COV884: Advanced Topic in NLP

Multilingual Pre-training

Instructor - Prof. Mausam

Rocktim Jyoti Das-2018EE10493
March 23,2022

Overview

- Why Multilingual?
- Revisiting Transformer Architecture.
- Tokenization: WordPiece, SentencePiece.
- Pretraining Data.
- Some common Training Objectives.
- Benchmarks for Evaluation of Multilingual Language Models.
- BERT & mBERT.
- T5 and mT5.
- Review of Students.
- Conclusion

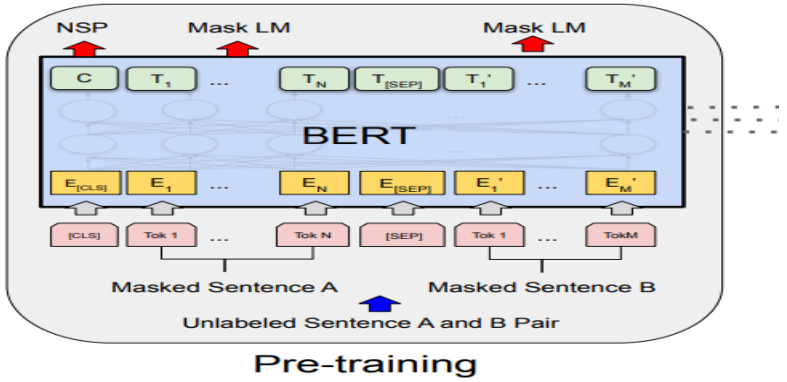
Why Multilingual?

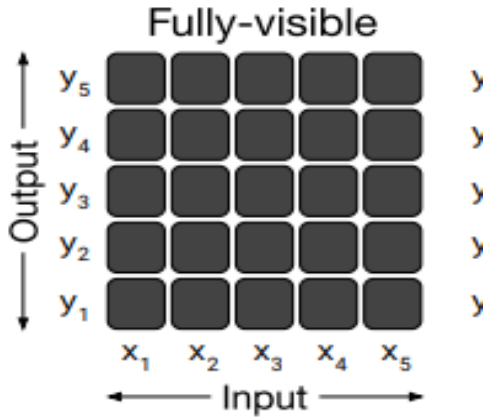
- Usefulness of BERT like model that are trained on large amount of data in an unsupervised manner which results in encoder which learns good sentence representations.
- Language specific models like FlauBERT, CamemBERT, BERTje are feasible for few languages which have the necessary data.
- This motivates training of Multilingual Language Models to enable transfer from high resource language to low resource language.
- Ability of MLLMs to facilitate zero-shot cross lingual transfer.

Revisiting Transformer Architecture

Common types of Transformer Architecture:

- Transformer Encoder Model like BERT.

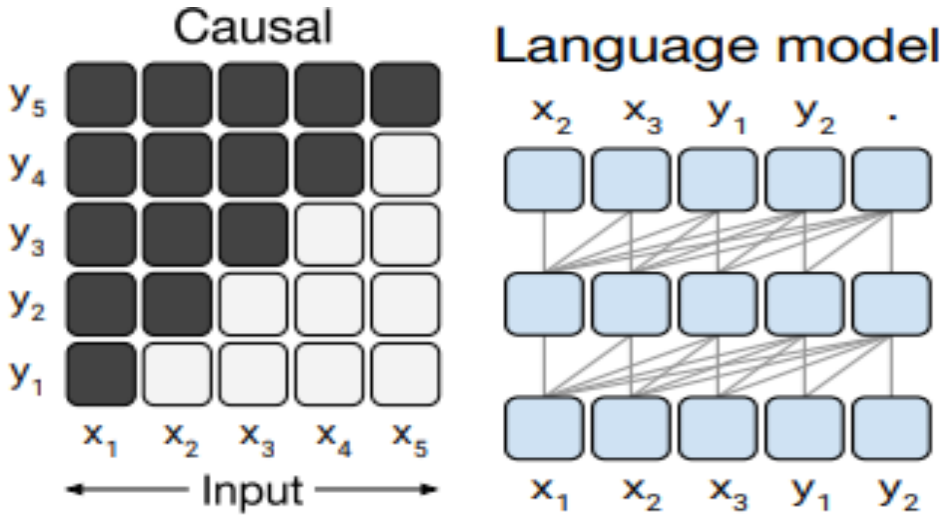




- Useful for getting token representation to solve downstream NLP tasks like NER and POS.
- Not used for Generation
- Training objective is predicting masked tokens.

