

Bonjour  
mademoiselle!

Wie geht  
es dir?

Wǒ  
hěn  
hǎo.

Dhanyavaad!

# How Multilingual is Your NLP Model?

Shreya Khurana

Data Scientist, GoDaddy



# whoami

- Data Scientist at  GoDaddy™
- MS Statistics from UIUC
- Work with language data and ML models
- Also worked on Bayesian Hierarchical Modeling

Follow this presentation:



# Outline



Introduction to Multilingual Data

Code-switched data and transliteration

Language Identification

Challenges

Transformer

BERT

BERT Masked LM

Evaluation Metrics

# Multilingual Data



# Languages on the Internet

72.1%

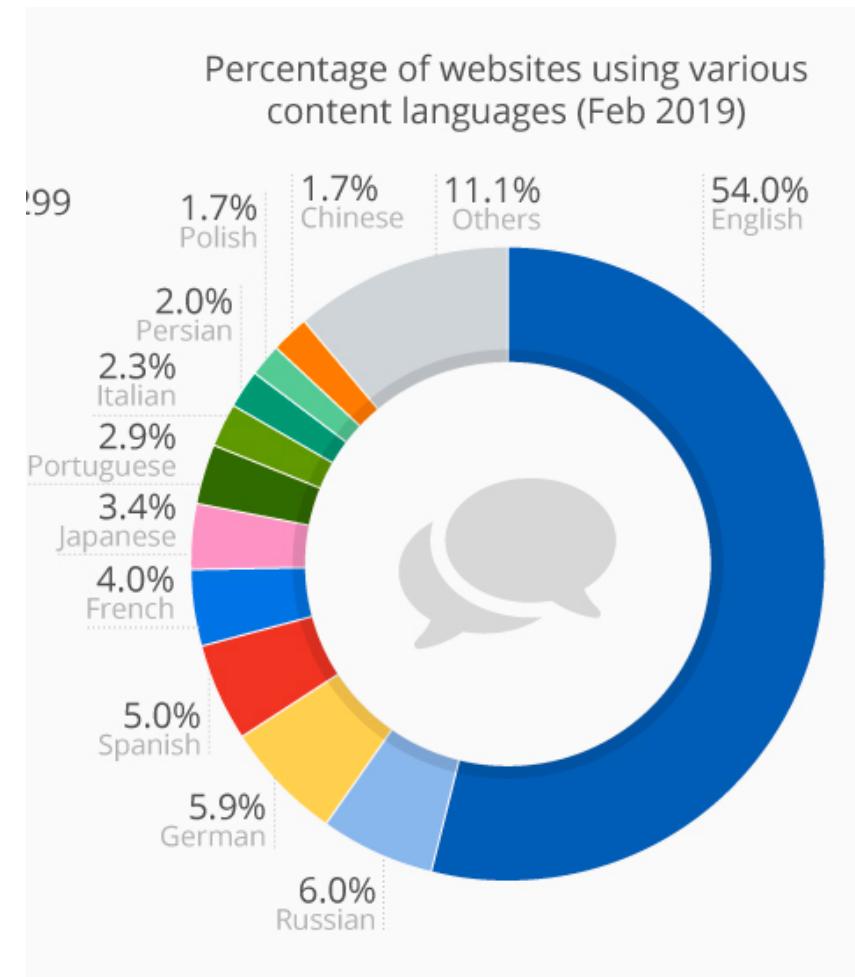
E-commerce consumers spend most or all of their time on websites in their own language.

72.4%

Consumers said they would be more likely to buy a product with information in their own language.

56.2%

Consumers said that the ability to obtain information in their own language is more important than price.



# Language Systems

Tokenization

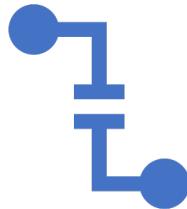
POS  
Tagging

Named Entity  
Recognition

Sequence  
classification

Machine  
Translation

Language  
Modeling



# Code-Switching

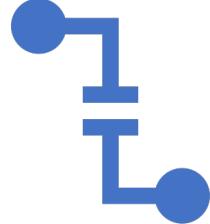
Alternating between two languages

*This morning I hantar my baby tu dekat babysitter tu lah.*

'This morning I took my baby to the babysitter.'  
(Malay)

*Sometimes I'll start a sentence in English y termino en español.*

'and finish it in Spanish.'



# Code-Switching

Alternating between two languages

November 9, 2017 · ▾

Chalo dollars bhejo is khushi me... let me celebrate...

मेरे घर आयी मेरी नन्ही परी... still remember those happy moments ... happy birthday [REDACTED]. love you Mera bachcha ❤️



**Richa Gandhi**

@beingtweet

When your Mom says "Aadha apne bhai/behen ko dena"

