဲ့ နည်းလမ်းတစ်ခု ဖြစ်ပါတယ်။ ဒါ Model က စာကြောင်းထဲက စကားလုံးတစ်ခုကို သူရဲ့ အရှေ့က စကားလုံး(များ) ပေါ်မှုတည်ပြီး ခန့်မှန်းလေ့ရှိပါတယ်။
- ဥပမာအားဖြင့် "ဗမာ စကား ပြောတတ် သလား" ဆိုတဲ့ စာကြောင်းတစ်ခုလုံးရဲ့ ဖြစ်နိုင်ခြေကို တွက်ချက်ဖို့ အတွက် စကားလုံးတစ်ခုချင်းစီရဲ့ ဖြစ်နိုင်ခြေကို အောက်ပါ formula နဲ့ တွက်လုံ့ရပါတယ်။

$$P(w_1, w_2, \dots, w_n) = \prod_{i=1}^n P(w_i | w_1, \dots, w_{i-1})$$

Statistical Language Model (Formula)

- အလွယ်ဆုံးနဲ့ အသုံးအများဆုံး နည်းလမ်းတစ်ခုဖြစ်တဲ့ Bigram ($N=2$) formula ကတော့ အောက်ပါအတွက်ဖြစ်ပါတယ်။

$$P(w_i | w_1, \dots, w_{i-1}) \approx P(w_i | w_{i-1})$$

$$P(w_1, w_2, \dots, w_n) \approx \prod_{i=1}^n P(w_i | w_{i-1})$$

Statistical Language Model (Formula)

တင်ကြားကို Bigram အပိုင်းလေးတွေအဖြစ် အရင်ဆုံး
ပိုင်းဖြတ်ရပါမယ်။

- $\langle s \rangle \rightarrow \text{ဗမာ}$
- $\text{ဗမာ} \rightarrow \text{စကား}$
- $\text{စကား} \rightarrow \text{ပြောတတ်}$
- $\text{ပြောတတ်} \rightarrow \text{သလား}$
- $\text{သလား} \rightarrow \langle /s \rangle$

Statistical Language Model (Formula)

$P(\text{ഒരു പദം | അവയാണ് ചേർക്കേണ്ട വാക്കുകൾ})$

$$= P(\text{ഒരു} | \langle s \rangle) \cdot P(\text{അവയാണ്} | \text{ഒരു}) \cdot P(\text{ചേർക്കേണ്ട} | \text{അവയാണ്}) \cdot P(\text{വാക്കുകൾ} | \text{ചേർക്കേണ്ട}) \\ \cdot P(\langle /s \rangle | \text{വാക്കുകൾ})$$

Word Frequency (stop words)

- තාල්පුරිතයාත්තු පොතාවහා
ආမුරාග්‍රීස් රුළුණ Frequency
ලෙග තුන්කුණුණුවෙන්
ආච්චාර්යාතුන් අභ්‍යන්තුවෙන්
ආමුරාග්‍රීස්පි
- උපහා stop words

| | | |
|---------------|-----------------|-----------------|
| 1051817 වැනි | 68971 ඩේ | 31223 ප්‍රේ |
| 681500 ගි | 66515 භාව | 31084 ඩි |
| 405563 ඕං | 62744 ප්‍රියාන් | 30977 යොගි |
| 395579 ප්‍රේ | 62719 ලිංග | 30641 රෙඟි |
| 365441 ග | 62576 මෝදු | 30549 ඇඩි |
| 347853 ගො | 61546 එල | 30512 ගො. |
| 326613 ඇ | 61313 ගා | 29882 අභුද් |
| 304877 ඩි | 60712 ඉංග්‍රීස් | 28337 වාස් |
| 288336 ගුද් | 58890 ඇං | 28270 ප්‍රියාන් |
| 269068 ගු | 58605 ගු | 27958 ගො: |
| 261954 ඇ | 57934 චිං | 27905 ගුණ්තෝරු |
| 259130 වෛව | 57385 එල | 27889 වා |
| 251962 ට | 56922 ගිග්‍රාම | 27620 සිං |
| 247774 ඔ | 56881 ඩී | 27498 ගැං |
| 243837 ඔව | 56766 අංද | 27434 වා |
| 240070 ඔ? | 54809 එදින් | 27285 මහුත් |
| 228722 ප්‍රී | 54672 වාව | 27176 පෙදින් |
| 220866 ගිද් | 54413 ගොජුවා | 26622 ප්‍රියාන් |
| 220817 ගෑ | 54312 ගාව | 26582 ගිංයි |
| 216702 ගිං | 54059 එලයා | 26386 දැංග |
| 213391 ඩුන් | 53860 ගිං | 26018 අභ්‍යන්තර |
| 210427 ඩිතායි | 53500 මැන්දි | 25652 මොගි |
| 203227 ඩුන් | 51894 ගිං | 25630 අභ්‍යන්තර |
| 198766 මුද් | 51519 වද් | 25609 ඩිං |
| 173716 ඝ | 51341 ගිත් | 25330 ගේ |
| 157537 එලුන් | 50841 ගුව | 24756 ගිග්‍රාම් |
| 156259 ගැ | 49493 ප්‍රියාන් | 24692 වාස් |
| 153894 ග | 48294 ග | 24343 වාගින් |
| 152471 ඇ | 48195 ගිං | 24332 ලෙගින් |
| 141021 ගාව | 47939 ප්‍රියාන් | 24201 ගේ |
| 138490 ඩුන් | 47332 ගැත් | 23923 අභ්‍යන්තර |

WordCloud (Visual Display of Word Frequency)

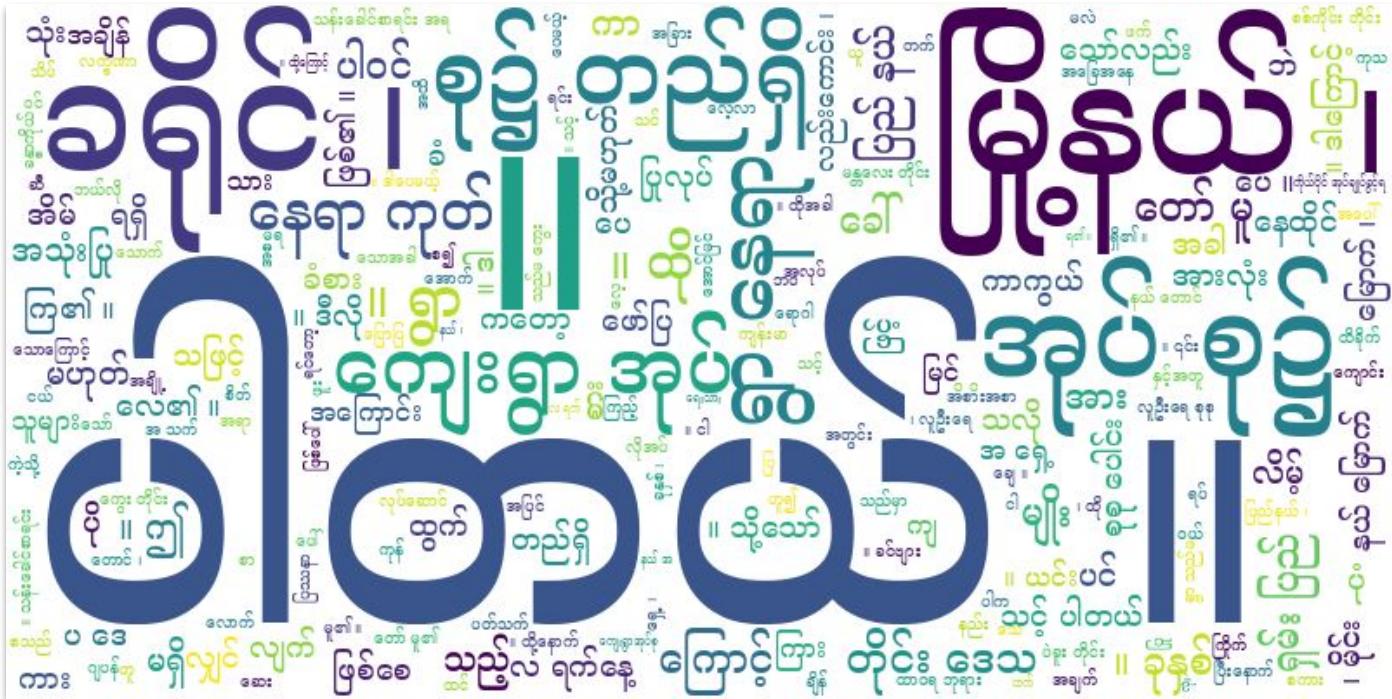


Fig. WordCloud of myMono Corpus version 2.0

WordCloud



Fig. WordCloud of myMono Corpus version 2.0 (removed 100 stopwords + top 100 words)

