Next in NLP

Prompt Engineering for NLU

Avinash Kaur, Data Scientist (avkaur@nvidia.com)
Ashish Sardana, Deep Learning Engineer (asardana@nvidia.com)

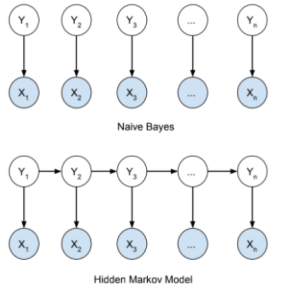
27th Oct 2021

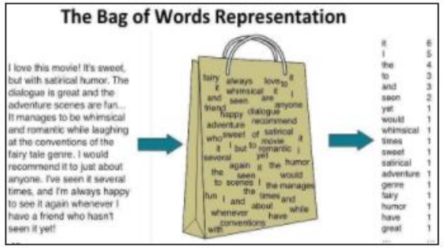
Agenda

- Four Paradigms in NLP
- Prompt Engineering & P-tuning
- Future directions
- Pass it to Ashish

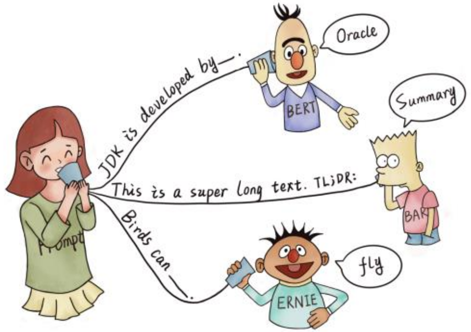
TWO SEA CHANGES IN NLP

statistical methods

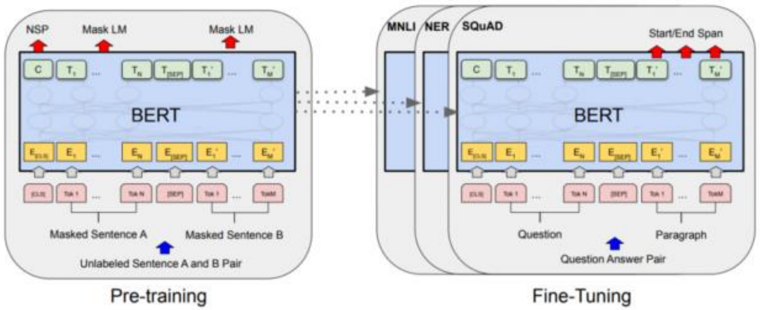




2019 – now pre-train, prompt, and predict

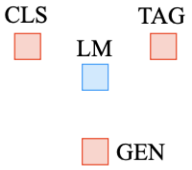
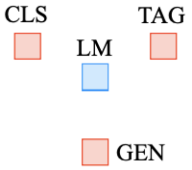
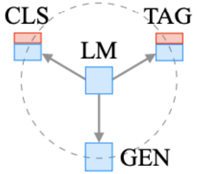
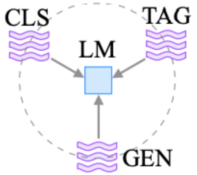


2017-2019 pre-train, fine-tune



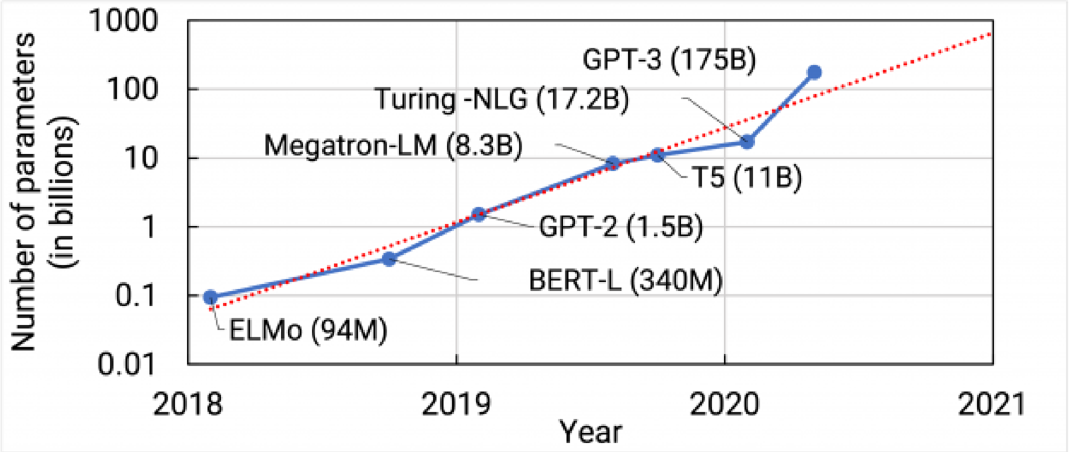
[Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing, Liu et al 2021]