Revisiting Transformer Architecture

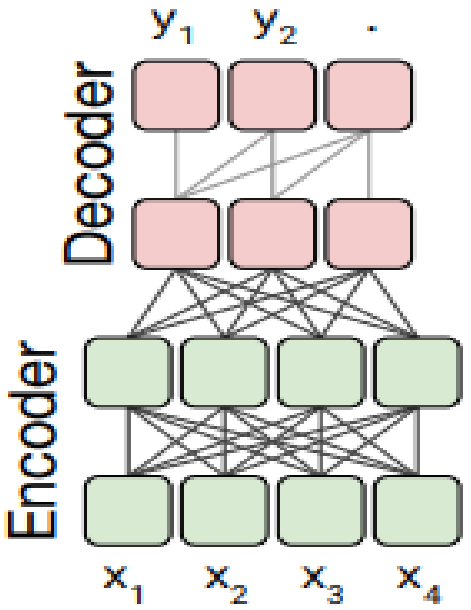
- Transformer Decoder Model like GPT2 and variants of T5.
  - Trained via LM objective.
  - The Attention head are masked.
  - Useful for generation task.



Revisiting Transformer Architecture

- Transformer Encoder-Decoder Model like T5.
  - Useful for Conditional Text Generation.
  - The encoder has unmasked self attention
  - The decoder has causal masked self attention
  - Training objective like predicting masked tokens

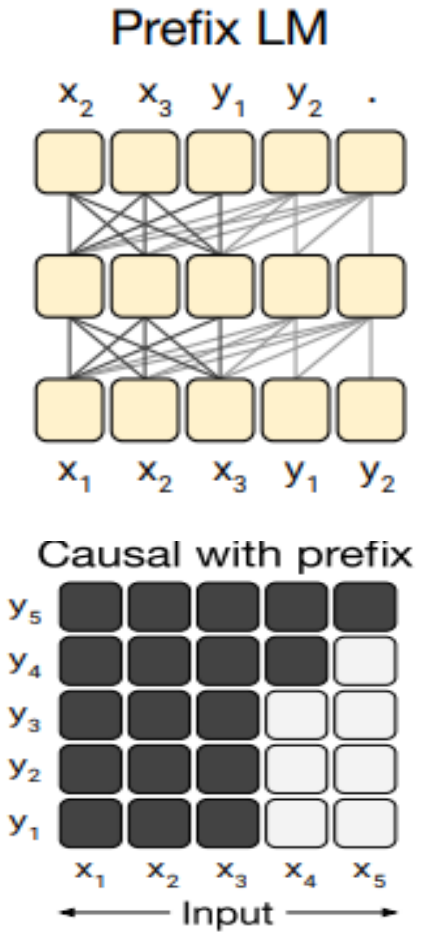
  Is used for these models.



Revisiting Transformer Architecture

- Prefix LM.
  - Useful for Conditional Text Generation.
  - Alternative to encoder/decoder approach

  With just one model instead of 2.



Tokenization

- Tokenization is the very first preprocessing step for any input data as models cannot process sentences or words.
- All of tokenizers used in MLLMs uses some form of subword tokenization.
- Subword because words result in a very large word vocabulary and a lot of unknown words are encountered at test time. Also Word-level tokenization treats different forms of the same word as different. Eg: look, looking, looks.
- There are three main type of tokenizers used in MLLMs:
  - Byte-Pair Encoding
  - Word Piece
  - Sentence Piece

Tokenization: Byte Pair Encoding

- Form Base vocabulary (all characters that occur in the training data)

word	frequency
hug	10
pug	5
pun	12
bun	4
hugs	5

- Base vocab: b,g,h,n,p,s,u

Tokenization: Byte Pair Encoding

- Now, count up the frequency of each character pair in the data and choose the one that occurs most frequently.