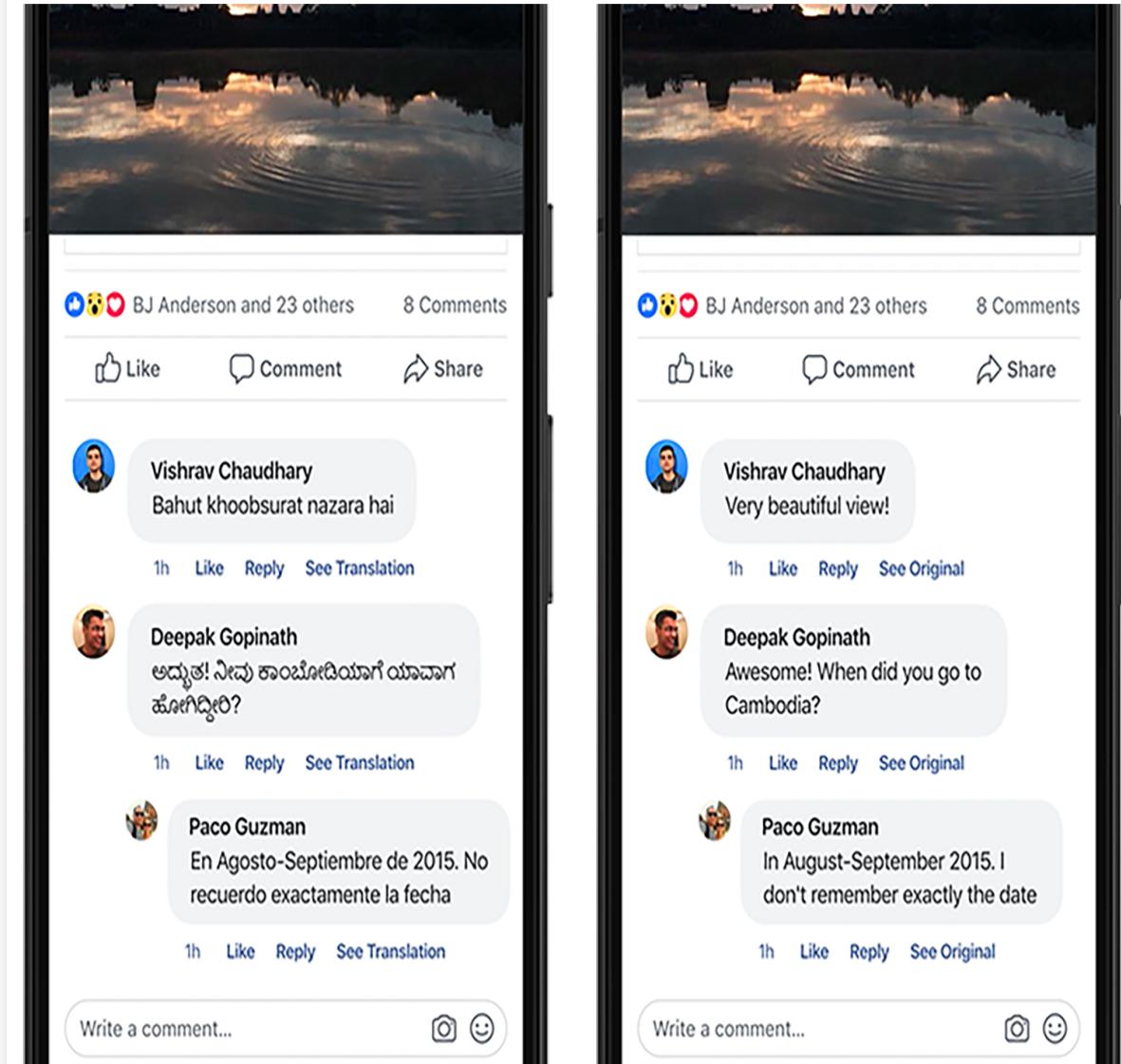
#GrowingUpWithSiblings

Source: My FB profile

# Transliteration



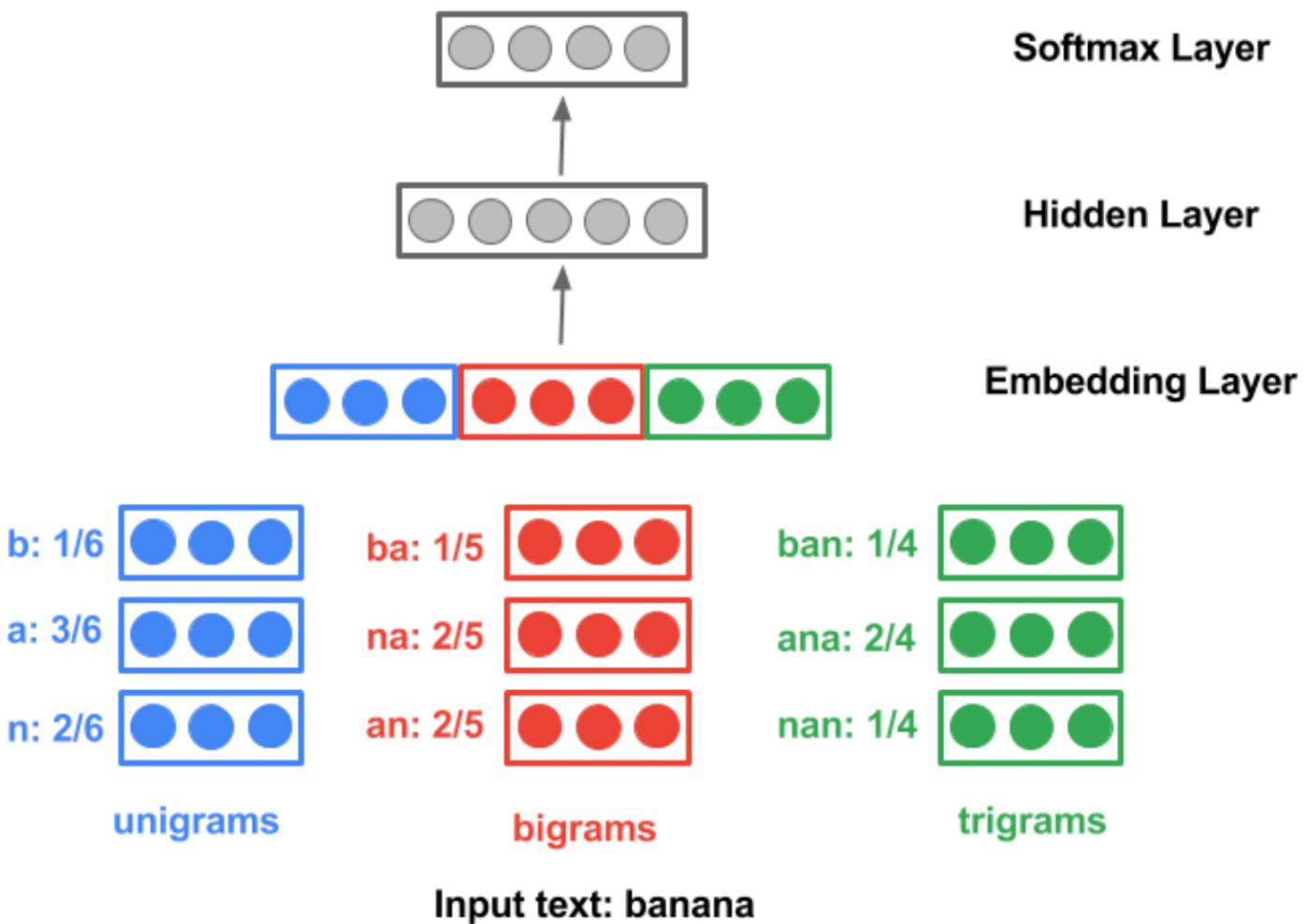
Converting words written in alphabet of one language to another



# International Components for Unicode

C/C++ and Java libraries for **Unicode** support

Devanāgarī	IAST	ISO 15919	Harvard-Kyoto	ITRANS	Velthuis	SLP1	WX
अ	a	a	a	a	a	a	a
आ	ā	ā	A	A/aa	aa	A	A
इ	i	i	i	i	i	i	i
ई	ī	ī	I	I/ii	ii	I	I
उ	u	u	u	u	u	u	u
ऊ	ū	ū	U	U/uu	uu	U	U
ए	e	ē	e	e	e	e	e
ऐ	ai	ai	ai	ai	ai	E	E
ओ	o	ō	o	o	o	o	o
औ	au	au	au	au	au	O	O
ऋ	r	ṛ	R	RRi/R^i	.r	f	q
ॠ	ṝ	ṝ	RR	RRI/R^I	.rr	F	Q
ऌ	l	়l	IR	LLi/L^i	.l	x	L
়l	়l	়l	IRR	LLI/L^I	.ll	X	LY
়m	m̥	m̥	M	M/.n./m	.m	M	M
়h	h̥	h̥	H	H	.h	H	H
়N		়N		.N		~	az
S	'	'	'	.a	.a	'	Z



# Language Identification: cld3

# cld3: Supported Languages

```
"eo", "co", "eu", "ta", "de", "mt", "ps", "te", "su", "uz", "zh-Latn", "ne",
"nl", "sw", "sq", "hmn", "ja", "no", "mn", "so", "ko", "th", "kk", "sl",
"ig", "mr", "zu", "ml", "hr", "bs", "lo", "sd", "cy", "hy", "uk", "pt",
"yi", "lv", "iw", "cs", "vi", "jv", "be", "km", "mk", "tr", "am", "zh",
"da", "sv", "fi", "ht", "af", "la", "id", "fil", "sm", "ca", "el", "ka",
"sr", "it", "sk", "ru", "ru-Latn", "bg", "ny", "fa", "fy", "haw", "gl",
"et", "ms", "gd", "bg-Latn", "ha", "is", "ur", "mi", "hi", "bn", "hi-Latn",
"fr", "hu", "xh", "my", "tg", "ro", "ar", "lb", "el-Latn", "st", "ceb",
"kn", "az", "si", "ky", "mg", "en", "gu", "es", "pl", "ja-Latn", "ga", "lt",
"sn", "yo", "pa", "ku",
```

```
2]: cld3.get_language("Je veux que: https://site.english.com/this/is/a/url/path/component#fragment")
2]: LanguagePrediction(language='en', probability=0.5319557189941406, is_reliable=False, proportion=1.0)

5]: cld3.get_frequent_languages("Je veux que: https://site.english.com/this/is/a/url/path/component#fragment",
                               num_langs=3)

5]: [LanguagePrediction(language='en', probability=0.5319557189941406, is_reliable=False, proportion=1.0),
      LanguagePrediction(language='und', probability=0.0, is_reliable=False, proportion=0.0),
      LanguagePrediction(language='und', probability=0.0, is_reliable=False, proportion=0.0)]
```