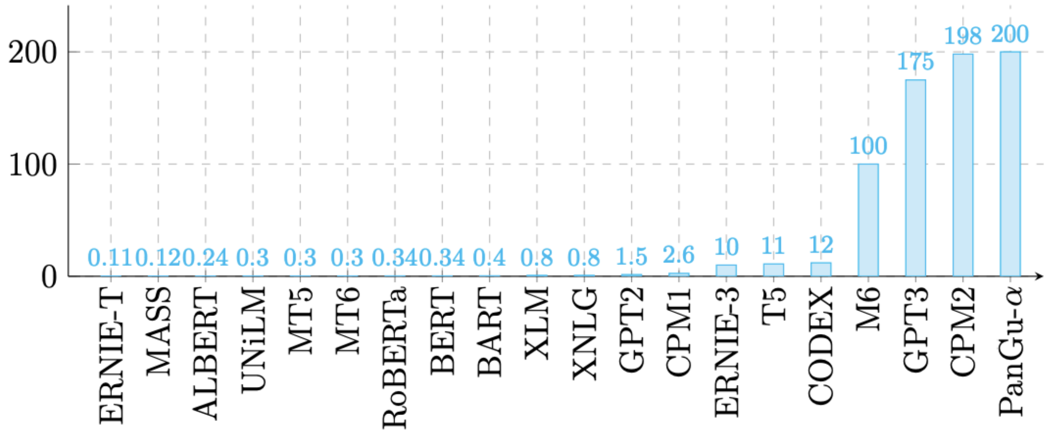
FOUR PARADIGMS IN NLP

Paradigm	Engineering	Task Relation
a. Fully Supervised Learning (Non-Neural Network)	Features (e.g. word identity, part-of-speech, sentence length)	
b. Fully Supervised Learning (Neural Network)	Architecture (e.g. convolutional, recurrent, self-attentional)	
c. Pre-train, Fine-tune	Objective (e.g. masked language modeling, next sentence prediction)	
d. Pre-train, Prompt, Predict	Prompt (e.g. cloze, prefix)	

[Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing, Liu et al 2021]

MODEL SIZE EXPONENTIAL GROWTH





[Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing, Liu et al 2021]

HUAWEI PANGU (200B)

- First, it *surpasses GPT-3 in few-shot learning tasks, addressing issues the latter faces in dealing with complex commercial scenarios with few (training data) samples.*
- Second, the Pangu team added prompt-based tasks in the pre-training phase, which greatly reduced the difficulty of fine-tuning.

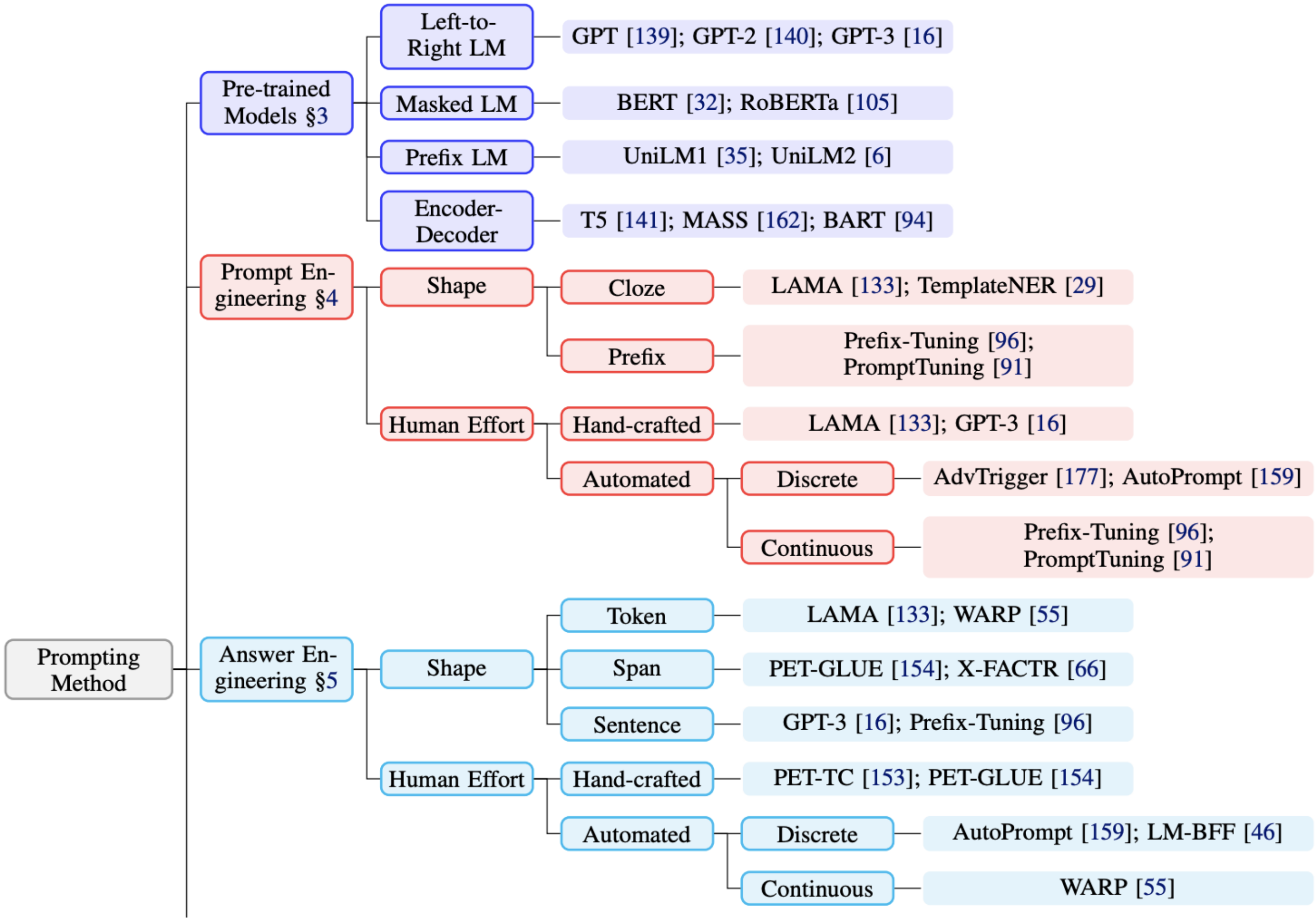
[GitHub: PCL-Platform.Intelligence/PanGu-Alpha: 2000亿开源中文预训练语言模型「鹏程·盘古α」 - PanGu-Alpha - OpenI]

NLP TASKS SOLVED BY PROMPTING

Type	Task	Input ([X])	Template	Answer ([Z])
Text CLS	Sentiment	I love this movie.	[X] The movie is [Z].	great fantastic ...
	Topics	He prompted the LM.	[X] The text is about [Z].	sports science ...
	Intention	What is taxi fare to Denver?	[X] The question is about [Z].	quantity city ...
Text-span CLS	Aspect Sentiment	Poor service but good food.	[X] What about service? [Z].	Bad Terrible ...
Text-pair CLS	NLI	[X1]: An old man with ... [X2]: A man walks ...	[X1]? [Z], [X2]	Yes No ...
Tagging	NER	[X1]: Mike went to Paris. [X2]: Paris	[X1][X2] is a [Z] entity.	organization location ...
Text Generation	Summarization	Las Vegas police ...	[X] TL;DR: [Z]	The victim ... A woman ... ...
	Translation	Je vous aime.	French: [X] English: [Z]	I love you. I fancy you. ...