WordCloud



Fig. WordCloud of myMono Corpus version 2.0 (removed 100 stopwords + top 1000 words)

Embedding Techniques (TF-IDF)

input ဖိုင်ထဲက မြန်မာစာ စာကြောင်းတွေကို syllable ဖြတ်ပြီးတော့ tf-idf တွက်တာကို ဥပမာအနေနဲ့ ရေးပြထားတာပါ။
input ဖိုင်က အောက်ပါအတိုင်း

```
(py3.8.10) ye@ykt-pro:/media/ye/project1/cadt/student/internship/demo/text$ cat eg-corpus.txt  
နေကောင်း တယ် ရှေ့  
အခု ဘာ လုပ် နေ သလဲ  
နေကောင်း အောင် ရေ့ ပါ ရှေ့  
အခု အလုပ် လုပ် နေ တယ်
```

run ကြည့်ရင် အောက်ပါအတိုင်း output ပြပေးလိမ့်မယ်။

```
(py3.8.10) ye@ykt-pro:/media/ye/project1/cadt/student/internship/demo/text$ python ./syl2tf-idf.py ./eg-corpus.txt  
ကောင်း ခု တယ် ရှေ့ ပါ ဘာ လုပ် လဲ သ ဒေ အောင်  
0 0.539313 0.000000 0.539313 0.356966 0.539313 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000  
1 0.000000 0.347852 0.000000 0.230239 0.000000 0.000000 0.441206 0.347852 0.441206 0.441206 0.347852 0.000000  
2 0.378779 0.000000 0.000000 0.501420 0.378779 0.480433 0.000000 0.000000 0.000000 0.000000 0.000000 0.480433  
3 0.000000 0.309520 0.309520 0.204868 0.000000 0.000000 0.000000 0.619041 0.000000 0.000000 0.619041 0.000000
```

Fig. Syllable to TF-IDF conversion demo

Embedding Techniques (Word2Vec)

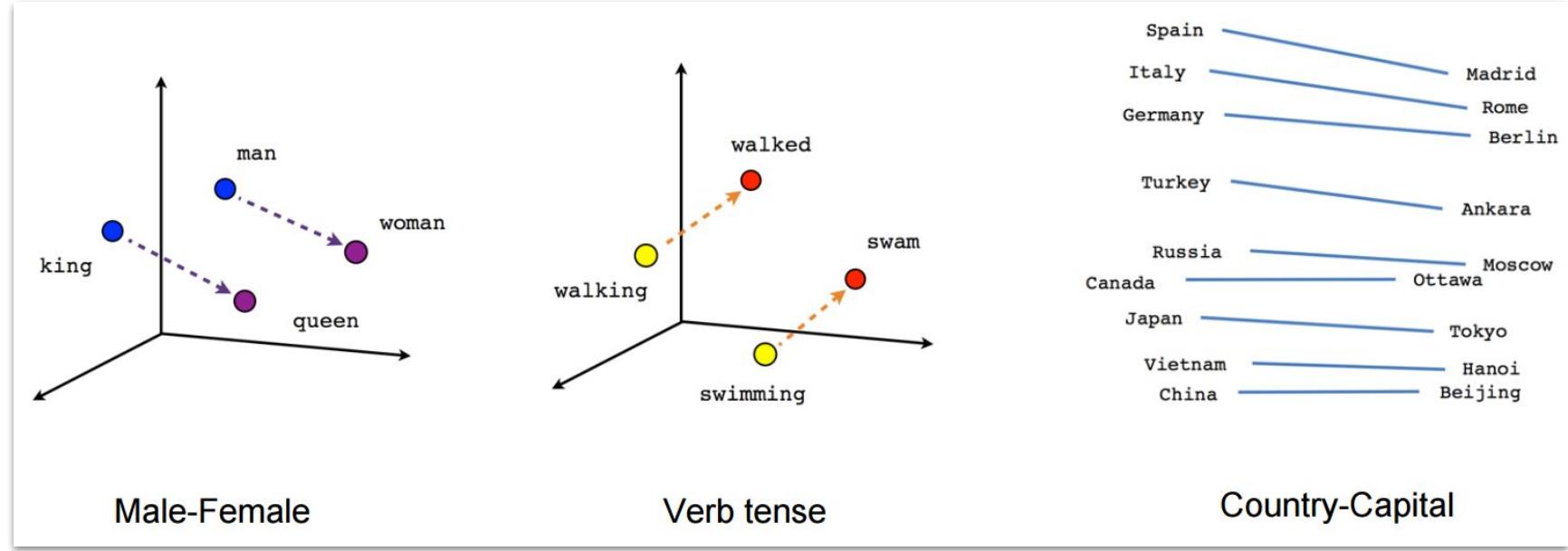


Fig. CBOW versus Skipgram

(Link: <https://www.tensorflow.org/images/linear-relationships.png>)

Embedding Techniques (Word2Vec)

The screenshot shows a red header bar with the arXiv logo and navigation links for 'Search.' and 'Help'. Below the header, the URL 'arXiv > cs > arXiv:1301.3781' is displayed. The main content area has a light gray background. It features the title 'Computer Science > Computation and Language' and a note about submission and revision dates. The full title of the paper is 'Efficient Estimation of Word Representations in Vector Space'. Below the title are several interactive icons: 'CODE', 'Notebook', 'Edit', 'Bookmark', 'Bell', and 'Share'. The authors' names are listed as 'Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean'. A text input field with placeholder 'Ask the author(s) a question! :)' and a blue 'Ask' button are shown. The text of the paper abstract is displayed at the bottom.

arXiv > cs > arXiv:1301.3781

Computer Science > Computation and Language

[Submitted on 16 Jan 2013 (v1), last revised 7 Sep 2013 (this version, v3)]

Efficient Estimation of Word Representations in Vector Space

CODE Notebook Edit Bookmark Bell Share

Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean

Ask the author(s) a question! :)

Ask

powered by CatalyzeX

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

Embedding Techniques (FastText)

The screenshot shows a research paper on the arXiv platform. The title of the paper is "Enriching Word Vectors with Subword Information". It was submitted on July 15, 2016, and last revised on June 19, 2017. The authors are Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. The paper discusses continuous word representations and proposes a new approach based on skipgram models using character n-grams. It claims state-of-the-art performance on word similarity and analogy tasks. The arXiv interface includes a search bar, help links, and various interaction buttons like "Ask".

arXiv > cs > arXiv:1607.04606

Computer Science > Computation and Language

[Submitted on 15 Jul 2016 (v1), last revised 19 Jun 2017 (this version, v2)]

Enriching Word Vectors with Subword Information

CODE Notebook

Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov

Ask the author(s) a question! :)

Ask

powered by CatalyzeX

Continuous word representations, trained on large unlabeled corpora are useful for many natural language processing tasks. Popular models that learn such representations ignore the morphology of words, by assigning a distinct vector to each word. This is a limitation, especially for languages with large vocabularies and many rare words. In this paper, we propose a new approach based on the skipgram model, where each word is represented as a bag of character n -grams. A vector representation is associated to each character n -gram; words being represented as the sum of these representations. Our method is fast, allowing to train models on large corpora quickly and allows us to compute word representations for words that did not appear in the training data. We evaluate our word representations on nine different languages, both on word similarity and analogy tasks. By comparing to recently proposed morphological word representations, we show that our vectors achieve state-of-the-art performance on these tasks.

Embedding Techniques (FastText)

- time python ./oppa_word.py --input
./mymono/myMono_corpus.ver.0.1.shuf.syl \
--dict data/myg2p_mypos_name.dict \
--space-remove-mode my_not_num \
--use-bimm-fallback --bimm-boost 150 \
--output ./mymono/myMono_corpus.ver.0.1.shuf.word

```
(pytorch_py3.10) ye@lst-hpc3090:~/exp/myTokenizer/oppaWord$ time python ./oppa_word.py --input ./mymono/myMono_corpus.ver.0.1.  
shuf.syl --dict data/myg2p_mypos_name.dict --space-remove-mode my_not_num --use-bimm-fallback --bimm-boost 150 --output ./mymo  
no/myMono_corpus.ver.0.1.shuf.word  
  
real    3m38.898s  
user    3m38.264s  
sys     0m0.603s  
(pytorch_py3.10) ye@lst-hpc3090:~/exp/myTokenizer/oppaWord$ -
```

Fig. Segmenting the myMono corpus with oppaWord.