# cld3: Supported Languages

```
In [8]: # Some languages are easy to understand because they're available in this library  
cld3.get_language("Privet, kak tebya zovut?")
```

```
Out[8]: [LanguagePrediction(language='ru-Latn', probability=0.8417865633964539, is_reliable=True, proportion=1.0),  
LanguagePrediction(language='und', probability=0.0, is_reliable=False, proportion=0.0),  
LanguagePrediction(language='und', probability=0.0, is_reliable=False, proportion=0.0),  
LanguagePrediction(language='und', probability=0.0, is_reliable=False, proportion=0.0)]
```

```
In [12]: # Russian native script for "I want to go to the market"  
cld3.get_frequent_languages("Я иду на рынок сегодня", num_langs=4)
```

```
Out[12]: [LanguagePrediction(language='ru', probability=0.995514452457428, is_reliable=True, proportion=1.0),  
LanguagePrediction(language='und', probability=0.0, is_reliable=False, proportion=0.0),  
LanguagePrediction(language='und', probability=0.0, is_reliable=False, proportion=0.0),  
LanguagePrediction(language='und', probability=0.0, is_reliable=False, proportion=0.0)]
```

# cld3: Transliterated data

```
In [7]: # This is a text piece in Hindi transliterated to Roman characters, should support it, but doesn't do a good job
cld3.get_frequent_languages("Main Madhuri Dixit banna chahti hoon", num_langs=4)
# Predicts Finnish
```

```
Out[7]: [LanguagePrediction(language='fi', probability=0.47270092368125916, is_reliable=False, proportion=1.0),
LanguagePrediction(language='und', probability=0.0, is_reliable=False, proportion=0.0),
LanguagePrediction(language='und', probability=0.0, is_reliable=False, proportion=0.0),
LanguagePrediction(language='und', probability=0.0, is_reliable=False, proportion=0.0)]
```

```
In [9]: # This is transliterated Hindi for "How are you? I'm good"
cld3.get_frequent_languages("Kya haal hai? Main achhi hoon", num_langs=4)
# Gaelic is the prediction
```

```
Out[9]: [LanguagePrediction(language='gd', probability=0.4288159906864166, is_reliable=False, proportion=1.0),
LanguagePrediction(language='und', probability=0.0, is_reliable=False, proportion=0.0),
LanguagePrediction(language='und', probability=0.0, is_reliable=False, proportion=0.0),
LanguagePrediction(language='und', probability=0.0, is_reliable=False, proportion=0.0)]
```

# Cld3: Code-switched data

```
In [14]: # Let's try mix and match - short phrase in Spanglish meaning "but like"  
cld3.get_frequent_languages("Pero like", num_langs=2)  
# Predicts Maori - a language used by an indigenous group in NZ
```

```
Out[14]: [LanguagePrediction(language='mi', probability=0.8353496193885803, is_reliable=True, proportion=1.0),  
LanguagePrediction(language='und', probability=0.0, is_reliable=False, proportion=0.0)]
```

```
In [16]: cld3.get_frequent_languages("Cojelo con take it easy", num_langs=2)
```

```
Out[16]: [LanguagePrediction(language='en', probability=0.41589194536209106, is_reliable=False, proportion=1.0),  
LanguagePrediction(language='und', probability=0.0, is_reliable=False, proportion=0.0)]
```

```
In [20]: # Franglais  
cld3.get_frequent_languages("Ce week-end va être super cool.", num_langs=2)  
# Predicts catalan- western Romance language derived from Latin
```

```
Out[20]: [LanguagePrediction(language='ca', probability=0.5457612872123718, is_reliable=False, proportion=1.0),  
LanguagePrediction(language='und', probability=0.0, is_reliable=False, proportion=0.0)]
```

# Language Identification: Langid

```
In [31]: identifier = LanguageIdentifier.from_modelstring(model, norm_probs=True)
identifier.set_languages(['ru', 'en', 'it', 'sk'])
identifier.classify("Privet, kak tebya zovut?")

Out[31]: ('sk', 0.66767735005164)

In [29]: identifier.set_languages(['hi', 'en'])

In [30]: identifier.classify("Main Madhuri Dixit banna chahti hoon")

Out[30]: ('en', 0.999999999991774)
```

## How to detect the text's language

- Each language has the peculiar characters and spelling rule.
  - The accented “é” is used in Spanish, Italian and so on, and not used in English in principle.
  - The word that starts with “Z” is often used in German and rarely used in English.
  - The word that starts with “C” and contains spell “Th” are used in English and not used in German.
- Accumulates the probabilities assigned to these features in given text, so the guessed language is obtained as one that has the maximum probability.

	¤C	¤L	¤Z	Th
English	0.75	0.47	0.02	0.74
German	0.10	0.37	0.53	0.03
French	0.38	0.69	0.01	0.01

Source: <https://www.slideshare.net/shuyo/language-detection-library-for-java>

# Language Identification: Langdetect

<https://github.com/shuyo/language-detection>

<https://github.com/Mimino666/langdetect>

# langdetect

```
In [52]: detect_langs("Privet, kak tebya zovut?")
# Predicts Slovakian, Albanian, Hungarian
```

```
Out[52]: [sk:0.7142799432378928, sq:0.14285822332557807, hu:0.14285767370622154]
```

```
In [53]: detect_langs("Privet, kak tebya zovut?")
```

```
Out[53]: [sk:0.4081787863462692,
          sl:0.30518415356009926,
          hu:0.14285673157416331,
          sq:0.14285645846989373]
```

# Challenges



Detecting text with  
Romanized script



Small text length



Slang/ borrowed  
words



Different  
transliteration  
schemes



Overlapping  
vocabulary



Limited Data

# Building Custom Datasets

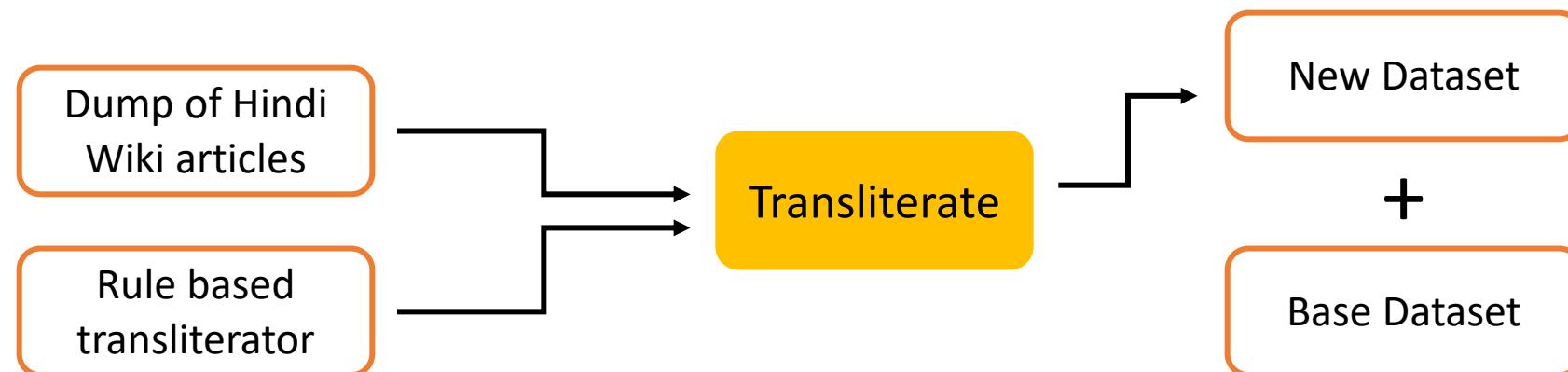
## WHY?

