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:

https://github.com/ShreyaKhurana/pycon

Outline

- Introduction to Multilingual Data
- Code-switched data and transliteration
- Language Identification
- Challenges
- Transformer
- BERT
- BERT Masked LM
- Evaluation Metrics

Multilingual Data

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.’

http://www.ello.uos.de/field.php/Sociolinguistics/Codeswitching

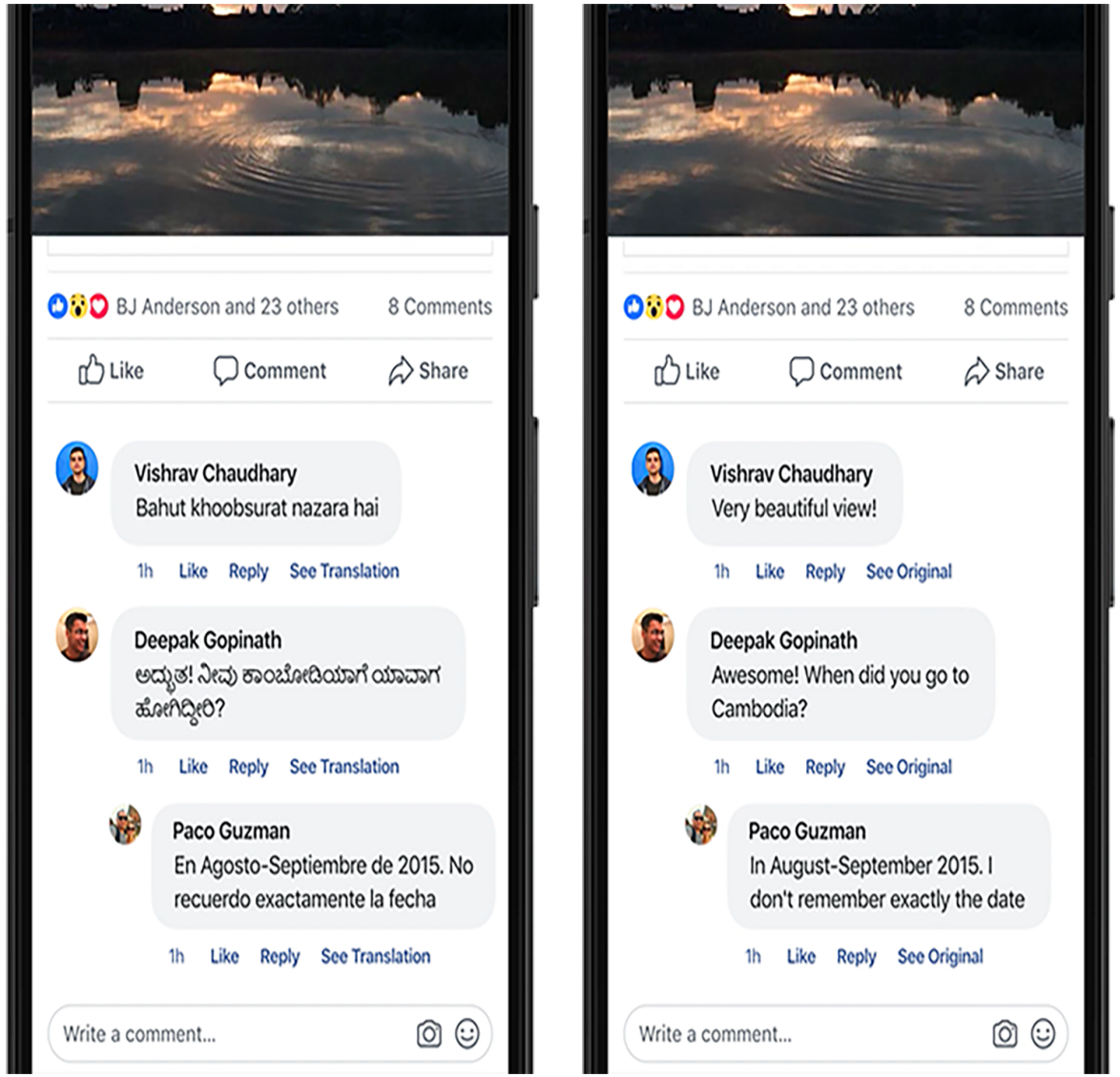
Code-Switching

Alternating between two languages

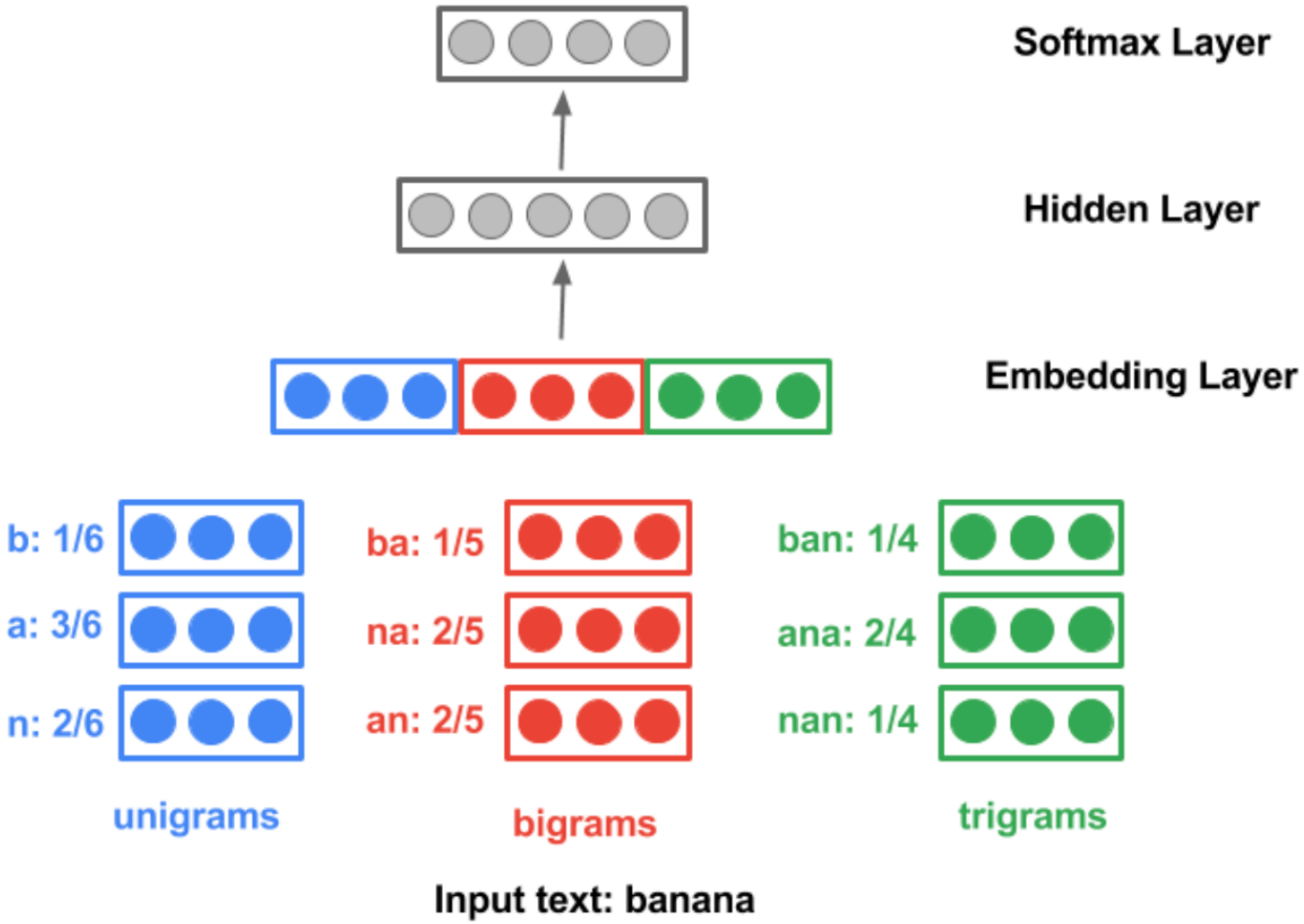
Source: My FB profile

Transliteration

Converting words written in alphabet of one language to another



https://engineering.fb.com/ml-applications/expanding-automatic-machine-translation-to-more-languages/



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)
```

https://www.aclweb.org/anthology/P12-3005.pdf

https://github.com/saffsd/langid.py

CybozuLabs

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:0.1428582232557807, hu:0.1428567370622154]

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

Out[53]: [sk:0.4081787863462692,
          sl:0.30518415356009926,
          hu:0.1428567315741633 1,
          sq:0.1428564584698937 3]
```