Embedding Techniques (FastText)

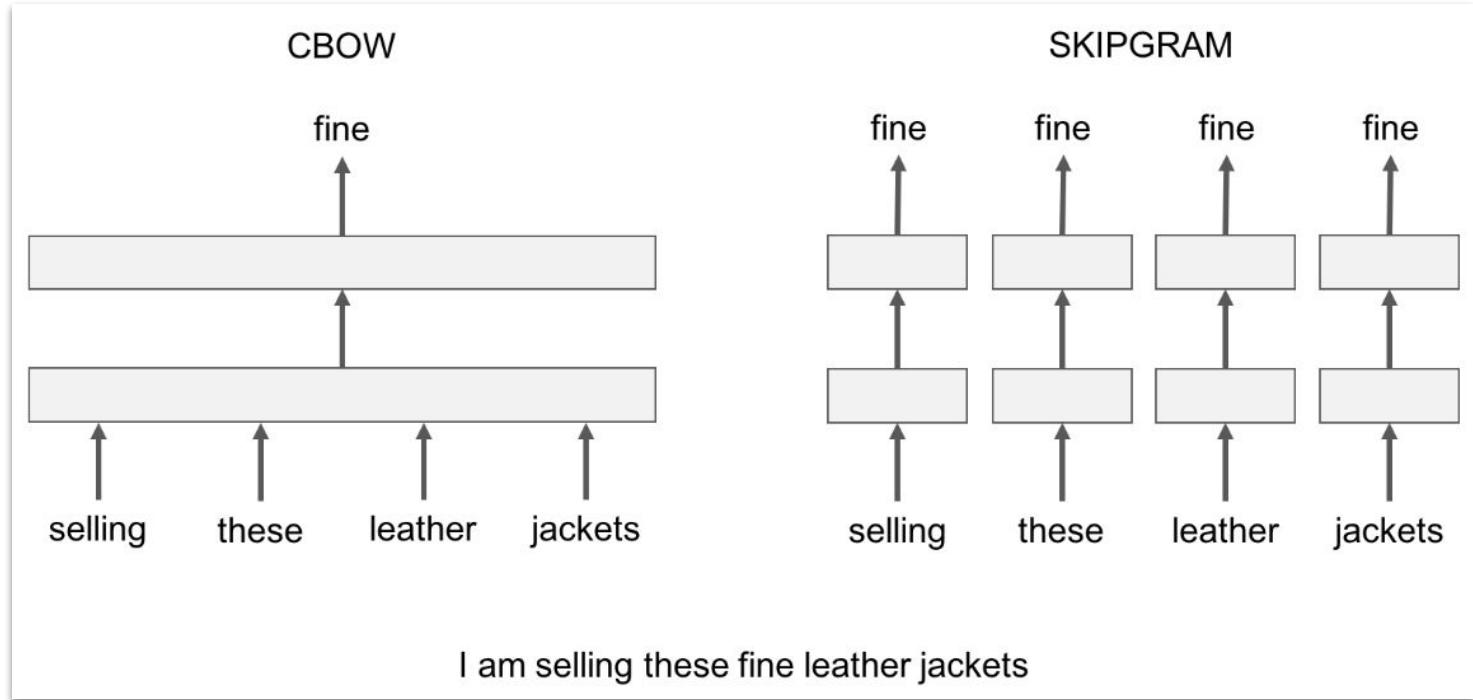


Fig. CBOW versus Skipgram

Embedding Techniques (FastText)

```
ye@lst-hpc3090:~/exp/myMono_embd/v.0.1$ time /home/ye/tool/fastText-0.9.2/fasttext skipgram -input ./myMono_corpus.ver.0.1.shuf.txt.word -output ./fasttext/myMono_ver.01.word
Read 34M words
Number of words: 57084
Number of labels: 0
Progress: 100.0% words/sec/thread: 240960 lr: 0.000000 avg. loss: 1.775276 ETA: 0h 0m 0s
real 1m2.781s
user 11m55.924s
sys 0m1.633s
ye@lst-hpc3090:~/exp/myMono_embd/v.0.1$ -
```

Fig. Building fasttext embedding model with skipgram

```
ye@lst-hpc3090:~/exp/myMono_embd/v.0.1/fasttext$ ls -lh
./myMono_ver.01.word.{vec,bin}
-rw-rw-r-- 1 ye ye 809M Sep 5 16:24 ./myMono_ver.01.word.bin
-rw-rw-r-- 1 ye ye 48M Sep 5 16:24 ./myMono_ver.01.word.vec
```

Embedding Techniques (FastText)

```
ye@lst-hpc3090:~/exp/myMono_embd/v.0.1$ time /home/ye/tool/fastText-0.9.2/fasttext cbow -input ./myMono_corpus.ver.0.1.shuf.txt.word -output ./fasttext/myMono_ver.01.word.cbow
Read 34M words
Number of words: 57084
Number of labels: 0
Progress: 100.0% words/sec/thread: 458794 lr: 0.000000 avg. loss: 1.694909 ETA: 0h 0m 0s
real    0m34.521s
user    6m17.170s
sys     0m1.376s
ye@lst-hpc3090:~/exp/myMono_embd/v.0.1$ -
```

Fig. Building fasttext embedding model with cbow

```
ye@lst-hpc3090:~/exp/myMono_embd/v.0.1/fasttext$ ls -lh
./myMono_ver.01.word.cbow*
-rw-rw-r-- 1 ye ye 809M Sep 5 16:37 ./myMono_ver.01.word.cbow.bin
-rw-rw-r-- 1 ye ye 46M Sep 5 16:37 ./myMono_ver.01.word.cbow.vec
```

Embedding Techniques (Nearest Neighbor Queries)

How is "Closeness" Measured?

"Closeness" is calculated using Cosine Similarity, a value between -1.0 and 1.0 that measures the angle between two vectors.

- A score close to 1.0 means the words are highly similar.
- A score close to 0.0 means there is little to no similarity.
- The scores you see are the cosine similarity values.

Embedding Techniques (Nearest Neighbor Queries)

$$\text{cosine similarity} = S_C(\mathbf{A}, \mathbf{B}) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \cdot \sqrt{\sum_{i=1}^n B_i^2}},$$

where A_i and B_i are the i th components of vectors \mathbf{A} and \mathbf{B} , respectively.

Source: https://en.wikipedia.org/wiki/Cosine_similarity

Embedding Techniques (Nearest Neighbor Queries)

Query word? ရန်ကုန်

မဂ္ဂလာဒံ 0.791902

တက္ကသံလိပ်သာ 0.777902

မြောက်ဥက္ကလာပ 0.764467

ညောင်တုန်း 0.764362

ကမာရွတ် 0.761159

ကြည့်မြင်တိုင် 0.760447

ပျော်မနား 0.756595

ဗလတထောင်ဘုရား 0.753197

ဗလတထောင် 0.751135

ပါသိမ် 0.740126

Query word? ရန်ကုန်

မြောက်ဥက္ကလာပ 0.64371

မဂ္ဂလာဒံ 0.630375

မြောက်အဂံ 0.627339

ရန်ကင်း 0.620133

ပါသိမ် 0.609955

ကမာရွတ် 0.601447

ကြည့်မြင်တိုင် 0.597066

မန္တလေး 0.594714

ကုစ္စီးခြေကုန်း 0.591976

ဒဂု 0.587514

Fig. Nearest neighbor queries with word vectors (left: 100, right: 300 dimensions)

Embedding Techniques (Nearest Neighbor Queries)

Query word? သရက်သီး

ရွှေက်သီး: 0.766829

မင်းကုတ်သီး: 0.766357

ကန်စွဲး: 0.756342

ဆီးသီး: 0.755452

သခ္ဓားမေးသီး: 0.754693

အချဉ်း၏ 0.74806

သဘောသီး: 0.746962

နဂါးမောက်သီး: 0.746099

ကောက်သီး: 0.745531

ပဲစောင်းလူး: 0.742606

Query word? သရက်သီး

ကောက်သီး: 0.675214

ရွှေက်သီး: 0.673202

မင်းကုတ်သီး: 0.666533

တည်သီး: 0.63993

နဂါးမောက်သီး: 0.639581

ကြက်မောက်သီး: 0.630164

အသီး: 0.622755

သစ်သီး: 0.619829

နာနတ်သီး: 0.616811

လိုမွေ့ဗုံးသီး: 0.61212

Fig. Nearest neighbor queries with word vectors (left: 100, right: 300 dimensions)

Embedding Techniques (Nearest Neighbor Queries)

Query word? ရခိုင်မှန်တီ

မှန်တီ 0.844894

သက်နှစ်ထမင်း 0.722245

မန်ဟင်းခါး 0.721326

မန်ဟင်းခါးဟင်းရည် 0.700476

မှန်လလုံးရေပြီ 0.68986

ရခုင် 0.687642

အသပ် 0.680808

ပေါင်မန်ကင် 0.672539

အုန်းဆုံးခေါက်ဆွဲ 0.661636

လမှန် 0.659123

Query word? ရခိုင်မှန်တီ

မှန်တီ 0.790716

ပေါင်မှန်ကင် 0.666917

ရခိုင် 0.638991

ပေါင်မန် 0.635233

မှန်ဟင်းခါးဟင်းရည် 0.616322

မှန်လလုံးရေပြီ 0.613742

ကတ္တမန် 0.60678

မှန်ဟင်းခါး 0.603243

လမှန် 0.586718

အသားညှပ်ပေါင်မှန် 0.585229

Fig. Nearest neighbor queries with word vectors (left: 100, right: 300 dimensions)