- Not enough data available in your desired language
- Do not want models trained on multiple languages to bring in noise

## HOW?

- Identify data source in any language that can be translated to desired language by rules/ machine
- Use this machine generated data to augment

## EXAMPLE



# Building Custom Datasets

{  
"କ": "k",  
"ଚ": "ch",  
"ଟ": "t",  
"ତ": "t",  
"ପ": "p",  
"ଖ": "kh",  
"ଛ": "chh",  
"ଠ": "th",  
"ଥ": "th",  
"ଫ": "ph",  
"ଗ": "g",  
"ଜ": "j",  
"ଡ": "d",  
"ଦ": "d",  
"ବ": "b",

```
In [6]: class Transliterator(object):

    def __init__(self):
        self.load_rules()

    def load_rules(self):
        with open("../rules.json", 'r', encoding='utf-8') as f:
            self.rules = json.load(f)

    self.consonants = CONSONANTS
    self.ein = ["ং"]
    self.n_sounds = ["ঁ", "ঃ"]

    def convert(self, string):
        converted = []
        for word in string.split(' '):
            res = ""
            for i, letter in enumerate(word):
                op = self.rules.get(letter, "")

                if isinstance(op, list):
                    op = op[0]
                if i < len(word) - 1 :
                    if letter in self.consonants and word[i+1] in self.consonants:
                        op += "a"
                    if letter in self.ein and word[i+1] in self.n_sounds:
                        op += "i"
                if i == len(word) - 1 and letter == "ঁ":
                    op = "ye"
                res += op
            converted.append(res)
        return " ".join(converted)
```

```
In [10]: # Here we transliterate a sentence from the Hindi Wiki page for Python  
translit.convert("पाइथन एक सामान्य कार्यों के लिए उपयुक्त, उच्च स्तरीय प्रोग्रामिंग भाषा, इन्टरैक्टिव, ॲब्जेक्ट ओरिएन्टेड, स्क्रिप्टिंग भाषा है। इस भाषा को इस तरह से डिजाइन किया गया है ताकि इसमें लिखे गए कोड आसानी से पढ़े और समझे जा सकें।")
```

**Out[10]:** 'paithan ek samany karyon ke liye upayukt uchch stariy programing bhasha intaraiktiv aubjekt oriented skripting bhasha hai is bhasha ko is tarah se dijain kiya gaya hai taki isamein likhe gye kod aasani se padhe aur samajhe ja sakein'

# Building Custom Datasets: csnli example

```
>>> from three_step_decoding import *
>>> tsd = ThreeStepDecoding('lid_models/hinglish', htrans='nmt_models/rom2hin.pt', etrans='nmt_models/eng2eng.
>>> print '\n'.join(['\t'.join(x) for x in tsd.tag_sent(u'i thght mosam dfrnt hoga bs fog h')])  
i i en  
thght thought en  
mosam मौसम hi  
dfrnt different en  
hoga होगा hi  
bs बस hi  
fog fog en  
h है hi  
>>> print '\n'.join(['\t'.join(x) for x in tsd.tag_sent(u'kafi dprsng situation hai yar')])  
kafi काफी hi  
dprsng depressing en  
situation situation en  
hai है hi  
yar यार hi
```

# Transformer

Introduced in Attention is All You  
Need by Vaswani et al.\*

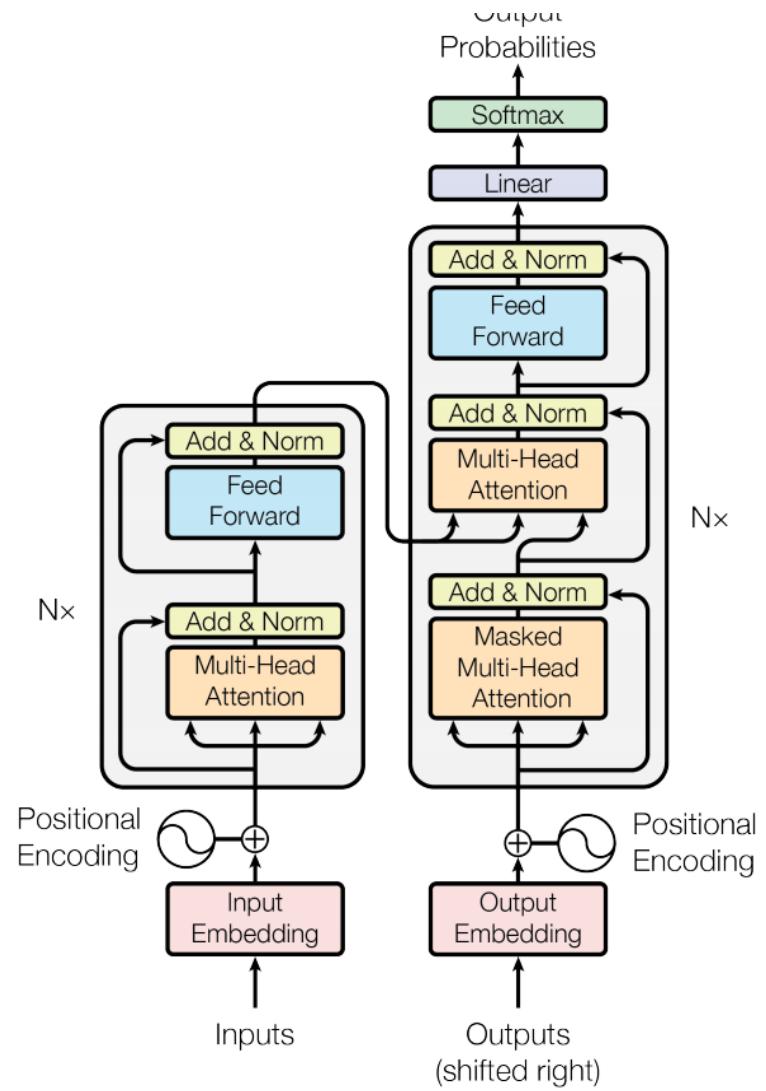
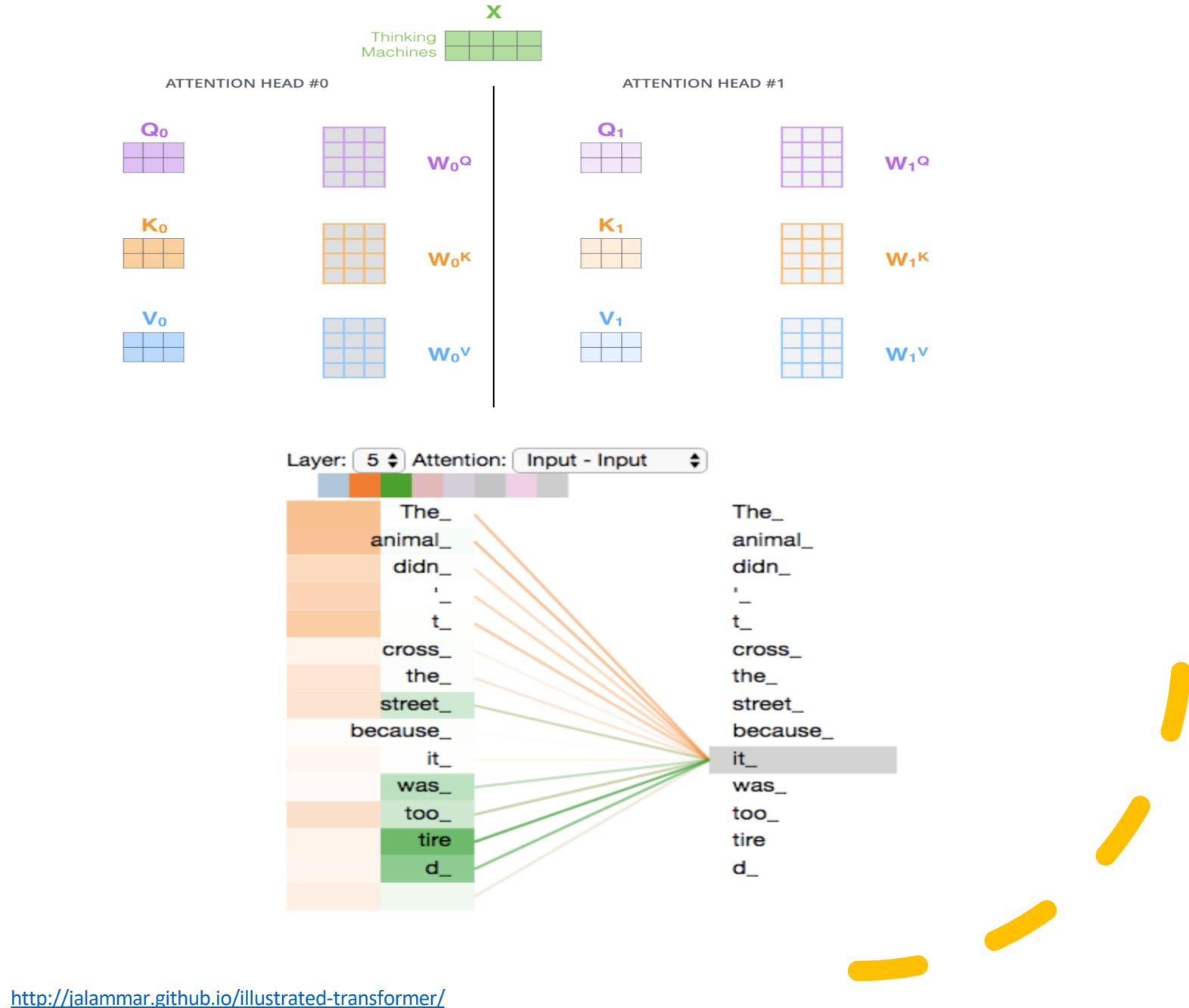