word	frequency
h+u+g	10
p+u+g	5
p+u+n	12
b+u+n	4
h+u+g+s	5

character pair	frequency
ug	20
pu	17
un	16
hu	15
gs	5

- Now , choose the most common pair (ug) and then merge the characters together into one symbol. Add this new symbol to the vocabulary.
- vocab: b,g,h,n,p,s,u,ug

Tokenization: Byte Pair Encoding

- Retokenize the data and repeat the process

word	frequency
h+*ug*	10
p+*ug*	5
p+u+n	12
b+u+n	4
h+*ug*+s	5

character pair	frequency
un	16
h+ug	15
pu	12
p+ug	5
ug+s	5

- vocab: b,g,h,n,p,s,u,ug,un

Tokenization: Byte Pair Encoding

- Eventually, after a fixed number of merge steps, we stop.

word	frequency
hug	10
p+*ug*	5
p+*un*	12
b+*un*	4
hug + s	5

- vocab: b,g,h,n,p,s,u,ug,un,hug

Tokenization: WordPiece and SentencePiece

- WordPiece is also similar to BPE but instead of choosing the most frequent symbol pair, chose the one that maximizes the likelihood of the training data once added to the vocabulary.
- All Other tokenization has an issue that they assume the input text uses spaces to separate words.
- SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing treats the input as a raw input stream, thus including the space in the set of characters to use. It then uses the BPE algorithm to construct the appropriate vocabulary.

Pretrained Data for Multilingual Language Model

Two different sources of Data are used:

- Large monolingual corpora in individual languages
- Parallel corpora between some languages.

Examples:

- mBERT is pretrained on the concatenation of monolingual Wikipedia corpora from 104 languages.
- Ernie-M is pretrained on Common Crawl data which has parallel corpora.

Training Objective Function for Multilingual Language Model

Training Objective Functions can be broadly classified into two categories:

- Monolingual Objectives.

These objective functions are defined on monolingual data alone.

  - Masked Language Model:
    - Random k tokens are masked and the goal is to predict the mask token using the remaining tokens.
  - Causal Language Model:
    - This is the traditional language modelling objective of predicting the next word given the previous words.

Training Objective Function for Multilingual Language Model

- Parallel Objectives.
  - These objectives require parallel corpora and are designed to explicitly force representations of similar text across languages to be close to each other in multilingual encoder space.
  - Parallel corpus is generally much smaller than monlingual data, so, parallel objectives are used in conjunction with monolingual objectives, with each objective weighted appropriately.
  - One example is Translation Language Model:
    - Here two sequences <x1,x2,x3,x4,……,xn>and <y1,y2,y3,y4,……,yn>, of language A and B respectively, are fed as input to the Multilingual Language Model with a [SEP] token in between.
    - K tokens are masked and the goal is to predict a masked word in language A relying on surrounding words in A or the translation in B.