Embedding Techniques (Nearest Neighbor Queries)

Query word? သမီးရည်းစား

ယောက်ရည်းစား: 0.828197
ရည်းစားထားထား: 0.824583
ရည်းစား: 0.811393
ယောက်ယောက်မယား: 0.776514
ချုစ်သူ 0.755538
အမှုထောင်သည် 0.754132
ချုစ်သူမျိုး: 0.741586
မြင်မြင်ချုင်း: 0.734387
အမှုထောင်ရက်သားကျိုး 0.732148
အကြောင်လင်မယား: 0.726207

Query word? သမီးရည်းစား

ယောက်ရည်းစား: 0.785964
ရည်းစားထားထား: 0.781809
ရည်းစား: 0.778414
ယောက်ယောက်မယား: 0.635797
အံမှုထောင်သည် 0.586299
သမီးယောက္ခာမ 0.574192
ရည်စား: 0.570556
အမှုထောင်ဖက် 0.57015
ချုစ်သူ 0.569804
လက်စွဲမထပ် 0.569387

Fig. Nearest neighbor queries with word vectors (left: 100, right: 300 dimensions)

Embedding Techniques (Nearest Neighbor Queries)

Query word? ဆရာမ

ကျောင်းအုပ်ဆရာမကြီး: 0.758842
ကျောင်းသု 0.720078
ကျောင်းအုပ်ဆရာကြီး: 0.699432
ပညာပြ 0.69658
ဆရာ 0.673178
ကျောင်းအုပ် 0.670142
ကျောင်းသားကတ် 0.669361
ညကျောင်း 0.662287
သူနာပြ 0.661204
ကျို၍ရင် 0.661188

Query word? ဆရာမ

ကျောင်းအုပ်ဆရာမကြီး: 0.558769
ဆရာ 0.558493
ဆရာလေး: 0.556287
ဆရာ့ 0.505054
ဆရာလပ် 0.504805
ကျောင်းအုပ်ဆရာကြီး: 0.500963
ဆရာကတော် 0.497177
သူနာပြု 0.495417
ကျောင်းသု 0.491647
စာသင်တာ 0.489204

Fig. Nearest neighbor queries with word vectors (left: 100, right: 300 dimensions)

Embedding Techniques (Nearest Neighbor Queries)

Query word? ဆရာဝန်

ဆရာဝန် 0.928535

သားဆရာဝန် 0.857536

ခွစ်တ်ကုဆရာဝန် 0.83778

အထေးကာ 0.790519

ဆရာဝန်ကြီး 0.756213

ဝန်(0.734541

ဝန်/ 0.70739

Laryngoscopy 0.696624

ကုသမ 0.688599

သူနာပြ 0.686658

Query word? ဆရာဝန်

သားဆရာဝန် 0.717381

ခွစ်တ်ကုဆရာဝန် 0.699172

အထေးကာ 0.65095

ဆရာဝန်ကြီး 0.646915

OG 0.64477

ကုသမ 0.625344

တတိုင်ပင် 0.613686

ဆေးအညွှန်း 0.602823

cystoscopy 0.594916

Laryngoscopy 0.594098

Fig. Nearest neighbor queries with word vectors (left: 100, right: 300 dimensions)

Embedding Techniques (Nearest Neighbor Queries)

Query word? အမေရိကန်

ကနေဒါ 0.802885
ပြတိန် 0.764929
ပြည်ထောင်စု 0.75218
ကယ်လီဖိုးနှီးယား 0.720743
ရုရှား 0.716525
မကဆီကို 0.706001
ဆုံး 0.701716
အော်ရှိဂုံန် 0.701043
ပြည်ထောင် 0.699589
ဂျာမျန် 0.697735

Query word? အမေရိကန်

ပြည်ထောင်စု 0.650969
ကနေဒါ 0.649022
အမေရိက 0.608724
ပြည်ထောင် 0.590172
မြဲ့ 0.58434
အမေရိကား 0.570884
ဗာဂျုံးနှီးယား 0.569301
ကန္တာ့ 0.56374
ကယ်လီခိုးနှီးယား 0.559637
အော်ရှိဂုံန် 0.554418

Fig. Nearest neighbor queries with word vectors (left: 100, right: 300 dimensions)

Embedding Techniques (Word Analogies with FastText)

Word analogy is a way to test if a model has learned the relationships between words. The formula $A - B + C$ asks the model to find a word D that has the same relationship to C as B has to A.

The calculation is performed in the vector space, where each word is a point. The formula is:

$$V_D = V_A - V_B + V_C$$

- V_A : Vector for word A
- V_B : Vector for word B
- V_C : Vector for word C
- V_D : The resulting vector, which is numerically closest to the vector of the desired answer word D.

Embedding Techniques (Word Analogies with FastText)

Query triplet (A - B + C)? ရန်ကုန် မြန်မာ ထိုင်း

ဘန်ကောက် 0.58378

ထိုင်းနိုင်း 0.541044

မြောက်ဥက္ကလာပ 0.535116

ထင်ပေါ် 0.530853

ချင်းမှုင် 0.508272

မဂ်လာဒံ 0.494238

တောင်ဥက္ကလာ 0.49088

လိုင်သာယာ 0.479702

ထိုင်းဘုရား 0.467541

သယ်နှံး 0.466632

Fig. (ရန်ကုန် - မြန်မာ + ထိုင်း)?

Embedding Techniques (Word Analogies with FastText)

Query triplet (A - B + C)? စား စားခဲ့ အိပ်ခဲ့

အိပ်ချု 0.663952

အိပ် 0.659455

အိပ်- 0.654276

အိပ်စက် 0.631872

အိပ်ချိန် 0.61765

အိပ်ချင် 0.605428

အိပ်ယာထ 0.564114

အိပ်ပျက် 0.555627

အိပ်၏ 0.551869

အိပ်ယာဝင် 0.541464

Fig. (စား - စားခဲ့ + အိပ်ခဲ့)?

Embedding Techniques (Word Analogies with FastText)

Query triplet (A - B + C)? ကြိုးစား မကြိုးစားဘူး မရှိသေဘူး
ရှိသေ 0.595436
လူရှိသေရှင်ရှိသေ 0.578414
မရှုမသေ 0.566819
မရှုံးမချဖြစ် 0.565208
မထိလေးစား 0.544146
ရှိသေလေးမြတ် 0.543523
မရှုံး 0.538855
ရှိသေကိုင်းညွတ် 0.523898
ကိုးစား 0.49914
နိုင်ထက်စီးနင်း 0.495257

Fig. (ကြိုးစား - မကြိုးစားဘူး + မရှိသေဘူး)?

Embedding Techniques (Word Analogies with FastText)

Query triplet (A - B + C)? လုပ် လုပသော မှန်းသော

မှန်း 0.499901

မှန်း) 0.468084

နပူ။ 0.456985

ရုမန်း 0.430707

နစ်လုပ်ဖယ်ကောင်း 0.42774

မှန်းညီ 0.420947

ပန်းသော 0.413806

စိတ်ပျို့ကိုယ်နဲ့ 0.412342

ပံ့ပန်းသဏ္ဌာန် 0.41114

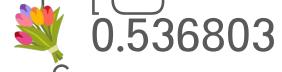
အပြစ်အနာအဆာ 0.407274

Fig. (လုပ် - လုပသော + မှန်းသော)?

Embedding Techniques (Word Analogies with FastText)

Query triplet (A - B + C)? ကိုယ်း ညီလေး ညီမလေး

ကိုကိုကိုး 0.567495



0.536803

ချစ်မမ 0.528624

ကုစိုင်း 0.525522

အားပေး 0.519377

စိုင်းကို 0.518356

ကစ်ကစ် 0.511995

ကစ်ကစ်ချော့ 0.510631

≥ Ö_Ö≤ 0.509732

နေတိုးကိုကို 0.509242

Fig. (ကိုယ်း - ညီလေး + ညီမလေး)?

Embedding Techniques (Word Analogies with FastText)

Query triplet (A - B + C)? အငောက် အဖွဲ့ အဖွဲ့

အဖွဲ့ထိုက် 0.46886

အဖွဲ့အနှစ် 0.445798

အဖွဲ့ထိုက်တန် 0.439098

အဖွဲ့တန် 0.425557

အဖွဲ့အခ 0.402559

အငောက် 0.396132

အရှင်ဖိုး 0.394288

အဖွဲ့ထားနား 0.387705

ဖိုးထိုက် 0.386101

တန်ဖူး 0.3828

Fig. (အငောက် - အဖွဲ့ + အဖွဲ့)?