Figure 1: The Transformer - model architecture.

# Multi-headed Attention

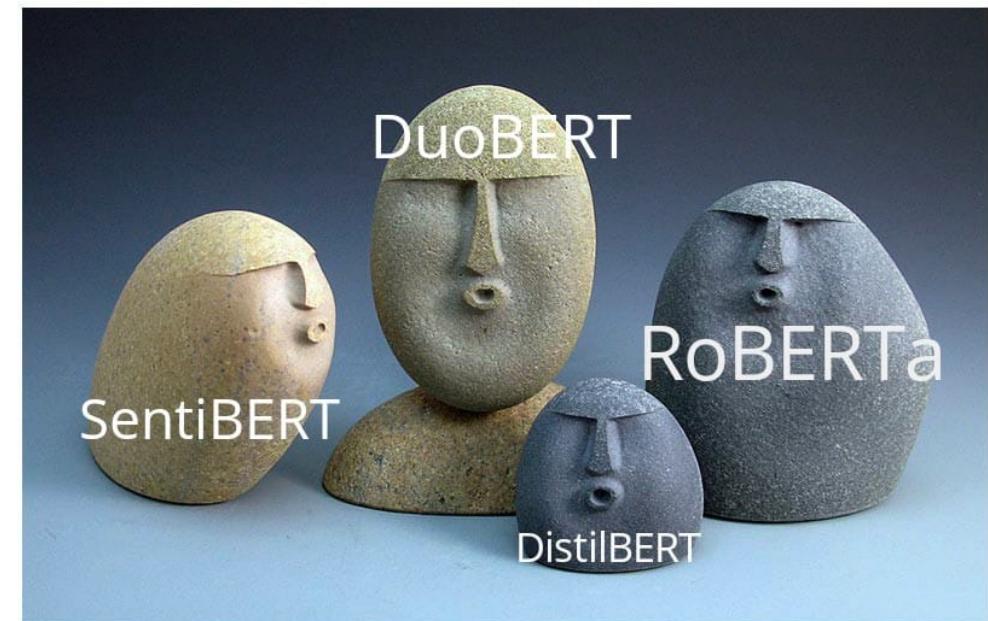


# BERT

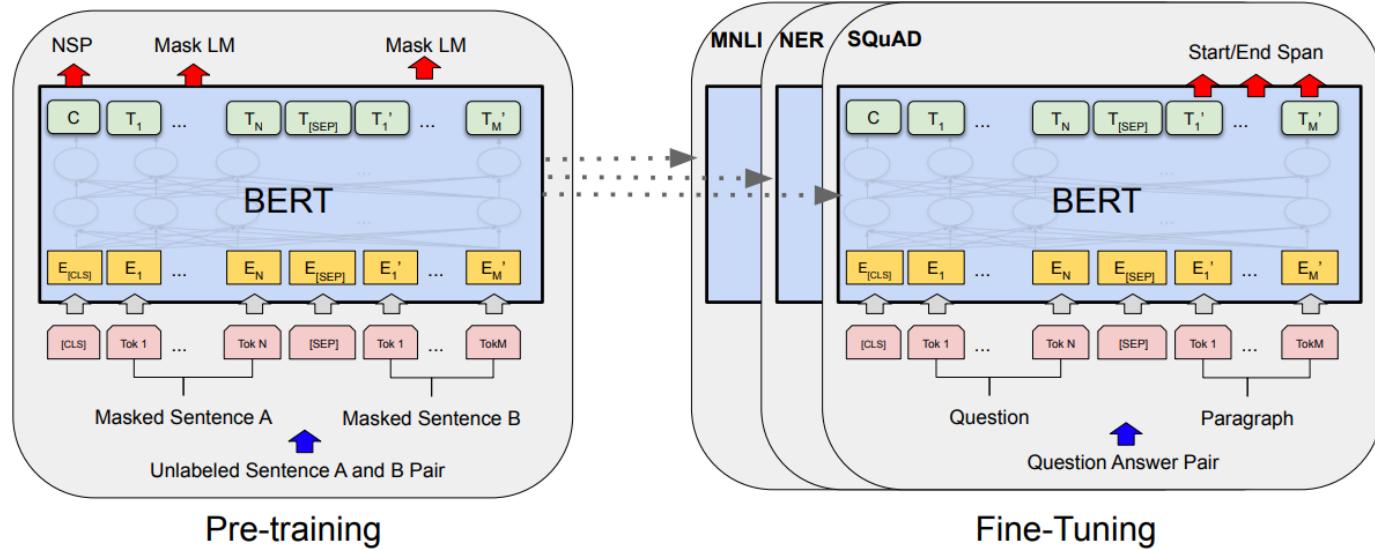


\*BERT invented \*

Literally every other new model:



# BERT - Tasks



- Pre-training: Train on unlabeled data
  - Next sentence prediction:
    - True label: IsNext 50% of the time
    - False Label: as NotNext
  - Masked LM
- Fine-tuning: Initialize with pre-trained weights and fine-tune for task with labeled data
  - Sentence pairs in paraphrasing
  - Hypothesis-premise pairs
  - Question-passage pairs in question answering
  - A degenerate text-Ø pair in text classification or sequence tagging.

# Pre-processing: WordPiece

What?

Replace common pairs of consecutive bytes with a byte that does not appear in that data.

Why?

Rare words' representation

How?