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

https://github.com/Mimino666/langdetect

Challenges

Detecting text with Romanized script

Small text length

Slang/ borrowed words

Different transliteration schemes

Overlapping vocabulary

Limited Data

Building Custom Datasets

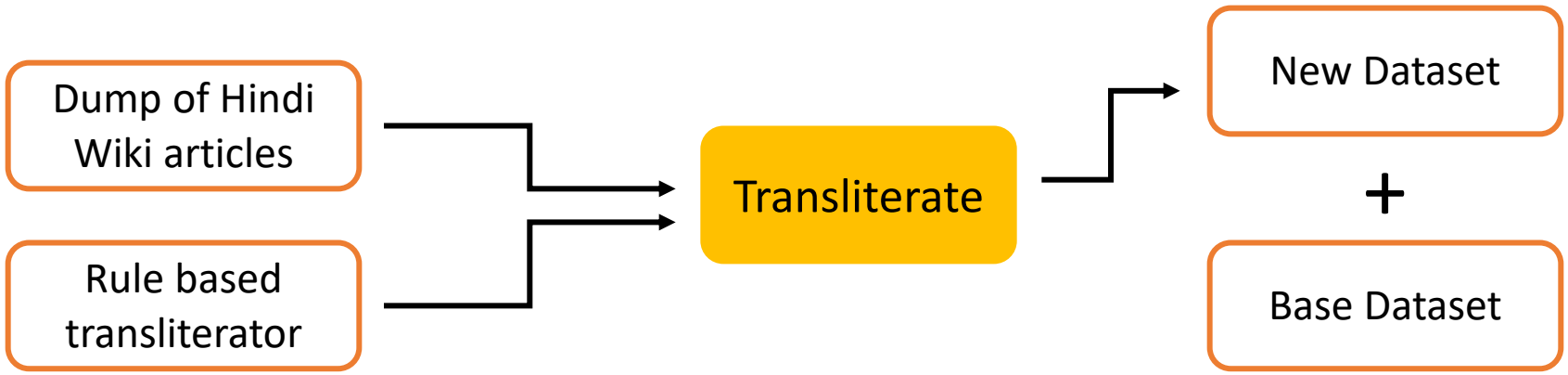
WHY?

- 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

```
{
    "क": "k",
    "च": "ch",
    "ट": "t",
    "त": "t",
    "प": "p",
    "ख": "kh",
    "छ": "chh",
    "ठ": "th",
    "थ": "th",
    "फ": "ph",
    "ग": "g",
    "ज": "j",
    "ड": "d",
    "द": "d",
    "ब": "b",
```

In [6]:
```python
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 == "ए":
                    op = "ye"
                res += op
            converted.append(res)
        return " ".join(converted)
```

In [10]:
```python
# 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, ऑब्जेक्ट 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 bhash
a 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
```