Embedding Techniques (Word Analogies with FastText)

Query triplet (A - B + C)? ကျောင်းသား ကျောင်း ဘုန်းကြီးကျောင်း
ဘုန်းကြီးကျောင်းသား 0.845826
ဆေးကျောင်းသား 0.717175
နှဲကျောင်းသား 0.712395
သုံးကျောင်းသား 0.666482
ကျောင်းသားကုဒ် 0.607626
အပောင်းသား 0.566575
နွားကျောင်းသား 0.566006
ကျောင်းသားရေးရာ 0.553581
ရွားဦးဘုန်းတော်ကြီးကျောင်း 0.54597
အကောင်းသား 0.537656

Fig. (ကျောင်းသား - ကျောင်း + ဘုန်းကြီးကျောင်း)?

Embedding Techniques (TF-IDF, Word2Vec, FastText)

```
##Testing 3
```

```
Result for tfidf:
```

```
Top 10 similar words to "ကျောင်းသား" are: [ ('ကျောင်းသား', 1.0), ('ကျောင်းသူ', 0.2716536698781178), ('သမဂ္ဂ', 0.08297832320870044), ('ကျောင့်', 0.08245099511953359), ('တလ္လသိုလ်', 0.0771137405996299), ('ဂိုစာဓရီ', 0.07279020382802218), ('ဆေးကျောင်းသား', 0.06600522742497493), ('တွေ', 0.0652168748380101), ('သစ်စံ', 0.06240310265969447), ('အဆိုကျော်', 0.06091473920127301) ]
```

```
Result for word2vec:
```

```
Top 10 similar words to "ကျောင်းသား" are: [ ('ကျောင်းသူ', 0.7797659635543823), ('လူထှု', 0.7149488925933838), ('သင်တန်းသား', 0.6295153498649597), ('မှလတန်း', 0.6149675846099854), ('ဆရာ', 0.6093628406524658), ('ကျောင်းအပ်ပြီး', 0.5891454219818115), ('ကလေးထှု', 0.5886870622634888), ('ကျောင်းဆရာ', 0.5840626955032349), ('သူနာပြု', 0.5738692283630371), ('ဆရာကြီး', 0.5653407573699951) ]
```

```
Result for fasttext:
```

```
Top 10 similar words to "ကျောင်းသား" are: [ ('ကျဲ့ကျောင်းသား', 0.9848452210426331), ('ကျောင်းသား', 0.9800302386283875), ('နွေးကျောင်းသား', 0.9744942784309387), ('ကျောင်းသူ', 0.9632477760314941), ('ကျောင်းသားကြီး', 0.9572359919548035), ('ကျောင်းသား', 0.9550759792327881), ('ဘန်းကြီးကျောင်းသား', 0.9491896033287048), ('ကျောင်းသားလေး', 0.9451152682304382), ('မိကျောင်းသမား', 0.9421031475067139), ('ကျောင်းဆရာ', 0.936782717704773) ]
```

Fig. Similarity measurement with TF-IDF, Word2Vec, FastText

Embedding Techniques (CLIP)

- လက်တွေ့မှာက ပုံတွေကိုလည်း embedding လုပ်ဖို့လိုအပ်တယ်



pexels-goochie-poochie-3361723.jpg



pexels-lina-1741205.jpg



pexels-pixabay-76957.jpg

Embedding Techniques (CLIP)

```
1 [ .....  
2   ..... {  
3     ..... "text": "a photo of a cute puppy",  
4     ..... "image": "./animal/pexels-goochie-poochie-3361723.jpg"  
5   },  
6   ..... {  
7     ..... "text": "a photo of a frog",  
8     ..... "image": "./animal/pexels-pixabay-76957.jpg"  
9   },  
10  ..... {  
11    ..... "text": "a photo of a cat",  
12    ..... "image": "./animal/pexels-lina-1741205.jpg"  
13  },  
14  ..... {  
15    ..... "text": "a photo of a cat",  
16    ..... "image": "./animal/pexels-pixabay-76957.jpg"  
17  },  
18  ..... {  
19    ..... "text": "a photo of a cute puppy",  
20    ..... "image": "./animal/pexels-pixabay-76957.jpg"  
21  }]  
22 ]
```

- Json ဖိုင်မှာ
အောက်ဆုံးက
ပုနစ်ပုက ဖူးပုံ
- တမ်တကာ
လွှပြီး
ထည့်သွေး

Embedding Techniques (CLIP)

```
Using GPU for acceleration.
```

```
--- Processing: 'a photo of a cute puppy' vs './animal/pexels-goochie-poochie-3361723.jpg' ---
```

```
Cosine Similarity Score: 0.2485
```

```
--- Processing: 'a photo of a frog' vs './animal/pexels-pixabay-76957.jpg' ---
```

```
Cosine Similarity Score: 0.2864
```

```
--- Processing: 'a photo of a cat' vs './animal/pexels-lina-1741205.jpg' ---
```

```
Cosine Similarity Score: 0.2840
```

```
--- Processing: 'a photo of a cat' vs './animal/pexels-pixabay-76957.jpg' ---
```

```
Cosine Similarity Score: 0.1979
```

```
--- Processing: 'a photo of a cute puppy' vs './animal/pexels-pixabay-76957.jpg' ---
```

```
Cosine Similarity Score: 0.2062
```

```
real    0m7.589s
```

```
user    0m7.715s
```

```
sys     0m2.706s
```

```
(pytorch_py3.10) ye@lst-hpc3090:~/exp/txt-img-embd$
```

Fig. Similarity measurement between input sentence and image

Embedding Techniques (CLIP)

The screenshot shows a red header bar with the arXiv logo and navigation links for 'Search...' and 'Help'. Below the header, the page title is 'Computer Science > Computer Vision and Pattern Recognition'. A note indicates it was submitted on '26 Feb 2021'. The main title of the paper is 'Learning Transferable Visual Models From Natural Language Supervision'. Below the title are several interactive icons: 'CODE', 'Notebook', 'Share', 'Bookmark', 'Bell', and 'Print'. The authors listed are Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever. There is a text input field for asking questions and a blue 'Ask' button. A note at the bottom states: 'powered by CatalyzeX'. The abstract text discusses the limitations of traditional computer vision supervision and how learning from raw text can overcome these, demonstrating its effectiveness across various downstream tasks.

arXiv > cs > arXiv:2103.00020

Computer Science > Computer Vision and Pattern Recognition

[Submitted on 26 Feb 2021]

Learning Transferable Visual Models From Natural Language Supervision

CODE Notebook

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever

Ask the author(s) a question! :)

Ask

powered by CatalyzeX

State-of-the-art computer vision systems are trained to predict a fixed set of predetermined object categories. This restricted form of supervision limits their generality and usability since additional labeled data is needed to specify any other visual concept. Learning directly from raw text about images is a promising alternative which leverages a much broader source of supervision. We demonstrate that the simple pre-training task of predicting which caption goes with which image is an efficient and scalable way to learn SOTA image representations from scratch on a dataset of 400 million (image, text) pairs collected from the internet. After pre-training, natural language is used to reference learned visual concepts (or describe new ones) enabling zero-shot transfer of the model to downstream tasks. We study the performance of this approach by benchmarking on over 30 different existing computer vision datasets, spanning tasks such as OCR, action recognition in videos, geo-localization, and many types of fine-grained object classification. The model transfers non-trivially to most tasks and is often competitive with a fully supervised baseline without the need for any dataset specific training. For instance, we match the accuracy of the original ResNet-50 on ImageNet zero-shot without needing to use any of the 1.28 million training examples it was trained on. We release our code and pre-trained model weights at [this https URL](https://https://).

Embedding Techniques (EmbeddingGemma)

| Model* | Size | Mean (Task) | MTEB (Multilingual, v2) | | |
|-------------------------------------|------|-------------|-------------------------|----------------|------------|
| | | | Retrieval | Classification | Clustering |
| EmbeddingGemma | 308M | 61.15 | 62.49 | 60.90 | 51.17 |
| granite-embedding-278m-multilingual | 278M | 53.74 | 52.20 | 54.09 | 41.41 |
| gte-multilingual-base | 305M | 58.24 | 56.50 | 57.17 | 44.33 |
| multilingual-e5-large | 560M | 58.55 | 54.08 | 59.43 | 41.70 |
| bge-m3 | 568M | 59.56 | 54.60 | 60.35 | 40.88 |
| jina-embeddings-v3 | 572M | 58.37 | 55.76 | 58.77 | 45.65 |
| Qwen-Embedding-0.6B | 595M | 64.34 | 64.65 | 66.83 | 52.33 |

*GENERAL-PURPOSE OPEN EMBEDDING MODELS

- EmbeddingGemma is a 300M parameter embedding model from Google.