Start from character level and build up using likelihood of the possible candidates till desired vocab size is reached

# Pre-processing: WordPiece

```
In [3]: from tokenizers import *
from transformers import *

In [4]: tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-cased')
model = BertModel.from_pretrained('bert-base-multilingual-cased')

In [5]: tokenizer.tokenize("Hola quieres ir al centro comercial hoy")

Out[5]: ['Ho', '#la', 'quiere', '#s', 'ir', 'al', 'centro', 'comercial', 'hoy']

In [6]: tokenizer.tokenize("चलो आज कहीं चलते हैं")

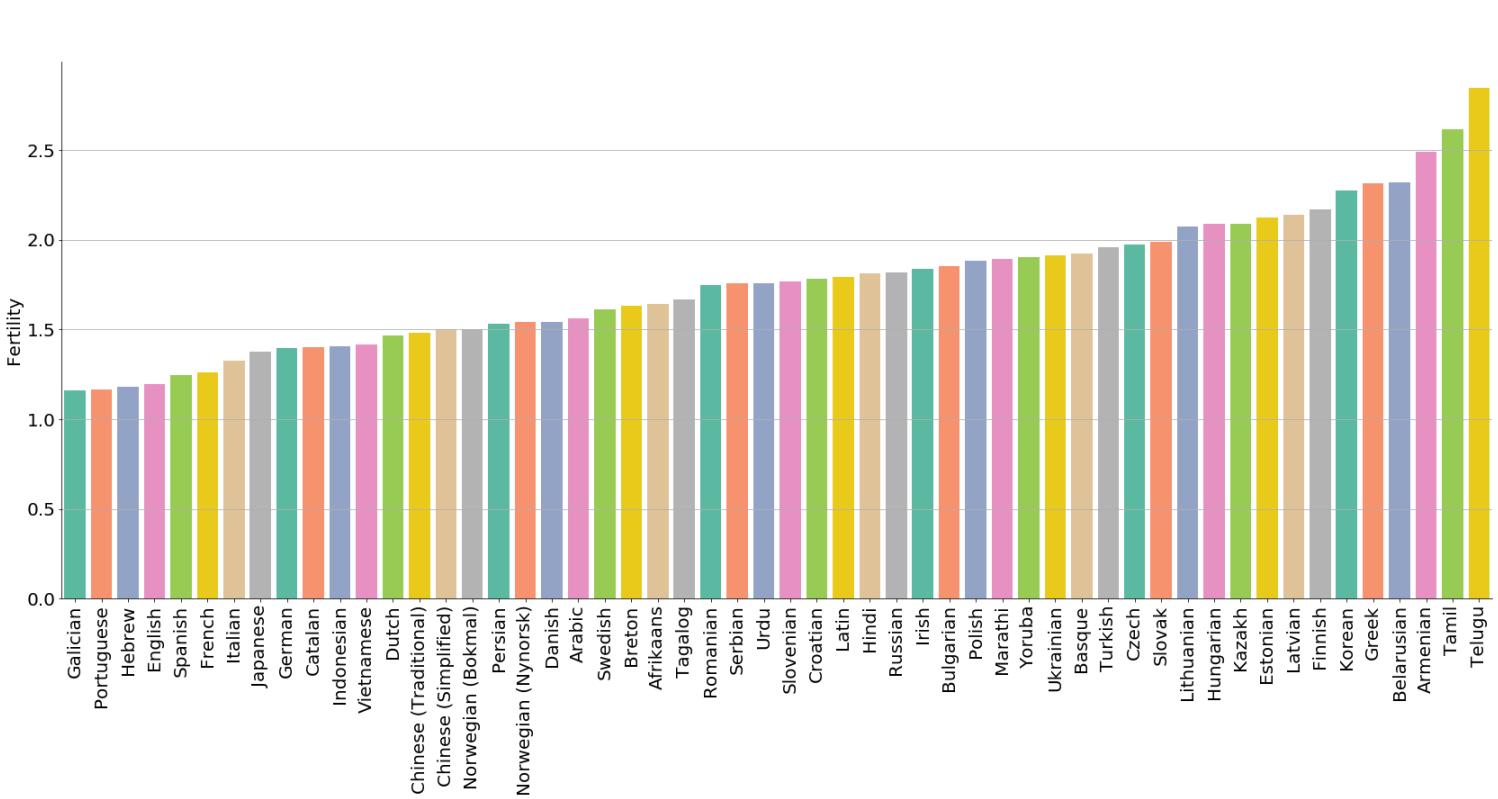
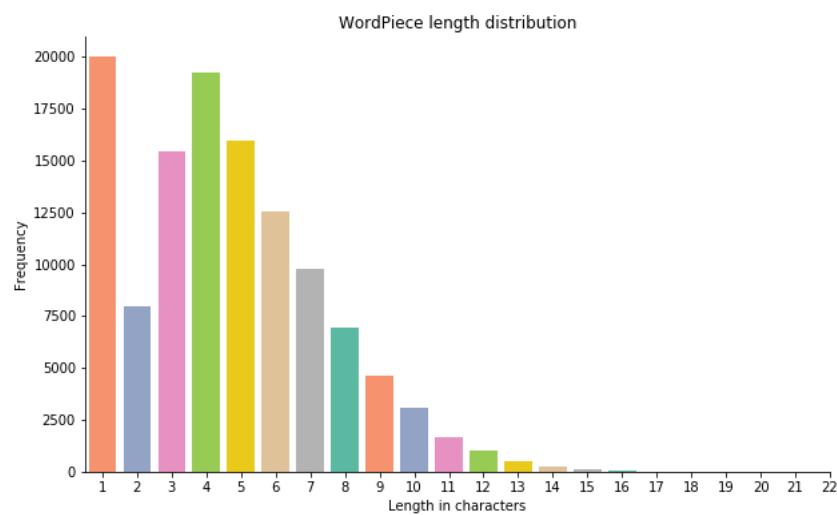
Out[6]: ['च', '#ल', '#ो', 'आज', 'क', '#ही', '#ते', 'च', '#ल', '#रै', 'हैं']

In [8]: tokenizer.tokenize("Bears. Beets. Battlestar Gallactica.")

Out[8]: ['Bears',
        '.',
        'Bee',
        '#ts',
        '.',
        'Battle',
        '#star',
        'Gall',
        '#act',
        '#ica',
        '..']
```

## Masked LM

# Pre-processing: WordPiece



# BERT – Masked Language Model

---

- Bidirectional
- Mask 15% of all WordPiece tokens in each sequence at random. Replace masked token with :
  - 80% probability by: [MASK]
  - 10% probability by: <random token>
  - 10% probability by: <original token>
- Evaluate by predicting the masked words

```
In [9]: model = BertForMaskedLM.from_pretrained('bert-base-multilingual-cased')
model.eval()
```

...

```
In [11]: def predict_word(text, target):

    tokenized_text = tokenizer.tokenize(text)
    masked_index = tokenized_text.index(target)
    tokenized_text[masked_index] = '[MASK]'

    # Convert token to vocabulary indices
    indexed_tokens = tokenizer.convert_tokens_to_ids(tokenized_text)

    # Define sentence A and B indices associated to 1st and 2nd sentences (see paper)
    segments_ids = [1] * len(tokenized_text)

    # Convert inputs to PyTorch tensors
    tokens_tensor = torch.tensor([indexed_tokens])
    segments_tensors = torch.tensor([segments_ids])

    # Predict all tokens
    predictions = model(tokens_tensor, segments_tensors)
    predicted_index = torch.argmax(predictions[0][0, masked_index, :]).item()
    predicted_token = tokenizer.convert_ids_to_tokens([predicted_index])

    print("Original:", text)
    print("Masked:", " ".join(tokenized_text))
    print("Predicted token:", predicted_token)
    print("Other options:")

    for i in range(10):
        predictions[0][0, masked_index, predicted_index] = -11100000
        predicted_index = torch.argmax(predictions[0][0, masked_index]).item()
        predicted_token = tokenizer.convert_ids_to_tokens([predicted_index])
        print(predicted_token)
```



```
In [16]: # iViva!
predict_word(text, "dejar")
```

Original:

This is what I want to say to everyone out there - vivir y dejar vivir!