Comparison of Existing Multilingual Language Model

Model	Architecture				pretraining				Languages	
	N	*k*	*d*	*#Params.*	*Objective Function*	*Mono.*	*Parallel*	*Task specific data*	*#langs.*	*vocab.*
IndicBERT (Kakwani et al., 2020)	12	12	768	33M	MLM	IndicCorp	✗	✗	12	200K
Unicoder (Huang et al., 2019)	12	16	1024	250M	MLM, TLM, CLWR, CLPC, CLMLM	Wikipedia	✓	✗	15	95K
XLM-15 (Conneau and Lample, 2019)	12	8	1024	250M	MLM, TLM	Wikipedia	✓	✗	15	95K
XLM-17 (Conneau and Lample, 2019)	16	16	1280	570M	MLM	Wikipedia	✓	✗	17	200K
MuRIL (Khanuja et al., 2021a)	12	12	768	236M	MLM, TLM	CommonCrawl + Wikipedia	✓	✗	17	197K
VECO-small (Luo et al., 2021)	6	12	768	247M	MLM, CS-MLM†	CommonCrawl	✓	✗	50	250K
VECO-Large (Luo et al., 2021)	24	16	1024	662M	MLM, CS-MLM	CommonCrawl	✓	✗	50	250K
XLM-align (Chi et al., 2021b)	12	12	768	270M	MLM, TLM, DWA	CommonCrawl + Wikipedia	✓	✗	94	250K
InfoXLM-base (Chi et al., 2021a)	12	12	768	270M	MLM, TLM, XLCO	CommonCrawl	✓	✗	94	250K
InfoXLM-Large (Chi et al., 2021a)	24	16	1024	559M	MLM, TLM, XLCO	CommonCrawl	✓	✗	94	250K
XLM-100 (Conneau and Lample, 2019)	16	16	1280	570M	MLM	Wikipedia	✗	✗	100	200K
XLM-R-base (Conneau et al., 2020a)	12	12	768	270M	MLM	CommonCrawl	✗	✗	100	250K
XLM-R-Large (Conneau et al., 2020a)	24	16	1024	559M	MLM	CommonCrawl	✗	✗	100	250K
X-STILTS (Phang et al., 2020)	24	16	1024	559M	MLM	CommonCrawl	✗	✓	100	250K
HiCTL-base (Wei et al., 2021)	12	12	768	270M	MLM, TLM, HICTL	CommonCrawl	✓	✗	100	250K
HiCTL-Large (Wei et al., 2021)	24	16	1024	559M	MLM, TLM, HICTL	CommonCrawl	✓	✗	100	250K
Ernie-M-base (Ouyang et al., 2021)	12	12	768	270M	MLM, TLM, CAMLM, BTMLM	CommonCrawl	✓	✗	100	250K
Ernie-M-Large (Ouyang et al., 2021)	24	16	1024	559M	MLM, TLM, CAMLM, BTMLM	CommonCrawl	✓	✗	100	250K
XLM-E (Chi et al., 2021c)	12	12	768	279M	MLM, TLM, MRTD, TRTD	CommonCrawl	✓	✗	100	250k
mBERT (Devlin et al., 2019)	12	12	768	172M	MLM	Wikipedia	✗	✗	104	110K
Amber (Hu et al., 2021)	12	12	768	172M	MLM, TLM, CLWA, CLSA	Wikipedia	✓	✗	104	120K
RemBERT (Chung et al., 2021a)	32	18	1152	559M‡	MLM	CommonCrawl + Wikipedia	✗	✗	110	250K

Table 1: A comparison of existing Multilingual Language Models. † - Cross sequence MLM which is useful for NLG tasks. ‡ - For pretraining, RemBERT uses 995M parameters

Benchmarks for evaluating Multilingual Language Model

- Common Evaluation: Finetune the model on task-specific data for a high resource language like English and evaluate on other languages.
- Evaluation benchmark contains NLP task which can be classification, structure prediction, question answering.
- **Classification**: Classification task are NLI task and some of the popular datasets used are: XNLI and PAWS-X.
- **Structure Prediction**: These task require predicting a label for every word in the sequence. Two popular task here are Parts of Speech(POS) tagging and Named Entity Recognition(NER).

Benchmarks for evaluating Multilingual Language Model

- **Question Answering**:
  - Here the task is to extract an answer span given a context and a question.
  - The training data is typically available only in English while the evaluation sets are available in multiple languages
  - The datasets used for this task include XQuAD , MLQA and TyDiQA

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

And

How multilingual is Multilingual BERT?

BERT: Encoder Transformer

- Architecture: BERT's model architecture is a multi-layer bidirectional Transformer encoder.
- Dataset: For the pre-training corpus, the BooksCorpus (800M words) and English Wikipedia (2,500M words) are used.
- Vocabulary: WordPiece embeddings with a 30,000 token vocabulary.
- Training Objective:
  - Masked LM: 15 % tokens are masked and goal is to predict them.
  - Next Sentence Prediction (NSP): a binarized next sentence prediction task that can be trivially generated from any monolingual corpus.

mBERT: Encoder Transformer

- Architecture: Almost same as the BERT model.
- Dataset:
  - For the pre-training, it is trained on the Wikipedia pages of 104 languages with a shared word piece vocabulary.
  - exponentially smoothed weighting of the data during pre-training data creation.
- Vocabulary: For tokenization, we use a 110k shared WordPiece vocabulary.
- Training Objective: Same as BERT.

mBERT: NER and POS Experiments

Fine-tuning \ Eval	EN	DE	NL	ES
EN	**90.70**	69.74	77.36	73.59
DE	73.83	**82.00**	76.25	70.03
NL	65.46	65.68	**89.86**	72.10
ES	65.38	59.40	64.39	**87.18**

Table 1: NER F1 results on the CoNLL data.