Fig. Benchmark of EmbeddingGemma

Source: <https://ollama.com/library/embeddinggemma>

Transformer (Attention is All you Need)

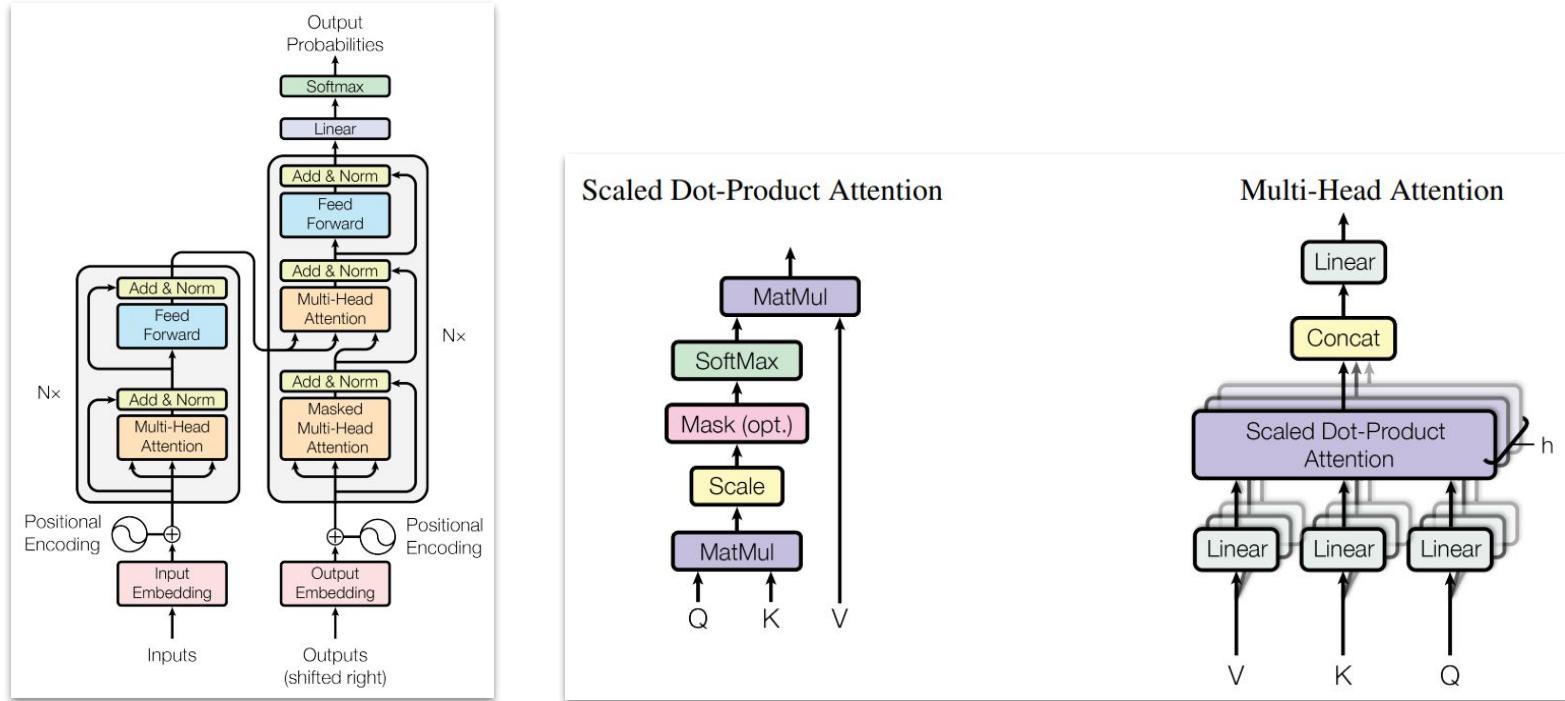
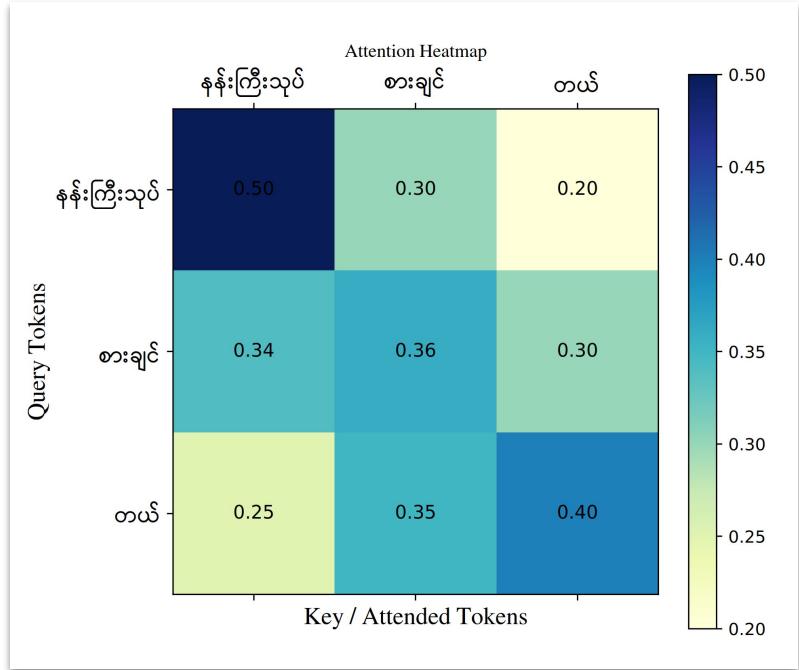


Fig. Transformer architecture (Ashish Vaswani et al., 2017)

Transformer (Positional Encoding)

- Transformer ဟာ စကားလုံးတွေကို တစ်ပြိုင်နက်တည်းကြသုပ္ပါး လုပ်ဆောင်တာကြောင် စကားလုံးတွေရဲ့ အစအစဉ်ကုတ် မသိနိုင်ပါဘူး။ ဒုပ္ပသာနာကုတ်ဖြေရှင်းဖွုံအတွက် နေရာပြကုတ်သွေးခြင်း (Positional Encoding) ကို အသုံးပြုပါတယ်။ ဥပမာ "နန်းကြီးသုပ် စားချုပ် တယ်" ဆိုတဲ့ တာကြောင်းမှာ
- နန်းကြီးသုပ် - နံပါတ် (၁)
- စားချုပ် - နံပါတ် (၂)
- တယ် - နံပါတ် (၃)

Transformer (Scale Dot Product Attention)



- Rows = Query tokens
→ the token “asking” for context
- Columns = Key tokens
→ the tokens being “looked at”
- Color intensity = how much attention one token gives to another

Fig. Attention weights between tokens in “နှစ်းကြီးသုပ် စားချင် တယ်”. (example calculation)

Transformer (Scale Dot Product Attention)

- The heatmap directly visualizes the scaled dot-product attention results.

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^\top}{\sqrt{d_k}} \right) V$$

- only visualized the softmax scores:

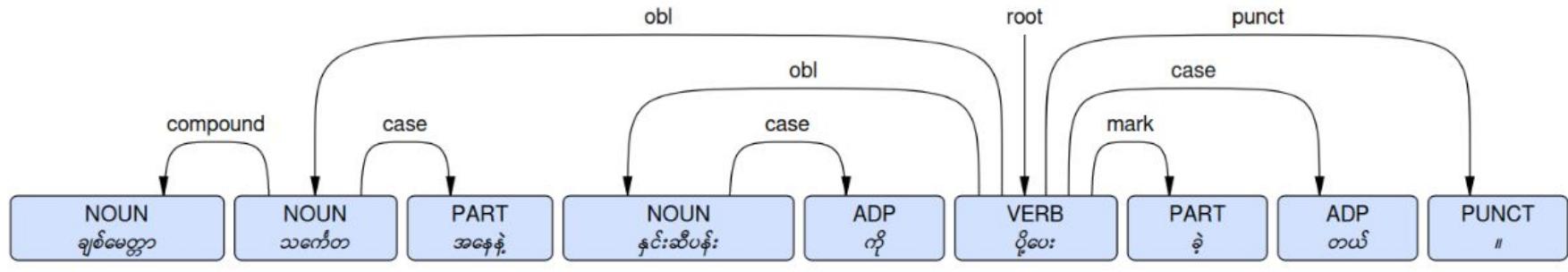
$$\text{weights} = \text{softmax} \left(\frac{QK^\top}{\sqrt{d_k}} \right)$$

Transformer (Multi-head Attention)

Multi-head attention = multiple scaled dot-product attentions in parallel.