Masked: This is what I want to say to everyone out there - vivir y [MASK] vivir !

Predicted token: ['no']

Other options:

- ['solo']
- ['non']
- ['bien']
- ['nunca']
- ['yo']
- ['a']
- ['sólo']
- ['también']
- ['hacer']
- [',']

---

```
In [14]: predict_word(text, "vivir")
```

Original:

This is what I want to say to everyone out there - vivir y dejar vivir!

Masked: This is what I want to say to everyone out there - [MASK] y dejar vivir !

Predicted token: ['i']

Other options:

- ['...']
- ['Viva']
- ['Vive']
- ['Hasta']
- ['dejar']
- ['Dar']
- ['empezar']
- ['Gracias']
- ['volver']
- ['Por']

---



# BERT - MLM

```
In [40]: predict_word(text, "मा")
```

Original:

भारतीय माँ सुबह 8 बजे - बेटा, उठो, 12 बज गए हैं

Masked: भारतीय [MASK] ##ॐ ता ##कु ##ब ##ह 8 ब ##जे - ब ##लेट ##ा , 3 ##अ ##ो , 12 ब ##ज गए हैं

Predicted token: ['मा']

Other options:

- ['आ']
- ['कहा']
- ['जा']
- ['अ']
- ['खा']
- ['हा']
- ['औ']
- ['का']
- ['बी']
- ['ऊ']

X

BERT-  
MLM:  
Short  
Hindi  
Example

# BERT – MLM: Code-switched Example

```
In [25]: # It was able to recognize the masked word as its predicted first choice! But notice how useful the bidirectionality is here.
```

```
predict_word(text, "blog")
```

Original:

अगर आप हिंदी में जानकारी पढ़ना पसंद करते हैं तो आपको best hindi blogs यानी भारत में popular Hindi bloggers कौन-कौन से हैं इसकी जानकारी होनी चाहिए।

बहुत से लोगों को इंग्लिश में जानकारी होते हुए भी वो हिंदी में blog पढ़ना पसंद करते हैं लेकिन इसके लिए हमारे पास कुछ popular Hindi blogs list होने चाहिए।

Masked: अ ##गर आ ##प हिंदी में जानकारी प ##ड ##ना प ##स ##ंद करते हैं तो आ ##प ##को best hindi [MASK] ##s या ##नी भारत में popular Hindi blog ##gers क ##ौ ##न - क ##ौ ##न से हैं इसकी जानकारी हो ##नी चाहिए। बहुत से लोगों को इंग्लिश में जानकारी होते हुए भी व ##ौ ##हिंदी में blog प ##ड ##ना प ##स ##ंद करते हैं लेकिन इसके लिए हम ##०रे पास कुछ popular Hindi blog ##s list होने चाहिए।

Predicted token: ['blog']

Other options:

- ['Blog']
- ['publisher']
- ['forum']
- ['tag']
- ['poster']
- ['photographer']
- ['peer']
- ['reporter']
- ['guru']
- ['pseudonym']

# BERT – MLM: Code-switched Example

In [28]: # Let's try another word here

```
predict_word(text, "इंग्लिश") —————> English
```

Original:

अगर आप हिंदी में जानकारी पढ़ना पसंद करते हैं तो आपको best hindi blogs यानी भारत में popular Hindi bloggers कौन-कौन से हैं इसकी जानकारी होनी चाहिए।

बहुत से लोगों को इंग्लिश में जानकारी होते हुए भी वो हिंदी में blog पढ़ना पसंद करते हैं लेकिन इसके लिए हमारे पास कुछ popular Hindi blogs list होने चाहिए।

Masked: अ ##गर आ ##प हिंदी में जानकारी प ##ढ़ ##ना प ##स ##ंद करते हैं तो आ ##प ##को best hindi blog ##s या ##नी भारत में popular Hindi blog ##gers क ##०१ ##०२ - क ##०१ ##०२ से हैं इसकी जानकारी हो ##नी चाहिए। बहुत से लोगों को [MASK] में जानकारी होते हुए भी व ##०१ हिंदी में blog प ##ढ़ ##ना प ##स ##ंद करते हैं लेकिन इसके लिए हम ##०३रे पास कुछ popular Hindi blog ##s list होने चाहिए।

Predicted token: ['हिंदी']

Other options:

- ['हिन्दी'] —————> Hindi
- ['अंग्रेजी'] —————> Hindi for “English”
- ['भारत'] —————> India
- ['अंग्रेजी']
- ['मराठी']
- ['उर्दू']
- ['Hindi']
- ['भाषा']
- ['ગुજરाती']
- ['इंग्रजी']

# BERT – MLM: Code-switched Example

```
In [35]: predict_word(text, "Mein")
# Notice WHY it was able to correctly predict it -> it has a couple of other similar contexts to compare to.
```

Original: Aasma Hai Neela Kyun, Paani Geela Geela Kyun  
Gol Kyun Hai Zameen, Sill Mein Hai Narmi Kyun  
Aag Mein Pai Garmi Kyun, Do Aur Do Paanch Kyun Nahi  
Pedd Le Gaye Kam Kyun, Teer Hain Ye Mausam Kyun  
Chand Do Kyun Nahi, Duniya Mein Hai Jung Kyun  
Behta Laal Rang Kyun, Saradein Hain Kyun Har Kahan  
Socha Hai... Yeh Tumne Kya Kabhi  
Socha Hai... Ki Hain Yeh Kya Sabhi Socha Hai  
Socha Nahi To Socho Abhi.....  
Behti Kyun Hai Har Nadi, Hoti Kya Hai Roshni  
Barf Girti Hai Kyun, Dost Kyun Hain Roothte  
Taare Kyun Hain Tuttey, Baadlon Mein Bijli Hai Kyun  
Socha Hai... Yeh Tumne Kya Kabhi  
Socha Hai... Ki Hain Yeh Kya Sabhi Socha Hai  
Socha Nahi To Socho Abhi  
Sannata Sunai Nahin Deta, Aur Hawayen Dikhayi Nahin Deti  
Socha Hai Kya Kabhi, Hota Hai Yeh Kyun

SIMILAR CONTEXTS!

Masked: Aa ##sma Hai Ne ##ela Ky ##un , Pa ##ani G ##eel ##a G ##eel ##a Ky ##un Gol Ky ##un Hai Za ##meen , Silk [MA SK] Hai Na ##rmi Ky ##un Aa ##g Mein Hai Ga ##rmi Ky ##un , Do Aur Do Pa ##anc ##h Ky ##un Na ##hi Pe ##dd Ho Gay ##e Ka ##m Ky ##un , Teen Hai ##n Ye Mau ##sam Ky ##un Chan ##d Do Ky ##un Na ##hi , Du ##niya Mein Hai Jung Ky ##un Be # #hta La ##al Rang Ky ##un , Sar ##had ##ein Hai ##n Ky ##un Har Ka ##hin Soc ##ha Hai . . . Ye ##h Tu ##mne Ky ##a Ka ##bh ##i Soc ##ha Hai . . . Ki Hai ##n Ye ##h Ky ##a Sa ##bh ##i Soc ##ha Hai Soc ##ha Na ##hi To Soc ##ho Ab ##hi . . . . Be ##hti Ky ##un Hai Har Nad ##i , Hot ##i Ky ##a Hai Ros ##hn ##i Bar ##f G ##irt ##i Hai Ky ##un , Dos ##t Ky ##un Hai ##n Root ##hte Ta ##are Ky ##un Hai ##n Tutte ##y , Ba ##ad ##lon Mein Bij ##li Hai Ky ##un Soc ##ha Hai . . . Ye ##h Tu ##mne Ky ##a Ka ##bh ##i Soc ##ha Hai . . . Ki Hai ##n Ye ##h Ky ##a Sa ##bh ##i Soc ##ha Hai Soc ##ha Na ##hi To Soc ##ho Ab ##hi San ##nata Sun ##ai Na ##hin Det ##a , Aur Ha ##way ##en Di ##kha ##yi Na ##hin Det ##i Soc ##ha Hai Ky ##a Ka ##bh ##i , Hot ##a Hai Ye ##h Ky ##un