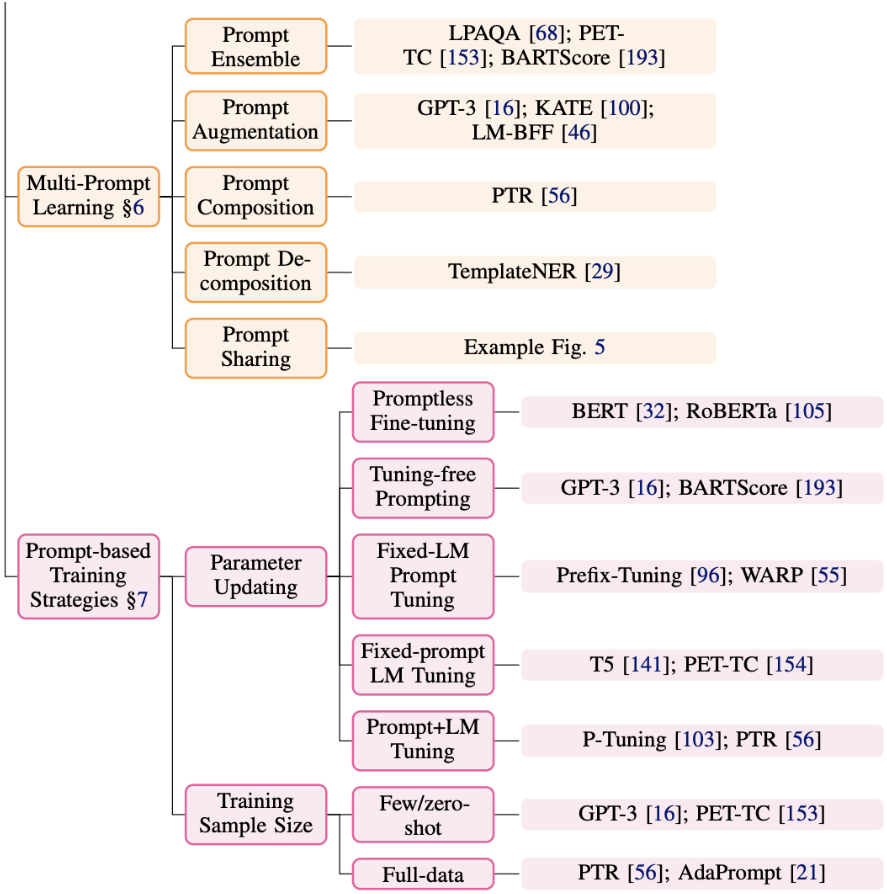
Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing, Liu et al 2021

PROMPTING METHOD



Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing, Liu et al 2021

PROMPTING METHOD



Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing, Liu et al 2021

P-TUNING

Trainable Continuous Prompt Embeddings

- Several issues with manually crafting templates: Creating and experimenting with these prompts takes time and experience. Even experienced prompt designers may fail to manually discover optimal prompts

- Handcraft prompt searching heavily relies on impractically large validations sets and its performance is volatile.

Prompt	P@1
[X] is located in [Y]. *(original)*	31.29
[X] is located in which country or state? [Y].	19.78
[X] is located in which country? [Y].	31.40
[X] is located in which country? In [Y].	51.08

- P-tuning regards individual prompt tokens as pseudo tokens and maps the template to include **trainable embedding tensors**

GPT Understands, Too, Liu et al 2021

P-TUNING

Trainable Continuous Prompt Embeddings

- The capital of _____ is _____ can be arranged as: $T = \{[\mathrm{P}_{0:i}], \mathbf{x}, [\mathrm{P}_{i+1:m}], \mathbf{y}\},$
- Traditional Discrete Prompt: $\{\mathbf{e}([\mathrm{P}_{0:i}]), \mathbf{e}(\mathbf{x}), \mathbf{e}([\mathrm{P}_{i+1:m}]), \mathbf{e}(\mathbf{y})\}$
- P-tuning instead regards individual prompt tokens as pseudo tokens and map the template to

$$\{h_0, ..., h_i, \mathbf{e}(\mathbf{x}), h_{i+1}, ..., h_m, \mathbf{e}(\mathbf{y})\}$$

where $h_i (0 \leq i < m)$ are **trainable embedding tensors.**

- With the downstream loss function L, we can optimize the **continuous prompt** by:

$$\hat{h}_{0:m} = \arg\min_{h} \mathcal{L}(\mathcal{M}(\mathbf{x}, \mathbf{y}))$$