- If your model has 8 heads, there are 8 different heatmaps.
- Each head learns different relationships:
 - Head 1 → subject ↔ verb (e.g. "နှစ်းကြီးသုပ်" → "စားချင်")
 - Head 2 → verb ↔ tense ("စားချင်" → "တယ်")
 - Head 3 → semantic similarity ("နှစ်းကြီးသုပ်" & "စားချင်" both food-related) ...and so on.
- After computing all heads, Transformer concatenates their outputs and applies a final linear projection.

LLMs (myUDTree)



- တကယ်တမ်းက စာကြောင်း ထဲမှပါဝင်တဲ့ စာလုံးတွေ တစ်လုံးနဲ့ တစ်လုံး ဘယ်လို့ ဆက်စပ်နေသလဲ ဆိုတာကို ကွန်ပျူးတာက နားလည်ဖို့ဆိုတာ လွှယ်ကူတဲ့ အလုပ် မဟုတ်ပါ
- အောင်လို့ (ဒေါက်တာတန်း KMITL., Thailand) နဲ့ အတူ လုပ်ခဲ့တဲ့ myUDTree

GitHub Link: <https://github.com/ye-kyaw-thu/myUDTree>

Transformer

- Transformer architecture ကို အသေးစိတ်
သံချွင်ရှင်တော့ Attention is All you Need ဆိုတဲ့
စွာတုမ်းကို အချိန်ယူဖတ်ကြည့်ပါ
- ကုယ်တွင် transformer model အသေးလေးကို coding
လုပ်ကြည့်ပါ
- ရှုပြုသား Transformer model ကို အမျိုးမျိုးကို
စမ်းသုံးကြည့်ပါ
- Neural Network အခြေခံလည်းသိဖို့ လုအပ်ပါတယ်
- ထဲထဲဝင်ဝင် သိဖို့က သချာတချို့လည်း နားလည်ဖို့
လုအပ်ပါတယ်

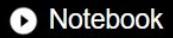
Transformer (Attention is All you Need)

arXiv > cs > arXiv:1706.03762

Computer Science > Computation and Language

[Submitted on 12 Jun 2017 (v1), last revised 2 Aug 2023 (this version, v7)]

Attention Is All You Need

 CODE  Notebook     

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin

Ask the author(s) a question! :)

 Ask

powered by [CatalyzeX](#)

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder-decoder configuration. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

LLMs (GPT-2, myPoetry)

myPoetry (Myanmar Poetry Corpus)

myPoetry corpus is especially suited to applications in creative computational Burmese poetic text generation.

If you are lucky, You might get some interesting poetry output with current myPoetry corpus as follows:

3 random sentences that generated with SRILM 5-gram language model

```
$ ngram -lm ./mypoetry-5gram.lm -gen 10 -lm ./mypoetry-5gram.lm -gen 3  
သူ တော် စင်  
ဘ ဝ ဘိ  
က ဗျာ စာ ရေး ပြန်
```

Generated poetic sentences of GPT-2 model is as follows:

```
ဟုတ်တဲ့ ခေါ်  
အာလယ်ဂွင်းထဲကနေ  
ကြားကြားရပေါ့ ...  
ဘယ်ကာလပဲ လိုက်တယ် ။
```

Latest Version: [Version 1.0](#) (Released Date: 20 April 2023)

- ခိုင်ဆွဲခေါ်
(လက်ရှိမှာ Akita
Univ, Japan) နဲ့
အထာ လုပ်ခဲ့တဲ့
အလုပ်ပါ
- မြန်မာကဗျာတွေ
ကို GPT-2 model
နဲ့ generate
လုပ်တဲ့
experiment

GitHub Link: <https://github.com/ye-kyaw-thu/myPoetry>

LLMs



- various dimensions of Large Language Models (LLMs)

source: <https://arxiv.org/pdf/2408.13296>

R&D with LLMs (myContradict)

1. Negation:

- Example: ခဲ့သော်ပါ။ → မဲ့ပါ။ (Translation: *It is okay.* → *It is not okay.*)
- In negation, the verb is modified with a negative particle (မဲ့), and post-positional markers are adjusted.

2. Antonyms:

- Example: ခဲ့သော်ဖို့လိုက်တော်မူ။ → ခဲ့သော်အနှစ်မူ။ (Translation: *Fat person is our father.* → *Our father is thin.*)
- Here, the verb ခဲ့သော် (fat) is replaced with its antonym အနှစ် (thin).

3. Counterpart:

- Example: မန္တလေး၏သို့။ → မန္တလေး၏မှု။ (Translation: *He goes to Mandalay.* → *He comes from Mandalay.*)
- This involves changing post-positional markers and verbs to indicate temporal or directional opposites.

GitHub Link: <https://github.com/ye-kyaw-thu/myContradict>

R&D with LLMs (myContradict)

Dataset Statistics

| Split | No. of Syllables | No. of Words | No. of Sentences |
|-------|-------------------------------------|-------------------------------------|---------------------------------|
| Train | 150,986 (Source) / 151,007 (Target) | 112,917 (Source) / 112,900 (Target) | 9,702 (Source) / 9,702 (Target) |
| Valid | 18,613 (Source) / 18,622 (Target) | 14,050 (Source) / 14,043 (Target) | 1,214 (Source) / 1,214 (Target) |
| Test | 19,418 (Source) / 19,392 (Target) | 14,498 (Source) / 14,499 (Target) | 1,214 (Source) / 1,214 (Target) |

- အိမ်တုန္ထယ် (KMITL)၊ သူရေအောင် (KMITL) တို့ LU Lab မှာ ဂျွဲက internship သုံးလ ဆင်စဉ်မှာ အတူတူလုပ်ခဲ့ကြတဲ့ အလုပ်ပါ

R&D with LLMs (myParaphrase)

Some examples of tagged paraphrase sentences are as follows:

```
(base) yekyaw.thu@gpu:~/exp/siamese/myParaphrase/corpus/ver1.0/csv-qqp$ shuf train-qqp.csv | head  
"19830","19831","19832","ဒီလို မကြာမကြာ အတူ စား ကြရအောင် ။","ငါတို့ မင်း ကို ကျေးဇူးတင် တယ် ။","0"  
"24566","24567","24568","မင်း ဘယ် ရထား စီးလာ မှာလဲ ။","ကျွန်မက သက်သက်လွတ်သမားရှင် ။","0"  
"28755","28756","28757","ရေအေးအေးလေး က အမောပြ စေ တယ် ။","ရေအေးအေးလေး က အမော ကို ပြ သွား တာ ပဲ  
"23088","23089","23090","ဘယ်သူ့ ကို သံသယဖြစ် တာ လဲ ။","ကဗျာ ရေး သလား ။","0"  
"28697","28698","28699","ရေချိုးခန်း လည်း ပါ သည် ။","နည်းလမ်း ရှာ တာ လား ။","0"  
"14700","14701","14702","တံခါးပေါက် ဆီ အလုအယက် ထွက် နေ တဲ့ လူ အပ် က ကျွန်တော့ ကို နင်း မိ တော့ မ လို ဖြစ် သွား  
"16027","16028","16029","တော် ပါ တယ် ကြိုးစား ပါ","တော် လိုက် တာ ကြိုးစား နော် အားမလျှော့ နဲ့","1"  
"22766","22767","22768","ဘယ် လို ပဲ ဒုက္ခ တွေ အနည်းနည်းအဖို့ဖူ့ လာ ပါစေ သတို့ရှိ ပါ ။","ဘယ် လို ပဲ ပျော် စရာ တွေ  
"26023","26024","26025","မနက်ဖြန် နောင်း ပွဲ ဆို ရင် ကော ဘယ်လို လဲ ။","မနက်ဖြန် နောင်း ပွဲ ကြည့် ကြ မယ် ။","0"  
"24002","24003","24004","မင်း ကို သူ လက်စားချေ မှာ မဟုတ်ပါ ဘူး ။","သူမ အဲဒါ ကို ထပ် စဉ်းစား ခဲ့ ပါ တယ် ။","0"
```

GitHub Link: <https://github.com/ye-kyaw-thu/myParaphrase>

R&D with LLMs (myParaphrase)

- ဒီ corpus က paraphrase လိုခေါ်တဲ့ စကားလုံး မတူတာတောက်
သုံးထားပေမဲ့ စာကြောင်း တစ်ကြောင်းလုံးအနေနဲ့က အဓိပါယ်အားဖြင့်
တဲ့တယ်၊ မတဘား ဆုတာကို ကွန်ပျူးတာက ခဲ့ခြား သံနှင့်တဲ့ မောဒယကု
စမ်းဆောက်ကြည့်ဖို့အတွက် အသုံးပြုဖို့ ရည်ရွယ်ပြီး ဆောက်ခဲ့တဲ့ corpus
တစ်ခုပါ။ မြန်မာစာ NLP သုတေသန အလုပ်အတွက်
အသုံးဝင်ပါလမှုမယ်။ ကျွန်တော်နဲ့ ကျွန်တော့ Ph.D. ကျောင်းသူ
မမြင်မြင်ငွေး တိုက ပြန်ကျော် အချိန်ယူ ပြင်ဆင်ထားခဲ့ကြတော်ပါ။
စာကြောင်းရေ စုစုပေါင်း လေးသောင်းလေးရာကျော် ရုပါတယ်။
open-test data အနေနဲ့လည်း သပ်သပ် စာကြောင်းရေ တစ်ထောင်ကို
ပြင်ဆင်ခဲ့ကြပါတယ်။
- ခုချွန်မှာတော့ LLM တွေကို သုံးပြီး synthetic parallel data တွေလည်း
ထုတဲ့တာမျိုး လုပ်လို့ ရုပါတယ်။ လူကတော့ ပြန်စစ်ပို့ လုံအပ်လိမ့်မယ်။

GitHub Link: <https://github.com/ye-kyaw-thu/myParaphrase>

R&D with LLMs (Testing LLM Intelligence)

To assess the intelligence of large language models (LLMs), I conducted experiments focusing on two key dimensions:

1. Multimodal reasoning testing: ChatGPT-4o's ability to interpret and explain the meaning behind a cartoon image, and
2. Cross-linguistic comprehension evaluating: Gemini 1.5 Pro's understanding of Myanmar-language proverbs by asking it to explain them in both English and Myanmar.