Predicted token: ['Mein']

Other options:

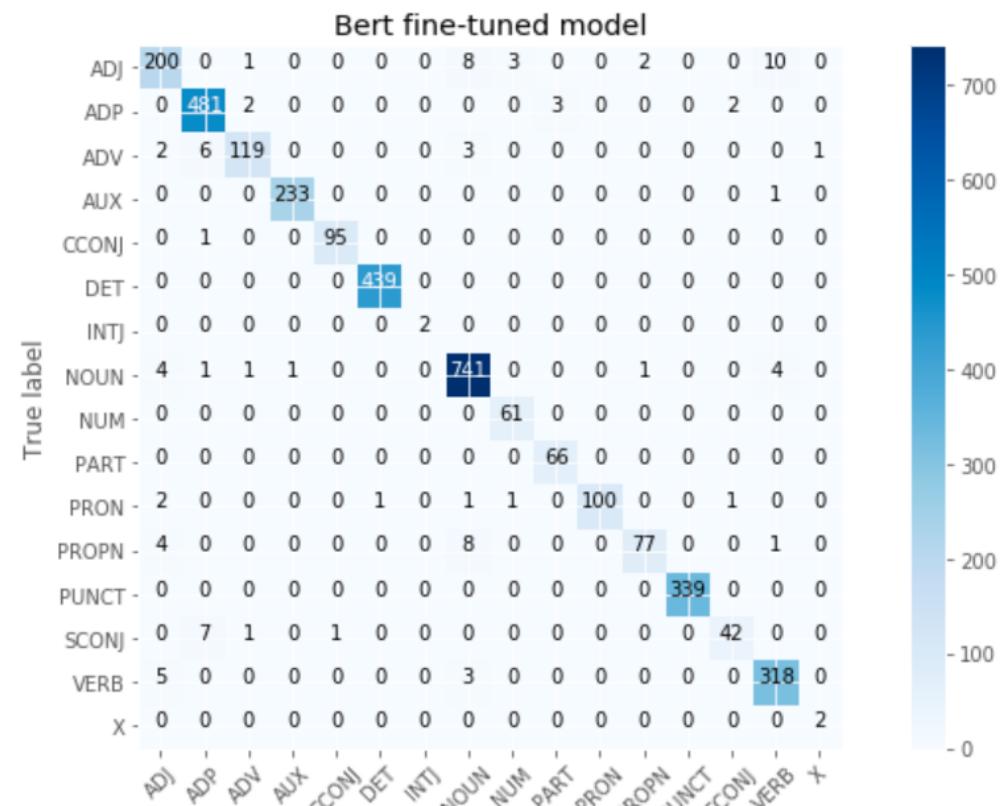
- ['##e']
- ['##en']
- ['##y']
- ['##i']
- ['##ei']
- ['Hai']
- ['##in']
- ['##a']
- ['##em']
- ['##er']

# BERT Evaluation: POS Tagging

- Using BERT representations as inputs
- Fine-tuning

Jim	Hen	##son	was	a	puppet	##eer
I-PER	I-PER	X	0	0	0	X

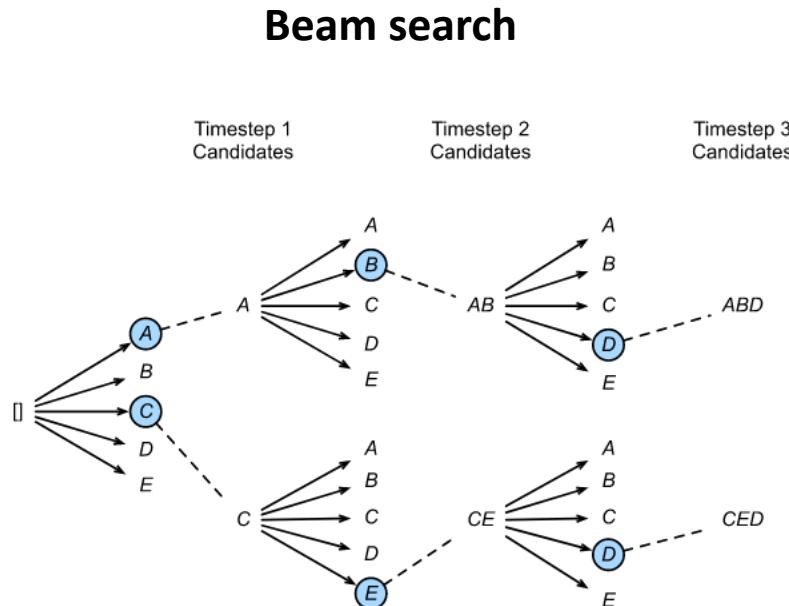
NOT PREDICTED!



# BERT Multilingual Evaluation: Generation

- Masked LM: Top k accuracy
- NMT: BLEU score, beam search

Predicted token: ['i']  
Other options:  
['...']  
['Viva']  
['Vive']  
['Hasta']  
['dejar']  
['Dar']  
['empezar']  
['Gracias']  
['volver']  
['Por']



## BLEU score

Here,  $c$  = candidate length  
 $r$  = reference length  
 $w_n$  = weights of n-gram  
 $p_n$  = modified precision of n-gram

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} .$$

Then,

$$\text{BLEU} = BP \cdot \exp \left( \sum_{n=1}^N w_n \log p_n \right) .$$

# How to make your model multilingual?

IS IT  
REQUIRED?

WHICH  
LANGUAGES?

ADD DATA!

CHOOSE PRE-  
TRAINED  
MODEL

FINE TUNE IF  
NEEDED

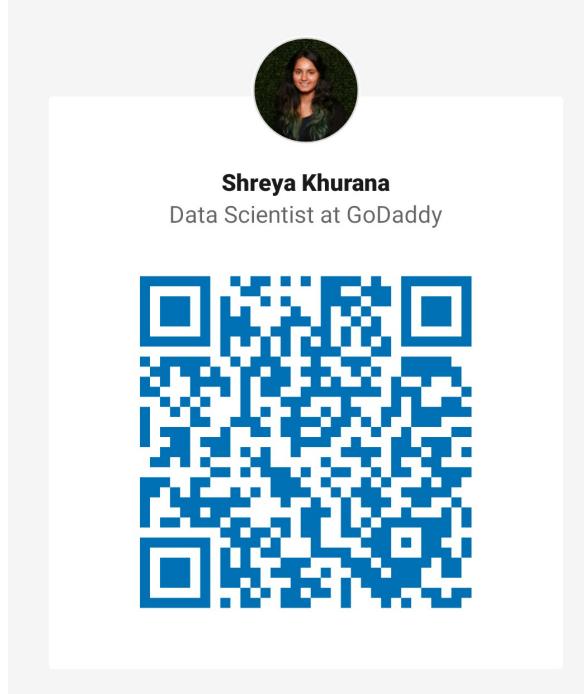
VOILA!



# Questions?

# Thank you!

---



A white rectangular card featuring a circular profile picture of a woman with dark hair, identified as Shreya Khurana. Below the photo, her name is printed in bold black text, followed by her title "Data Scientist at GoDaddy". To the right of the card is a large blue QR code.

**Shreya Khurana**  
Data Scientist at GoDaddy



<https://github.com/ShreyaKhurana>