GPT Understands, Too, Liu et al 2021

P-TUNING

Trainable Continuous Prompt Embeddings

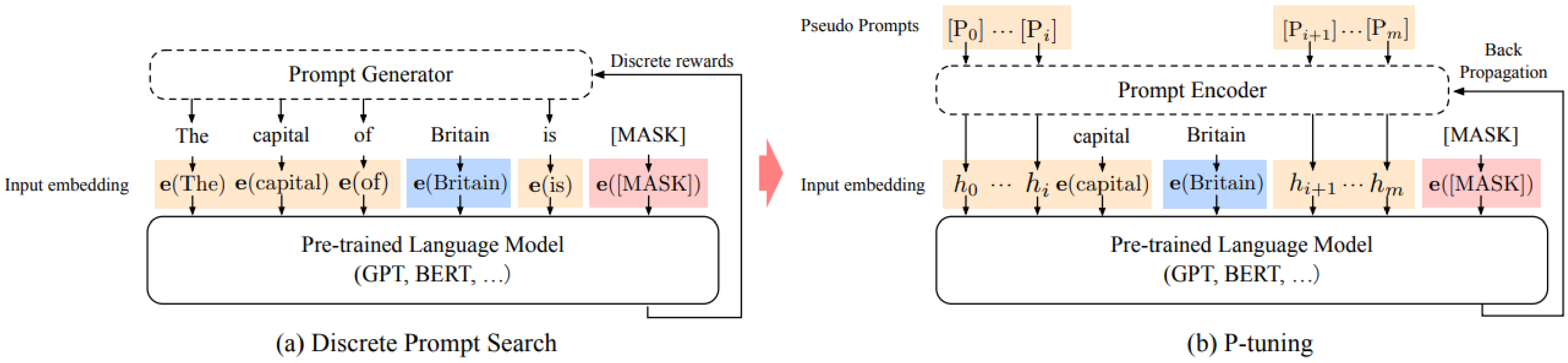


Figure 2. An example of prompt search for "The capital of Britain is [MASK]". Given the context (blue zone, "Britain") and target (red zone, "[MASK]"), the orange zone refer to the prompt tokens. In (a), the prompt generator only receives discrete rewards; on the contrary, in (b) the pseudo prompts and prompt encoder can be optimized in a differentiable way. Sometimes, adding few task-related anchor tokens (such as "capital" in (b)) will bring further improvement.

GPT Understands, Too, Liu et al 2021

P-TUNING

Knowledge probing Precision@1 on LAMA-34k (left) and LAMA-29k (right)

Prompt type	Model	P@1
Original (MP)	BERT-base	31.1
	BERT-large	32.3
	E-BERT	36.2
Discrete	LPAQA (BERT-base)	34.1
	LPAQA (BERT-large)	39.4
	AutoPrompt (BERT-base)	43.3
P-tuning	BERT-base	48.3
	BERT-large	**50.6**

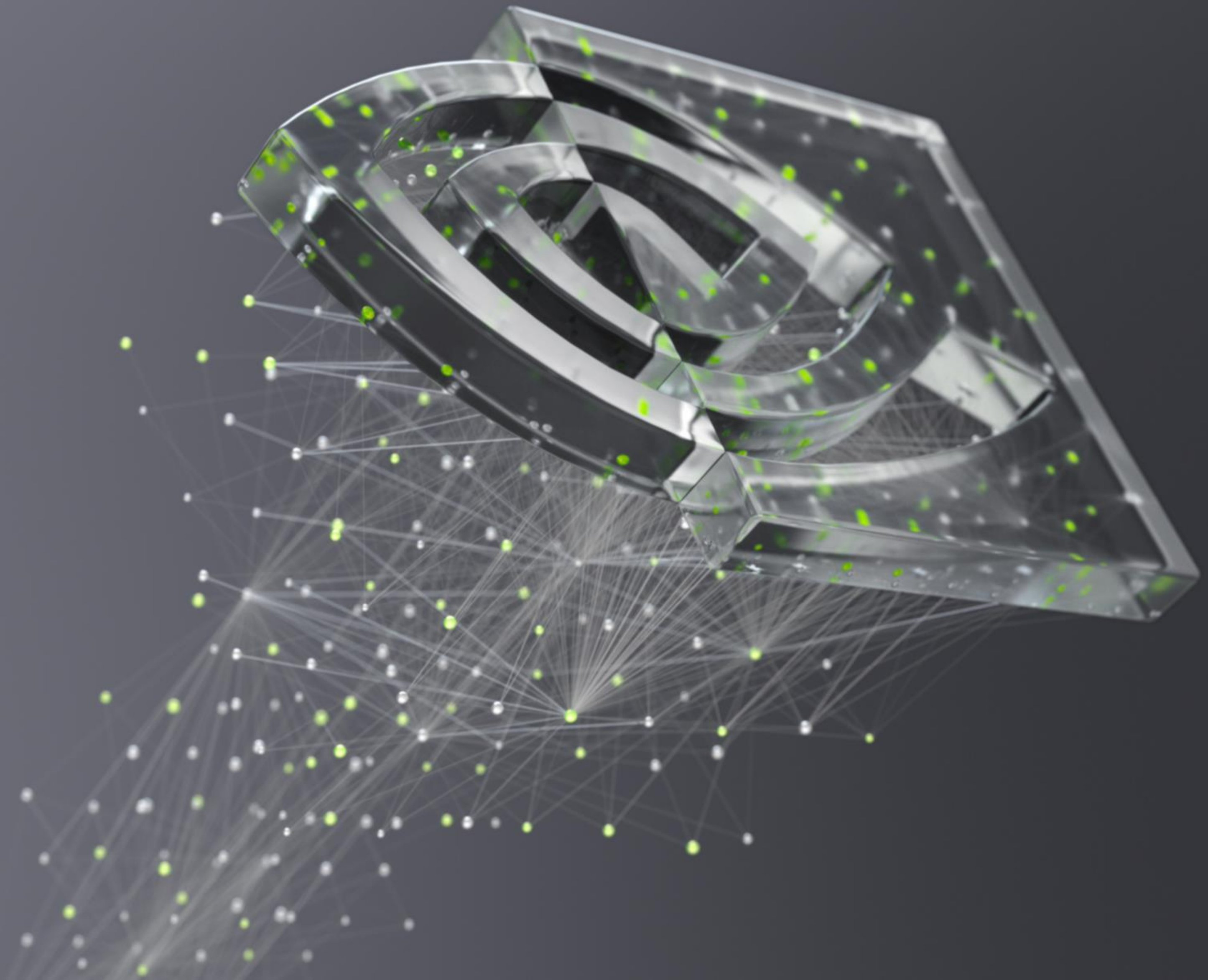
Model	MP	FT	MP+FT	P-tuning
BERT-base (109M)	31.7	51.6	52.1	52.3 (+20.6)
-AutoPrompt (Shin et al., 2020)	-	-	-	45.2
BERT-large (335M)	33.5	54.0	55.0	54.6 (+21.1)
RoBERTa-base (125M)	18.4	49.2	50.0	49.3 (+30.9)
-AutoPrompt (Shin et al., 2020)	-	-	-	40.0
RoBERTa-large (355M)	22.1	52.3	52.4	53.5 (+31.4)
GPT2-medium (345M)	20.3	41.9	38.2	46.5 (+26.2)
GPT2-xl (1.5B)	22.8	44.9	46.5	54.4 (+31.6)
MegatronLM (11B)	23.1	OOM*	OOM*	**64.2** (+41.1)