https://arxiv.org/pdf/1703.10772.pdf

https://github.com/irshadbhat/csnli

Transformer

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

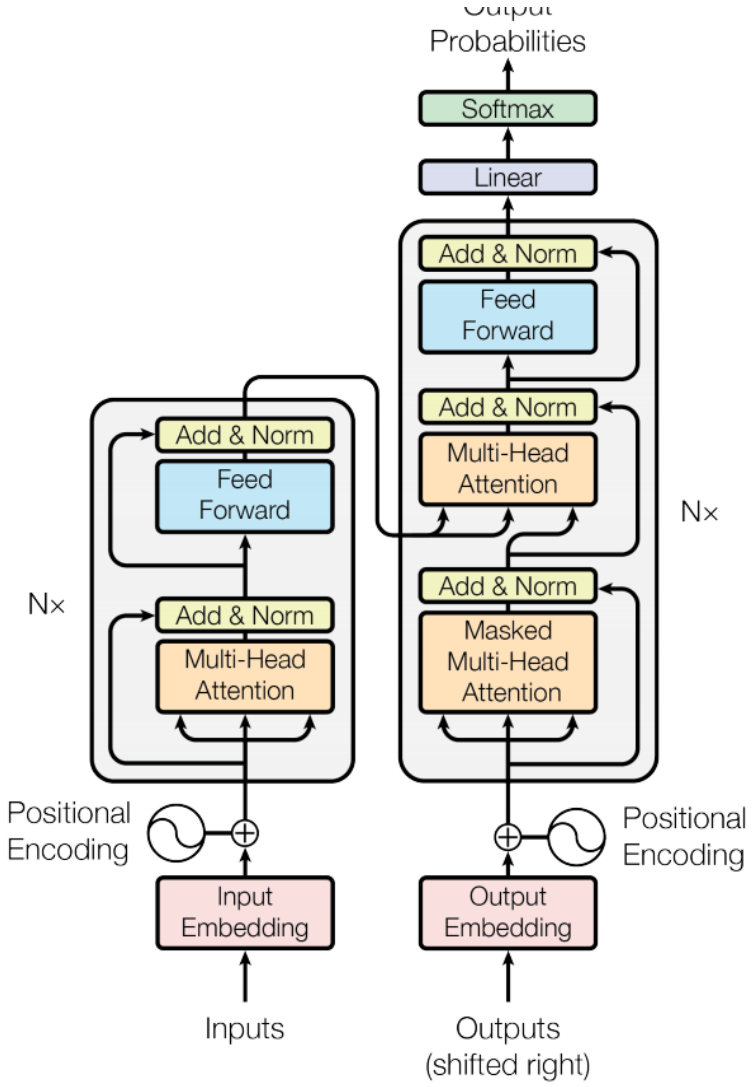
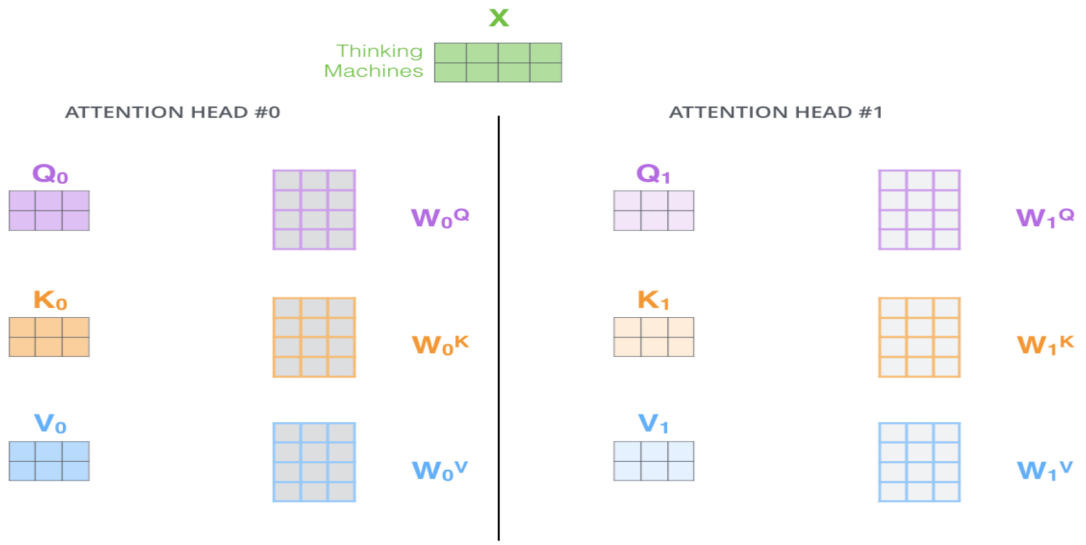
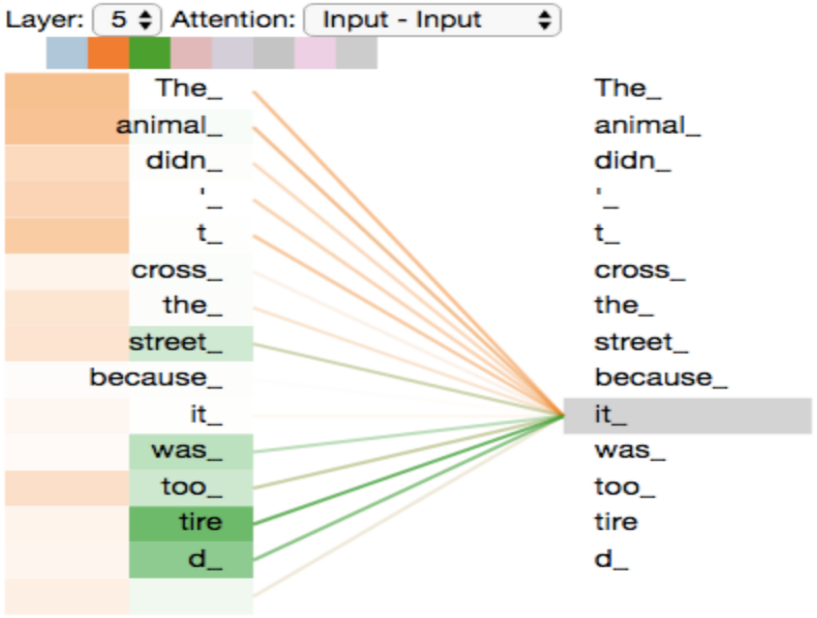


Figure 1: The Transformer - model architecture.