Fine-tuning \ Eval	EN	DE	ES	IT
EN	**96.82**	89.40	85.91	91.60
DE	83.99	**93.99**	86.32	88.39
ES	81.64	88.87	**96.71**	93.71
IT	86.79	87.82	91.28	**98.11**

Table 2: POS accuracy on a subset of UD languages.

- NER experiments were performed on CoNLL dataset which contain English, German, Dutch and Spanish.
- For POS experiments were performed on Universal Dependencies (UD) data for 41 languages.
- We can observe that mBERT generalizes well across languages for structure prediction task.

mBERT: Vocabulary Memorization

- En-BERT performance depends

Greatly on the word-piece overlap.

- M-BERT's performance is flat

for a wide range of overlaps,

and even for language pairs with

almost no lexical overlap.

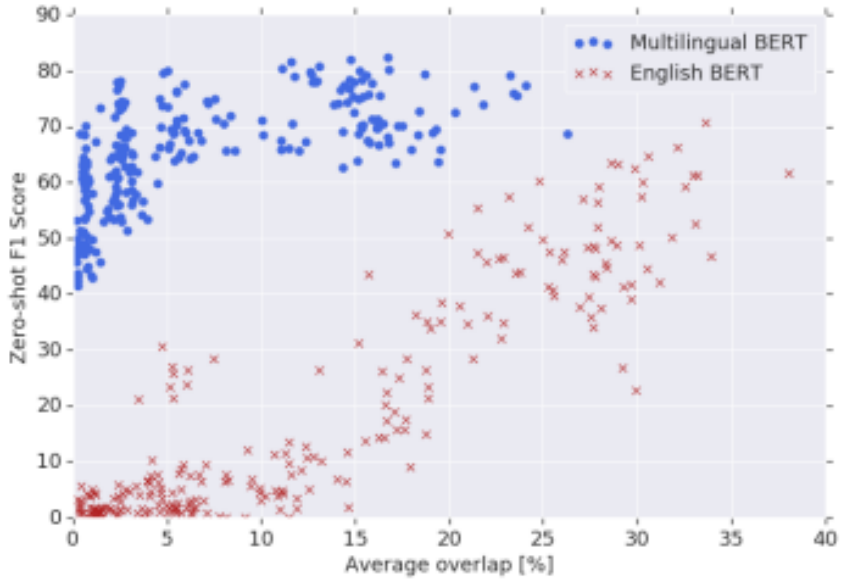


Figure 1: Zero-shot NER F1 score versus entity word piece overlap among 16 languages. While performance using EN-BERT depends directly on word piece overlap, M-BERT's performance is largely independent of overlap, indicating that it learns multilingual representations deeper than simple vocabulary memorization.

mBERT: Effect of Similarity of Linguistic Structure

- Both for En-BERT and m-BERT performance is best when transferring between languages that share word order feature.
- Thus though M-BERT's multilingual representation s able to map learned structures onto new vocabularies, it does not seem to learn systematic transformations of those structures to accommodate a target language with different word order.