P-tuning outperforms all the discrete prompt searching baselines. And interestingly, despite fixed pre-trained model parameters, P-tuning overwhelms the fine-tuning GPTs in LAMA-29k. (MP: Manual prompt; FT: Fine-tuning; MP+FT: Manual prompt augmented fine-tuning; PT: P-tuning).

GPT Understands, Too, Liu et al 2021

DIRECTIONS TO TAKE

- How to make LM model truly a database
- Select, Insert, Update, Delete (SIUD)
- Megatron-GPT3 for Prompt Engineering
- GPT application for automating data science work
- Human interface -> data science code



Indic Machine Translation

Scaling NLP for non-English tasks

Ashish Sardana, Deep Learning Engineer (asardana@nvidia.com)

27th Oct 2021

Agenda

- Challenges
- Training experiment
- Demo

CHALLENGES

Indic language brings unique hurdles

- Low Resource
- 22 widely spoken among total 198 regional languages
- Missing standardized lexicon set
- Relatively large character set
- Language specific challenges

Most Indian languages have distinct representations in their orthography for voiced and unvoiced sounds. However, this is not the case with Tamil, which does not have distinct letters for voiced and unvoiced stops. There are well defined rules for predicting voicing in Tamil. For example, the voiceless stop [p] occurs at the beginning of words, while the voiced stop [b] does not.

(Ramakrishnan and Laxmi Narayana, 2007) describes a frontend for Tamil with rules for predicting voicing, similar to those described below. They also use a lexicon for foreign words of Sanskrit and Urdu origin, which do not follow these rules.

The rules that we implemented for Tamil voicing are taken from (Albert and others, 1985) and are as follows:

Indo-Aryan languages such as Hindi, Bengali, Gujarati, et cetera, exhibit a phenomenon known as *schwa deletion*, in which a final or medial schwa is deleted from a word in certain cases. For example, in Hindi, the final schwa (realized as the sound [ə]) in the word कमल (pronounced 'kamal') is deleted. None of the consonants क, म, or ल have an attached vowel; hence, they have inherent schwas, and the inherent schwa on the last consonant ल gets deleted. The word लगभग (pronounced 'lagbhag') has consonants ल ग भ ग, from which both the medial schwa on the first consonant ग and the final schwa on the second consonant ग get deleted. If schwa deletion did not take place, these words would erroneously be pronounced as 'kamala' and 'lagabhaga' respectively. In both these cases, the orthography does not indicate which inherent schwas should be deleted.

Prosody and lexical stress have not been well studied in Indian languages. A technique for automatically identifying stress based on power, energy, and duration by clustering units is described in (Laxmi Narayana and Ramakrishnan, 2007). Experiments were carried out on Tamil for syllable-level lexical stress, based on which a rule was created for assigning stress in Tamil as follows: The first syllable is stressed if it does not contain a short vowel; otherwise, the second syllable is stressed.

The *halant* character under a consonant indicates that a schwa is deleted, so we remove schwas after consonants that have this character under them.

We handle consonants with nukta characters under them by mapping them to the consonant without the nukta, as these characters are usually very rare in our training corpora.

TRAINING EXPERIMENT

- Dataset

  Samanantar consists of 8.6M pairs between En->Hi

- Architecture

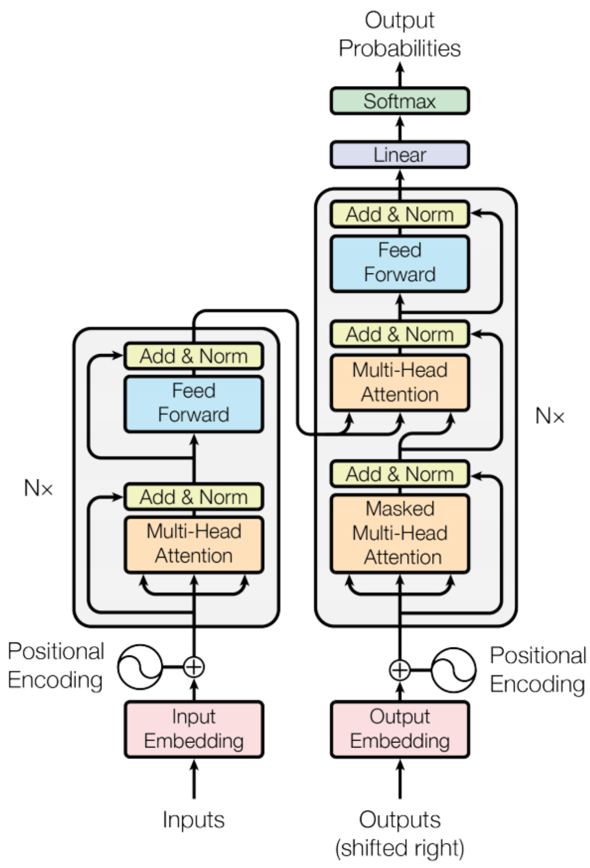
  Attention is all you need, Megatron BERT 345M & Megatron BERT 3.9B