https://arxiv.org/abs/1706.03762

Multi-headed Attention



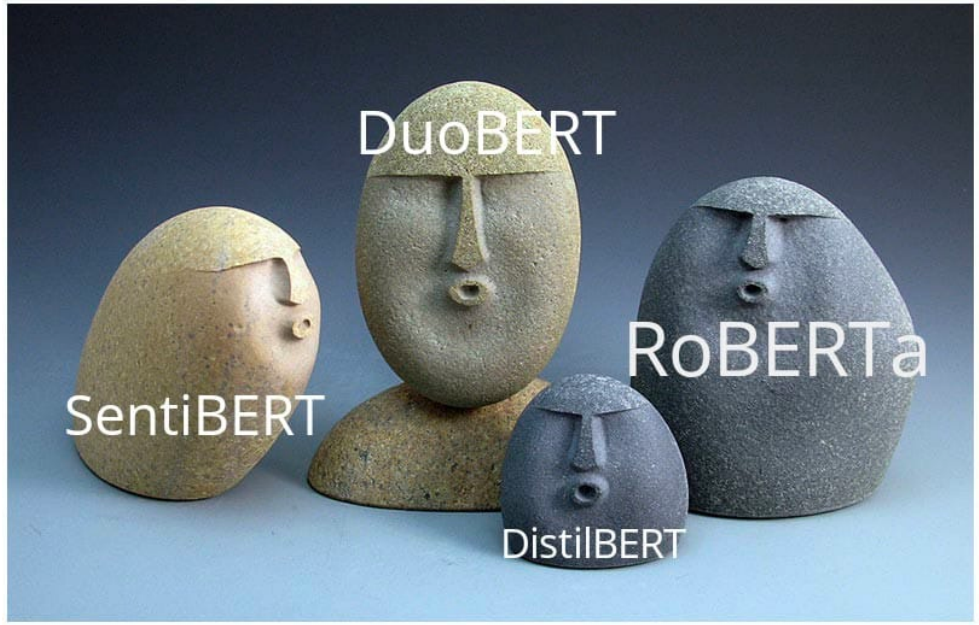


http://jalammar.github.io/illustrated-transformer/

BERT

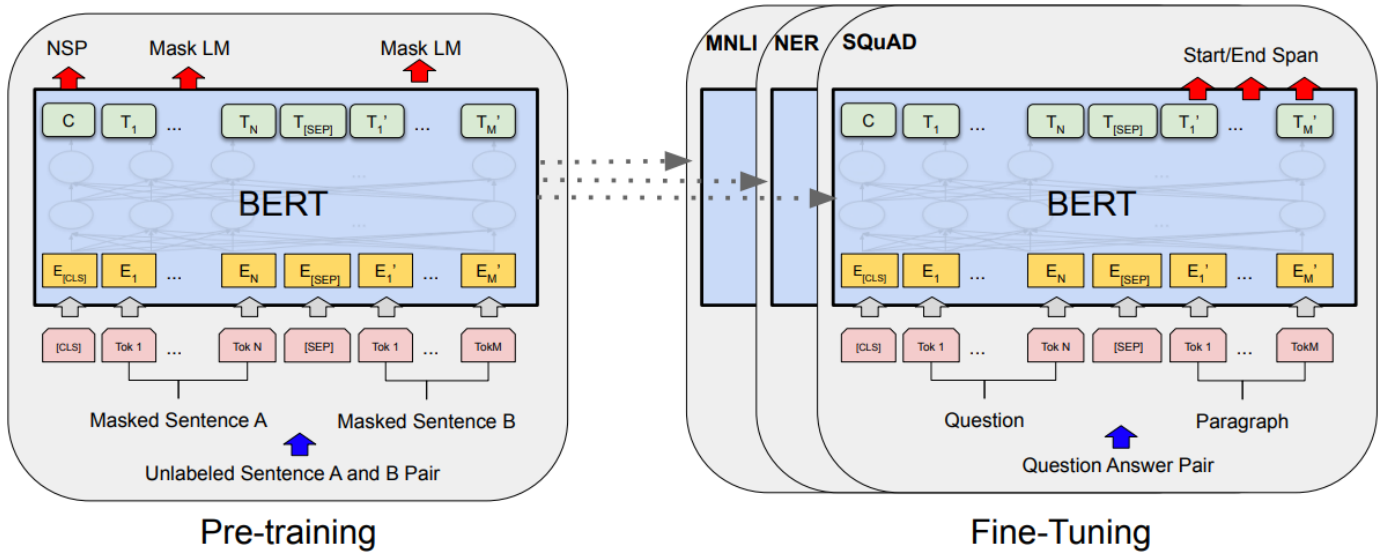
*BERT invented *

Literally every other new model:



Meme source: https://www.facebook.com/groups/1638417209555402/

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

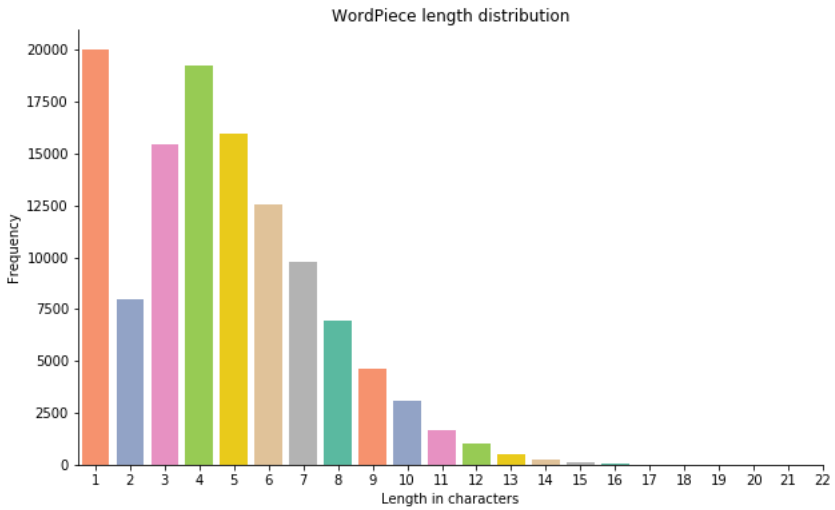
```
In [3]: from tokenizers import *
        from transformers import *

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', '##ं', 'च', '##ल', '##ते', 'हैं']

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

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

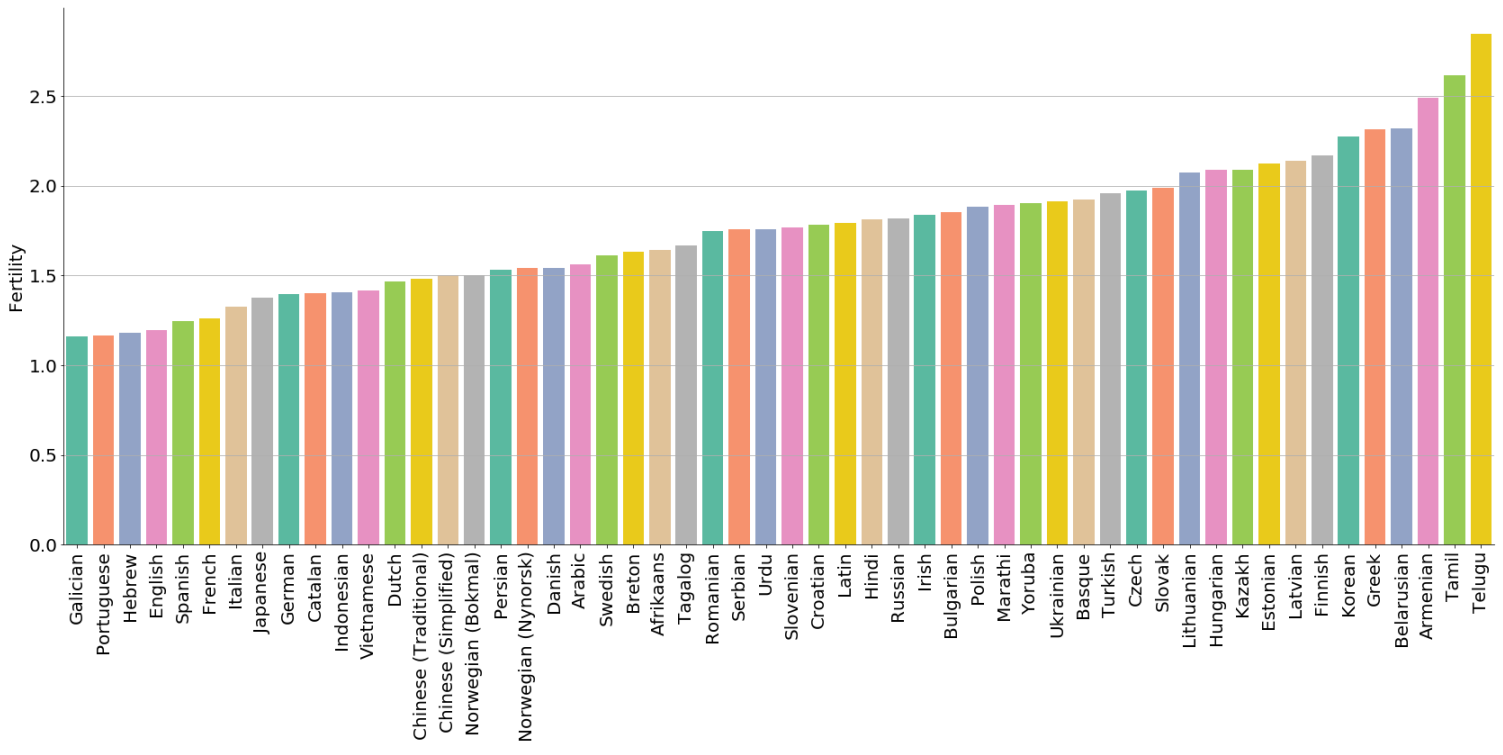
Masked LM

Paper: Sennrich, R., Haddow, B., & Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Pre-processing: WordPiece

Fertility = average number of BERT word pieces corresponding with a single real token





http://juditacs.github.io/2019/02/19/bert-tokenization-stats.html

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

BERT-MLM

```
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)
```

https://mayhewsw.github.io/2019/01/16/can-bert-generate-text/

BERT - MLM

```
In [16]: # ¡Viva!
         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']   ---> just
['non']
['bien']   ---> well
['nunca']
['yo']
['a']
['sólo']
['también'] ---> also
['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: ['¡']
Other options:
['...']
['Viva']
['Vive']
['Hasta']
['dejar']
['Dar']
['empezar']
['Gracias']
['volver']
['Por']
```