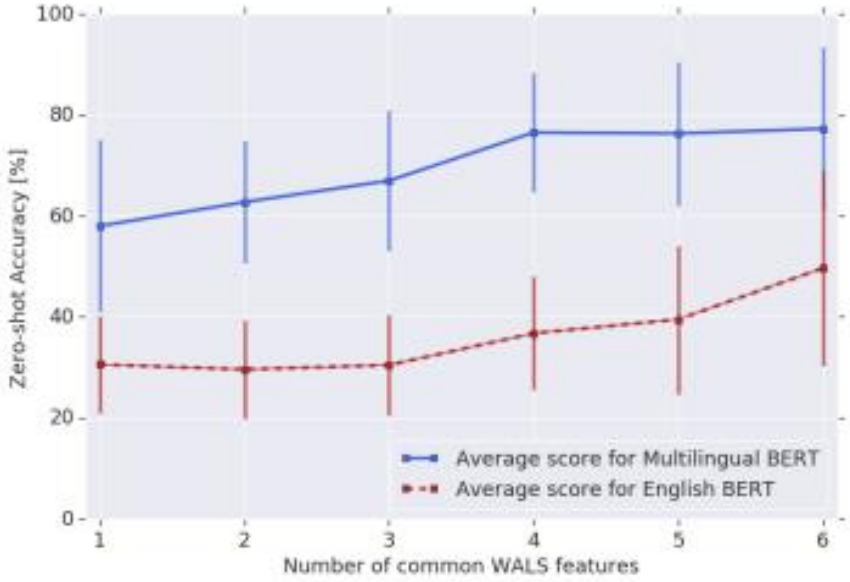


Figure 2: Zero-shot POS accuracy versus number of common WALS features. Due to their scarcity, we exclude pairs with no common features.

` https://wals.info/

Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

And

mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer

T5: Text to Text Transfer Transformer

- Architecture: Several architecture are tried out and finally encoder-decoder transformer was shown to perform best.
- Dataset:
  - Every NLP task is treated as a text-to-text task and “Colossal Clean Crawled Corpus”(C4), the common crawl based dataset was created as a source of unlabelled text data.
  - They defined some heuristics to filter out undesired text from the dataset and they ensured that the dataset only contains English data.
- Evaluation: Evaluation was performed on several downstream task like sentiment analysis, Natural Language Inference, Question Answering etc.

T5: Text to Text Transfer Transformer

- Input format:
  - To specify which task the model should perform, they add a task-specific (text) prefix to the original input sequence before feeding it to the model.
  - a consistent training objective both for pre-training and fine-tuning phase for multiple downstream task.
- Output format:
  - For text classification single word corresponding to the target label is generated.
  - For generative task output is sampled in an autoregressive manner.

T5: Text to Text Transfer Transformer

- Vocabulary: SentencePiece tokenizer was used. For all experiments, we used a vocabulary of 32000 wordpieces.
- Unsupervised Objective:
  - Inspired by BERT
  - Consecutive spans we

  Dropped out and replaced

  With sentinel tokens.
  - Goal is to predict and

  generate these

  Dropped out tokens.

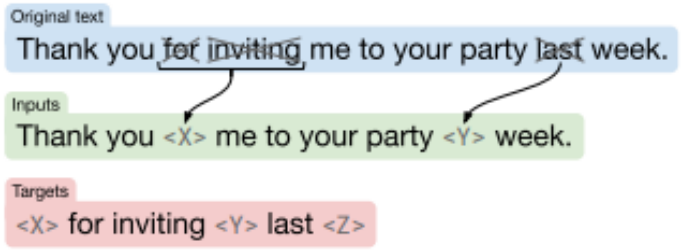


Figure 2: Schematic of the objective we use in our baseline model. In this example, we process the sentence "Thank you for inviting me to your party last week." The words "for", "inviting" and "last" (marked with an ×) are randomly chosen for corruption. Each consecutive span of corrupted tokens is replaced by a sentinel token (shown as **<X>** and **<Y>**) that is unique over the example. Since "for" and "inviting" occur consecutively, they are replaced by a single sentinel **<X>**. The output sequence then consists of the dropped-out spans, delimited by the sentinel tokens used to replace them in the input plus a final sentinel token **<Z>**.