These scenarios examine how well LLMs handle visual abstraction and cultural-linguistic nuance, revealing strengths and limitations in their intelligence.

R&D with LLMs, Testing LLM Intelligence (Gemini 2.5 Flash with Myanmar Proverbs)

4. ကြွေတစ်လက်၊ ကြက်တစ်ခုနှင့် (Kywe Tat Let, Kyat Ta Khun)

English Explanation:

This proverb translates to "One game of chance, one cockfight." It refers to risky or speculative ventures where one might either gain a lot or lose a lot in a single attempt. It suggests a high-stakes situation with unpredictable outcomes, often associated with gambling or bold decisions.

Myanmar Explanation:

ကံစမ်းမဲများ၊ လောင်းကစားများကဲ့သို့ တစ်ကြိမ်တည်းဖြင့် ကြီးမားသော အကျိုးအမြတ်ရနိုင်သလို၊ အားလုံးကို ဆုံးရှုံးသွားနိုင်သည့် အခြေအနေမျိုးကို ရည်ညွှန်းသည်။ ကံကိုသာ အားကိုပြီး အရှုံးအမြတ် ကြီးကြီးမားမား ဖြစ်နိုင်သည့် အခြေအနေများကို ဖော်ပြသည်။

R&D with LLMs, Testing LLM Intelligence (Gemini 2.5 Flash with labeled cartoon)



- Role: You are a professional cartoon caption writer and multilingual visual content analyst.
- Task: You are given a cartoon image. Generate two outputs in four languages (English, Japanese, Chinese, Myanmar):
 1. Caption: A short, witty, and creative caption (≤ 12 words).
 2. Description: A concise, neutral summary (2 to 3 sentences) explaining what is happening in the cartoon.

R&D with LLMs, Testing LLM Intelligence (Gemini 2.5 Flash with labeled cartoon)

The screenshot shows the Gemini 2.5 Flash interface. At the top, there's a blue star icon and a 'JSON' button. Below that is a code editor window displaying the following JSON object:

```
{  
  "caption": {  
    "en": "The perfect solution for a tangled problem.",  
    "ja": "絡まった問題を解決する完璧な方法。",  
    "zh": "解决缠绕问题的完美方案。",  
    "my": "ရှင်တွေးနေတဲ့ ပြဿနာအတွက် အကောင်းဆုံးဖြစ်လို့မယ်။"  
  },  
  "description": {  
    "en": "A cartoon depicts a person on a boat struggling with tangled ropes. A dolphin is swimming below the boat, ready to help untangle the ropes.",  
    "ja": "ボートに乗った人が絡まったロープに苦戦している様子が描かれています。ボートの下をイルカが泳いでおり、ロープを解く手助けをする準備をしている様子です。",  
    "zh": "漫画中描绘了一个人在船上与缠绕的绳索搏斗。一只海豚在船下游泳，正准备解开绳索以解决这个问题。",  
    "my": "လျှော်ကလူတစ်ယောက်က ရှင်တွေးနေတဲ့ ဤတွေကို ဖြဖိုရန်းကန်နေပုံကို ကာတွန်းထဲမှာ ဖော်ပြထားပါတယ်။"  
  }  
}
```

Below the code editor is a text input field with the placeholder 'Ask Gemini'. At the bottom, there are several navigation icons: a plus sign (+), 'Deep Research' (with a magnifying glass icon), 'Image' (with a camera icon), 'Canvas' (with a paint palette icon), 'Guided Learning' (with a book icon), and a microphone icon.

Summary

- အဓိက ပြောခဲ့တော့တွေက LLM နာမည်ကိုယ်တိုင်ကလည်း ဖော်ပြနေသလို ဘာသာစကားနဲ့ ပတ်သက်နေတဲ့ အချက်
- NLP သမားတွေနှစ်ပေါင်းများစွာ လုပ်လာခဲ့တဲ့ statistical language model ဖြစ်တဲ့ word prediction, sentence generation ဆိုတဲ့ အခြေခံအပိုင်းတွေက အရေးကြီးကြောင်း
- ပေးလိုက်တဲ့ prompt ကိုလည်း tokenization (or) word segmentation လုပ်ရကြောင်း
- Embedding ဆိုတာကလည်း မသိမဖြစ်ကြောင်း
- NLP, ML, DeepLearning, LLM အားလုံးက ဒေတာကို heavy ပုံနေတာများတယ်။ အထူးသဖြင့် Neural Network, DeepLearning နဲ့သွားရင်

Summary

- R&D အနေနဲ့က လက်ရှုမှာ LLM ကို မြှုပြုး လုပ်လို့ရတာတွေ အများကြီးပါပဲ
- Corpus Building အတွက်လည်း LLM ကို သုံးလုံးရတယ်။ ဥပမာ အရင်ဆုံး LLM နဲ့ annotation/labeling လုပ်ပြုးမှ လူက ပြန်စစ်တာမျှေး
- အသံဒေတာ၊ ပုံဒေတာတွေလည်း generate လုပ်လို့ရတာမှာလို့ multimodal အပိုင်းတွေကို အရင်ကထက် ပုံပြုး လုပ်လို့ လွယ်လာပြီ
- မြန်မာစာအတွက်က လုပ်စရာတွေအများကြီးထဲမှာမှ ဒေတာပြင်တာကို မပျော်ကြပါနဲ့။ ဒေတာမရှုရင် မြန်မာစာက ဆင်းရဲသား ဘာသာစကားလို့တောင် ပြောရမလားပဲ
- နောက်တစ်ခု NLP နဲ့ AI က ခွဲလို့မရပါဘူး။ NLP foundation ကောင်းရင် Advanced AI, AGI အပိုင်းတွေလည်း ဆက်လေ့လာသွားလို့ ရပါလိမ့်မယ်။

References

1. Example images for embedding demo: <https://www.pexels.com/>
2. CLIP: Connecting text and images: <https://openai.com/index/clip/>
3. Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever, Learning Transferable Visual Models From Natural Language Supervision: <https://arxiv.org/abs/2103.00020>
4. CLIP Github: <https://github.com/openai/CLIP>
5. myStopWord: <https://github.com/ye-kyaw-thu/myStopword>
6. Mikolov, Tomáš; Karafiát, Martin; Burget, Lukáš; Černocký, Jan; Khudanpur, Sanjeev (26 September 2010). "Recurrent neural network based language model". Interspeech 2010. ISCA: ISCA. pp. 1045–1048. Link: https://www.isca-archive.org/interspeech_2010/mikolov10_interspeech.pdf

References

7. Word2vec linear relationship figure:
<https://www.tensorflow.org/images/linear-relationships.png>
8. Fasttext Github: <https://github.com/facebookresearch/fastText>
9. Cosine similarity Wiki page: https://en.wikipedia.org/wiki/Cosine_similarity
10. KenLM GitHub: <https://github.com/kpu/kenlm>
11. Venkatesh Balavadhani Parthasarathy, Ahtsham Zafar, Aafaq Khan, and Arsalan Shahid, The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities (Version 1.1),
<https://arxiv.org/pdf/2408.13296>
12. myParaphrase: <https://github.com/ye-kyaw-thu/myParaphrase>

References

13. myContradict: <https://github.com/ye-kyaw-thu/myContradict>
14. oppaWord Myanar Word Segmenter:
<https://github.com/ye-kyaw-thu/oppaWord>
15. <https://lmstudio.ai/docs/python/embedding>
16. <https://ollama.com/library/embeddinggemma>
17. <https://fasttext.cc/docs/en/unsupervised-tutorial.html>