```
In [40]: predict_word(text, "मा")

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

Masked: भारतीय [MASK] ##ँ स ##ु ##ब ##ह 8 ब ##जे - ब ##ेट ##ा , उ ##ठ ##ो , 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 ##ढ़ ##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 ##ढ़ ##ना प ##स ##ंद करते हैं लेकिन इसके लिए हम ##ारे 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: Long Transliterated 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, Silk Mein Hai Narmi Kyun
Aag Mein Hai Garmi Kyun, Do Aur Do Paanch Kyun Nahi
Pedd Ke Gaye Kam Kyun, Teen Hain Ye Mausam Kyun
Chand Do Kyun Nahi, Duniya Mein Hai Jung Kyun
Behta Laal Rang Kyun, Sar[illegible] Hain Kyun Har Kahin
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

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 ##h
a 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']
```

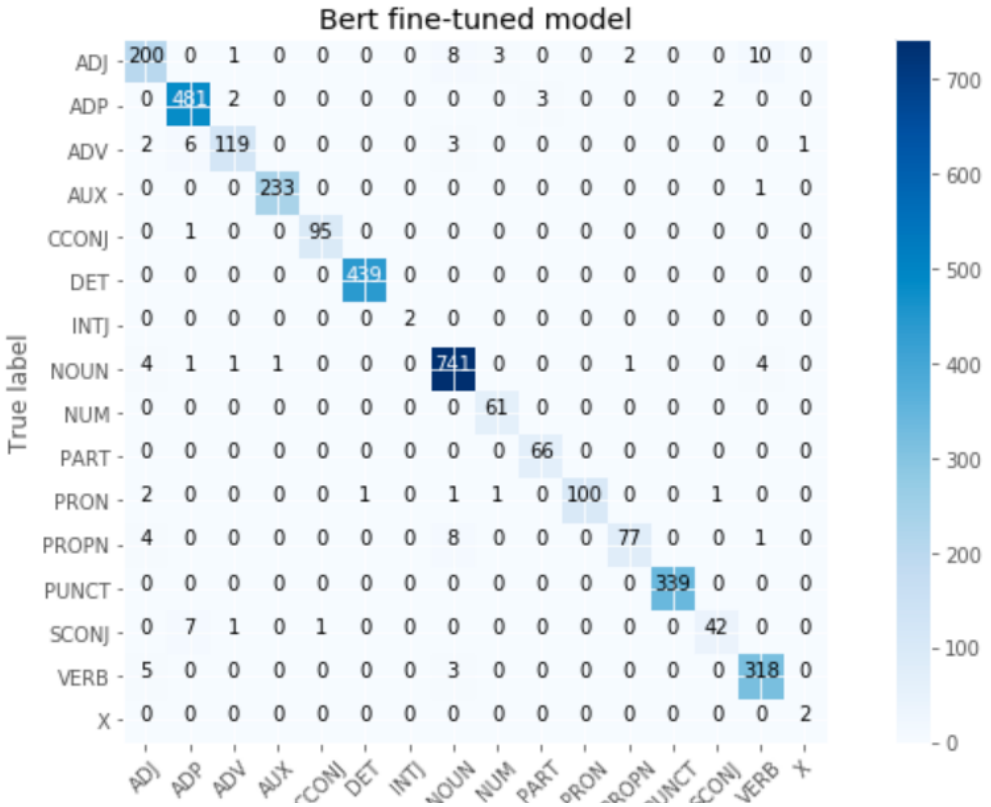
SIMILAR CONTEXTS!

BERT Evaluation: POS Tagging

- Using BERT representations as inputs
- Fine-tuning

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

NOT PREDICTED!



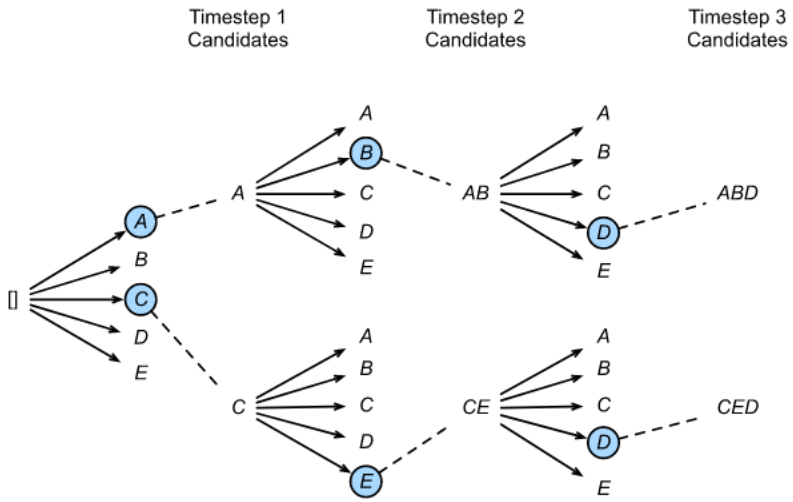
https://github.com/soutsios/pos-tagger-bert

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']
```