- Pre-processing

  Length filtering (<1000 words), text normalization and lower-casing

- Results

  State-of-the-art (as of 10/27/2021)



TOKENIZER

English Tokenizers	Hindi Tokenizers
Moses	IndicNLP
OpenNMT	iNLTK
SentencePiece	Moses
NLTK	OpenNMT
Gruut	CLTK

Sentence:

मिस्र और रोम में गुलामों के साथ बहुत बुरा सलूक किया जाता था ।

IndicNLP Tokenizer:

['मिस्र', 'और', 'रोम', 'में', 'गुलामों', 'के', 'साथ', 'बहुत', 'बुरा', 'सलूक', 'किया', 'जाता', 'था', '।', '\n’]

iNLTK Tokenizer:

['मिस्र', 'और', 'रोम', 'में', 'गुलाम', 'ों', 'के', 'साथ', 'बहुत', 'बुरा', 'सल', 'ूक', 'किया', 'जाता', 'था', '', '।']

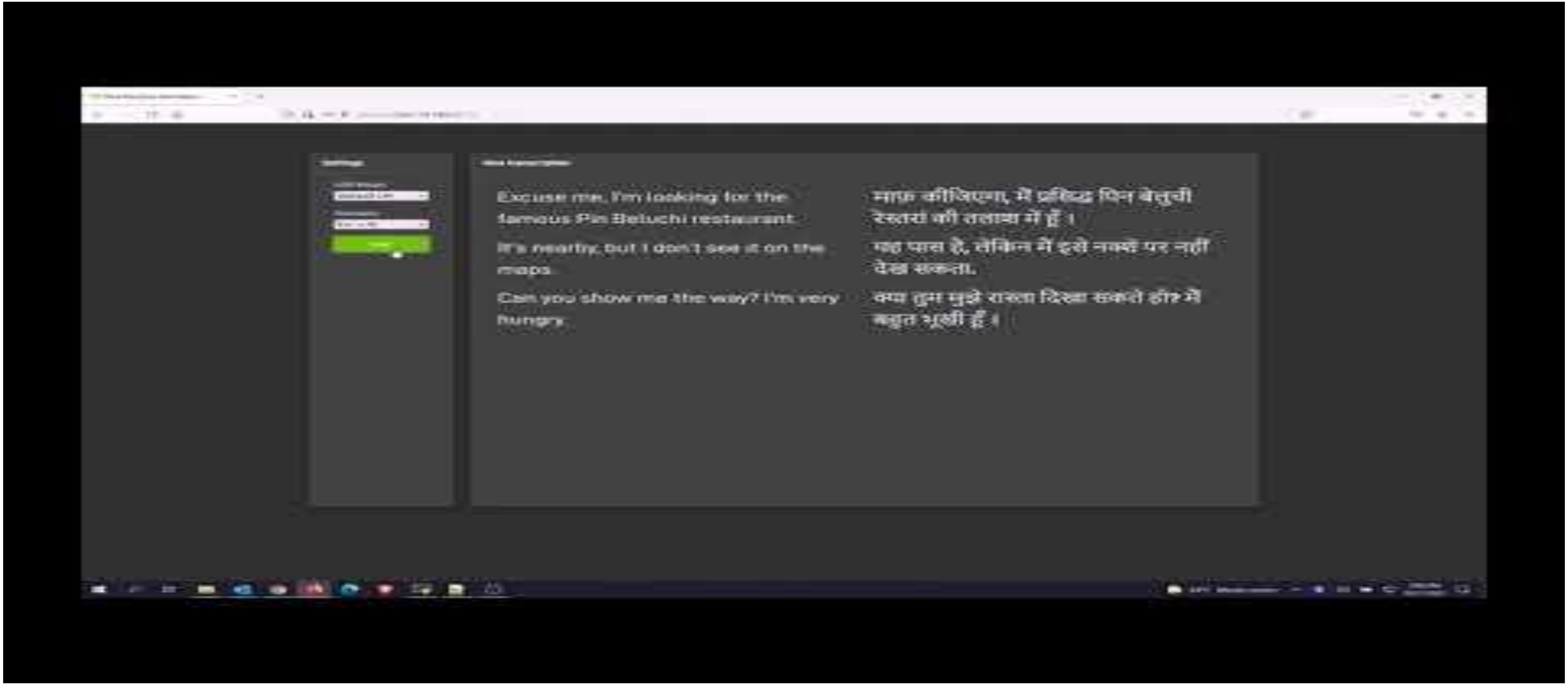
Sl. No	Model Name	Logs	English Tokenizer	Hindi Tokenizer	GPUs	No of Steps	Batch Size	Beam Size	Length Penalty	Training Loss	Validation Loss	sacreBLEU (val)	Rank
1	model_1	Logs	Moses	IndicNLP	8	85,000	12,500	4	0.60	2.633	1.528	32.134	
2	model_2	Logs	Moses	Moses	8	85,000	12,500	4	0.60	2.159	1.230	35.068	
2a	model_2a	Logs	Moses	Moses	8	85,000	12,500	3	0.60	2.377	1.232	34.915	
2b	model_2b	Logs	**Moses**	**Moses**	8	85,000	12,500	5	0.60	**2.144**	**1.222**	**35.289**	🥇
2c	model_2c	Logs	**Moses**	**Moses**	8	85,000	12,500	6	0.60	**2.501**	**1.227**	**35.255**	🥈
2d	model_2d	Logs	Moses	Moses	8	85,000	12,500	4	0.50	2.285	1.235	34.854	
2e	model_2e	Logs	Moses	Moses	8	85,000	12,500	4	0.70	2.552	1.229	35.096	
2f	model_2f	Logs	Moses	Moses	8	85,000	12,500	10	0.60	2.368	1.227	35.347	
3	model_3	Logs	Gruut	IndicNLP	8	85,000	12,500	4	0.60	2.560	1.620	28.526	
4	model_4	Logs	OpenNMT	CLTK	8	85,000	12,500	4	0.60	2.694	1.585	31.073	
5	model_5	Logs	Moses	OpenNMT	8	85,000	12,500	4	0.60	2.551	1.546	32.313	