Beam search



BLEU score

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

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

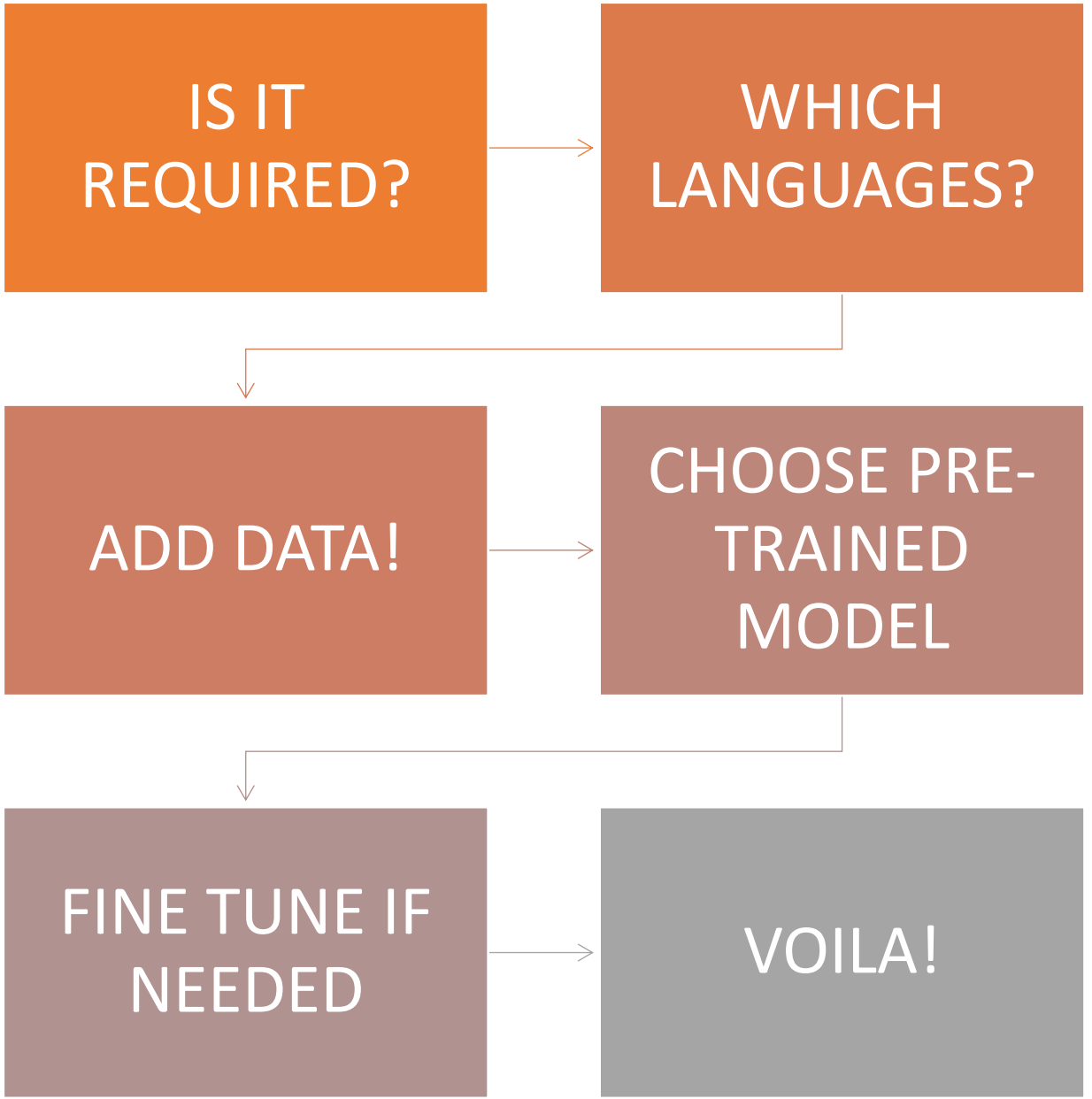
Then,

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

https://github.com/soutsios/pos-tagger-bert

BLEU paper: https://www.aclweb.org/anthology/P02-1040.pdf

How to make your model multilingual?



Key Takeaways

- Changing grammars with transliterated and code-switched data
- Language Identification Challenges
- Augmenting Datasets
- BERT Multilingual performance
- You CAN make a NLP presentation without a word cloud!

Questions?

Thank you!

Shreya Khurana
Data Scientist at GoDaddy

https://github.com/ShreyaKhurana