6	model_6	Logs	Moses	CLTK	8	82,649	12,500	4	0.60	2.812	1.586	30.931	
7	model_7	Logs	OpenNMT	IndicNLP	8	85,000	12,500	4	0.60	2.516	1.531	32.130	
8	model_8	Logs	**OpenNMT**	**Moses**	8	85,000	12,500	4	0.60	**2.507**	**1.242**	**35.107**	🥉
9	model_9	Logs	NLTK	CLTK	8	85,000	12,500	4	0.60	2.555	1.585	30.950	
10	model_10	Logs	SentencePiece	IndicNLP	8	85,000	12,500	4	0.60	2.660	1.530	31.908	
11	model_11	Logs	SentencePiece	Moses	8	85,000	12,500	4	0.60	2.379	1.227	35.066	
12	model_12	Logs	NLTK	IndicNLP	8	85,000	12,500	4	0.60	2.933	1.529	31.973	
13	model_13	Logs	OpenNMT	OpenNMT	8	85,000	12,500	4	0.60	2.578	1.544	32.618	
14	model_14	Logs	SentencePiece	OpenNMT	8	61,599	12,500	4	0.60	2.547	1.581	32.486	
15	model_15	Logs	SentencePiece	CLTK	8	85,000	12,500	4	0.60	2.698	1.586	30.706	
16	model_16	Logs	NLTK	Moses	8	85,000	12,500	4	0.60	2.388	1.225	34.960	
17	model_17	Logs	NLTK	OpenNMT	8	85,000	12,500	4	0.60	2.582	1.545	33.021	
18	model_18	Logs(18,20)	Gruut	Moses	8	85,000	12,500	4	0.60	2.359	1.285	31.737	
19	model_19	Logs	Gruut	OpenNMT	8	85,000	12,500	4	0.60	3.066	1.629	29.324	
20	model_20	Logs(18,20)	Gruut	CLTK	8	85,000	12,500	4	0.60	3.051	1.672	27.537	
21	model_21	Logs	Moses	iNLTK	8	85,000	12,500	4	0.60	2.552	1.483	32.712	
22	model_22	Logs	OpenNMT	iNLTK	8	85,000	12,500	4	0.60	2.566	1.489	32.734	
23	model_23	Logs	NLTK	iNLTK	8	85,000	12,500	4	0.60	2.563	1.488	32.708	
24	model_24	Logs	SentencePiece	iNLTK	8	85,000	12,500	4	0.60	2.615	1.484	32.731	
25	model_25	Logs	Gruut	iNLTK	8	85,000	12,500	4	0.60	2.485	1.568	29.111	
26	model_ckpt		Moses	Moses	8	85,000	12,500	3	0.60				

RESULTS

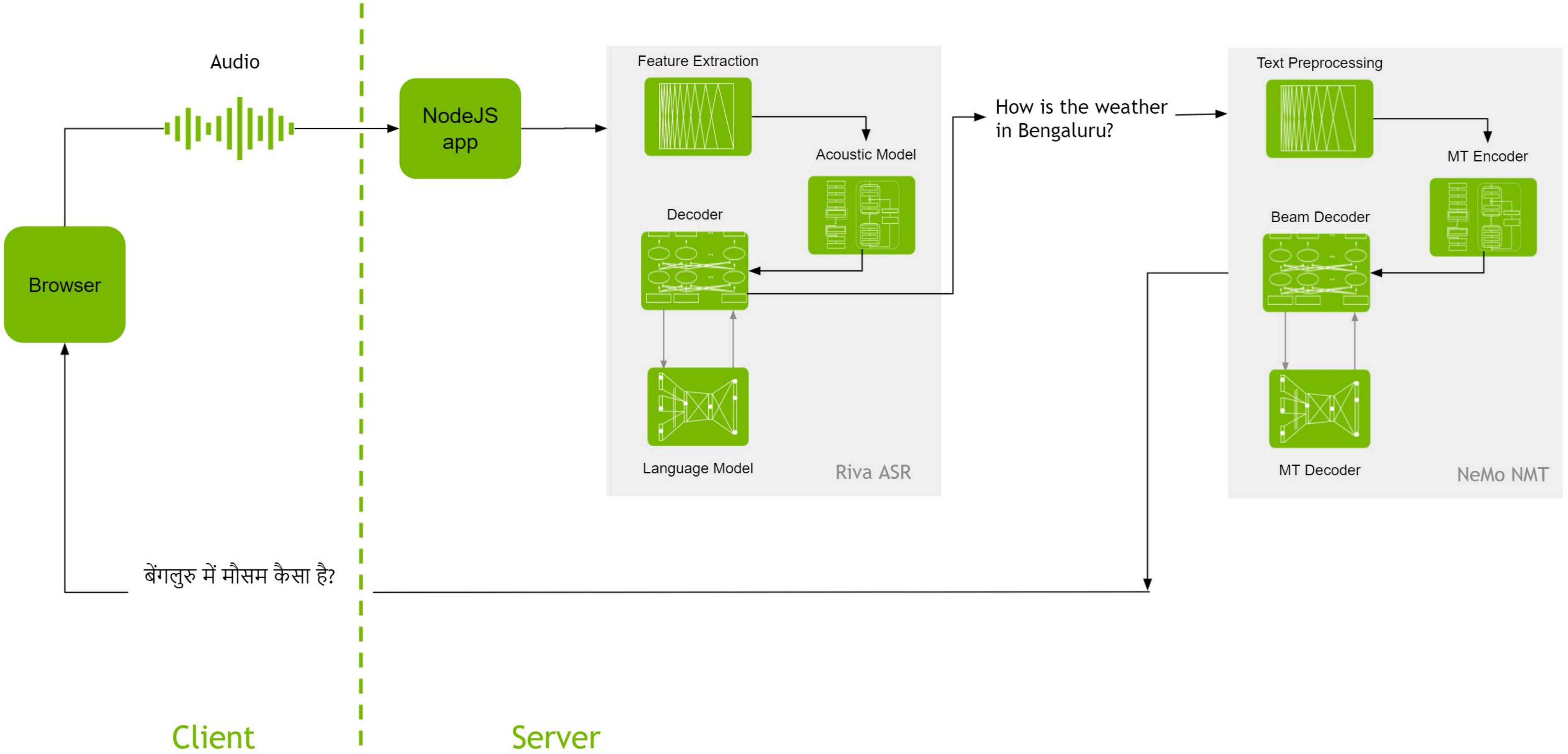
Achieved state-of-the-art

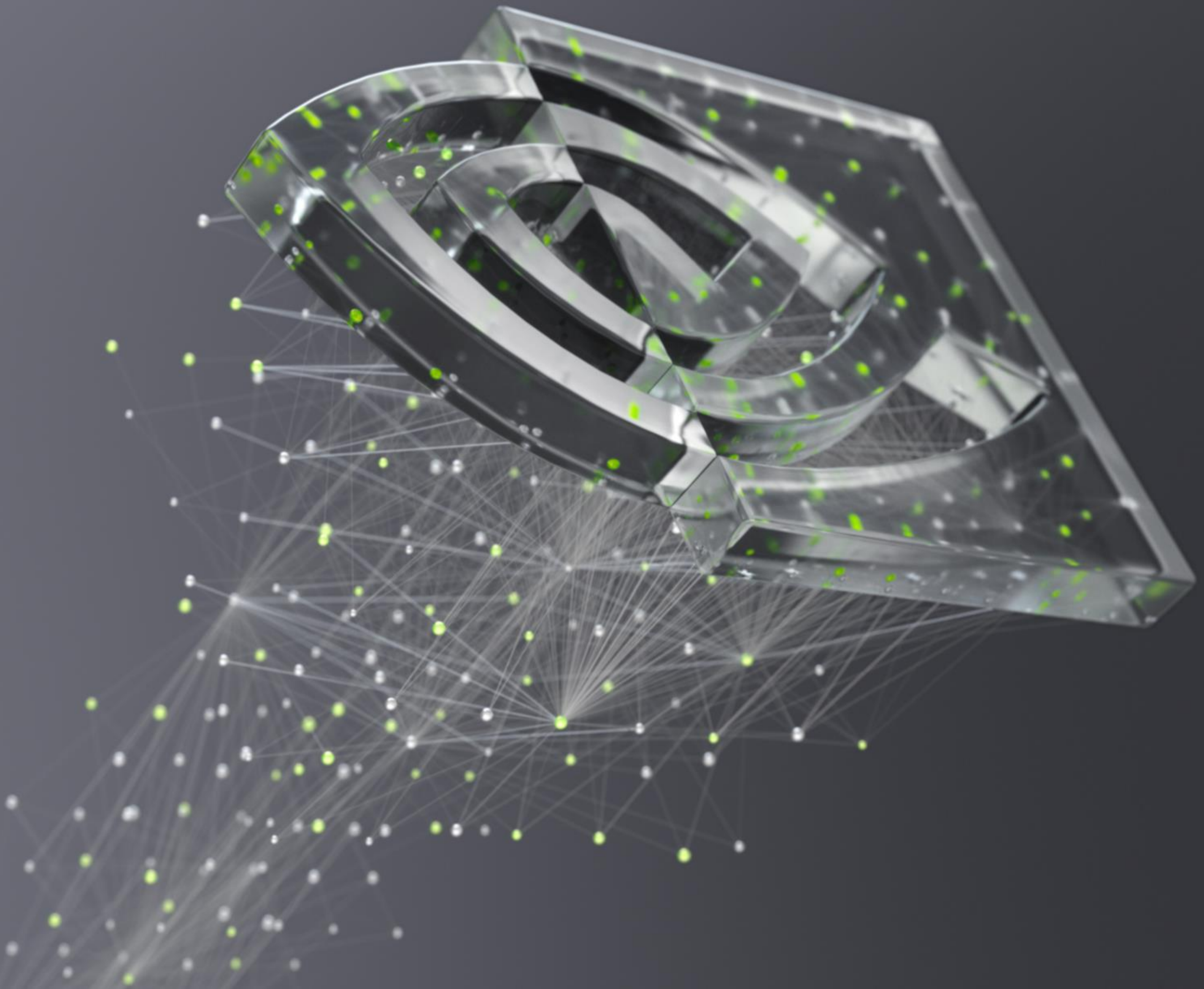
Model	English Tokenizer	Hindi Tokenizer	Beam Size	Length Penalty	Batch Size	No. of Steps	sacredBLEU WAT2020	sacredBLEU WAT 2021	sacredBLEU WMT
NVIDIA-Megatron BERT 3.9B	Moses	Moses	4	0.6	3,125	6M	**52.2**	**63.6**	**59.8**
NVIDIA-Megatron BERT 345M	Moses	Moses	4	0.6	6,250	3M	49.7	59.4	55.1
NVIDIA-AAYN	Moses	Moses	4	0.6	12,500	2.12M	46.2	56.8	52.9
IndicTrans (SoTa)	Moses	Moses	5	-	12,500	85k	19.4	37.9	25.0
GCP MT	-	-	-	-	-	-	22.6	36.7	31.3
Azure MT	-	-	-	-	--	-	21.3	38	30.1

DEMO APPLICATION



ARCHITECTURE





THANK YOU!

~ QUESTIONS?

NVIDIA