T5: Text to Text Transfer Transformer

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76

Table 2: Performance of the different architectural variants described in Section 3.2.2. We use P to refer to the number of parameters in a 12-layer base Transformer layer stack and M to refer to the FLOPs required to process a sequence using the encoder-decoder model. We evaluate each architectural variant using a denoising objective (described in Section 3.1.4) and an autoregressive objective (as is commonly used to train language models).

mT5: C4 vs mC4

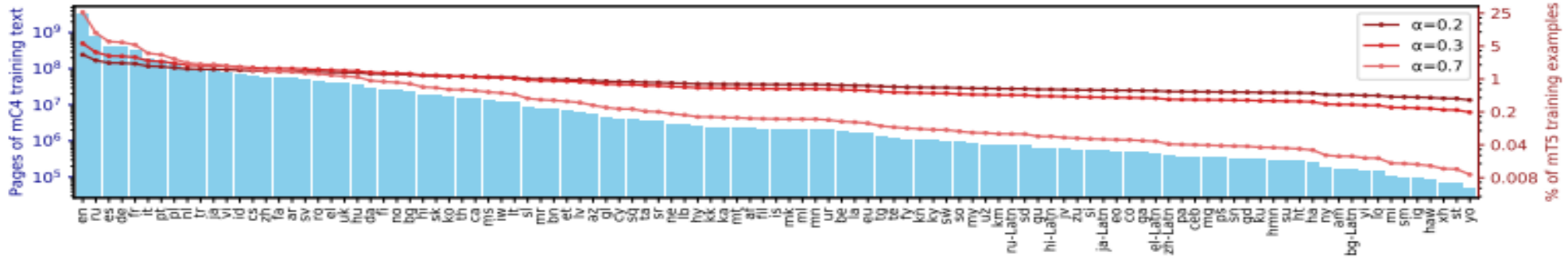


Figure 1: Page counts per language in mC4 (left axis), and percentage of mT5 training examples coming from each language, for different language sampling exponents α (right axis). Our final model uses α=0.3.

- C4 dataset was explicitly designed to be English only and langdetect to detect English language whereas mC4 used cld3 to identify over 100 languages.
- Exponential Sampling was used to boost the probability of training on low resource language.

mT5: Comparison of mT5 with other multilingual model

Model	Architecture	Parameters	# languages	Data source
mBERT (Devlin, 2018)	Encoder-only	180M	104	Wikipedia
XLM (Conneau and Lample, 2019)	Encoder-only	570M	100	Wikipedia
XLM-R (Conneau et al., 2020)	Encoder-only	270M – 550M	100	Common Crawl (CCNet)
mBART (Lewis et al., 2020b)	Encoder-decoder	680M	25	Common Crawl (CC25)
MARGE (Lewis et al., 2020a)	Encoder-decoder	960M	26	Wikipedia or CC-News
mT5 (ours)	Encoder-decoder	300M – 13B	101	Common Crawl (mC4)

Table 1: Comparison of mT5 to existing massively multilingual pre-trained language models. Multiple versions of XLM and mBERT exist; we refer here to the ones that cover the most languages. Note that XLM-R counts five Romanized variants as separate languages, while we ignore six Romanized variants in the mT5 language count.

mT5: Benchmark Comparison of mT5 with other models

Model	Sentence pair		Structured	Question answering		
	XNLI	PAWS-X	WikiAnn NER	XQuAD	MLQA	TyDiQA-GoldP
Metrics	Acc.	Acc.	F1	F1 / EM	F1 / EM	F1 / EM
Cross-lingual zero-shot transfer (models fine-tuned on English data only)						
mBERT	65.4	81.9	62.2	64.5 / 49.4	61.4 / 44.2	59.7 / 43.9
XLM	69.1	80.9	61.2	59.8 / 44.3	48.5 / 32.6	43.6 / 29.1
InfoXLM	81.4	-	-	- / -	73.6 / 55.2	- / -
X-STILTs	80.4	87.7	64.7	77.2 / 61.3	72.3 / 53.5	76.0 / 59.5
XLM-R	79.2	86.4	65.4	76.6 / 60.8	71.6 / 53.2	65.1 / 45.0
VECO	79.9	88.7	65.7	77.3 / 61.8	71.7 / 53.2	67.6 / 49.1
RemBERT	80.8	87.5	**70.1**	79.6 / 64.0	73.1 / 55.0	77.0 / 63.0
mT5-Small	67.5	82.4	50.5	58.1 / 42.5	54.6 / 37.1	35.2 / 23.2
mT5-Base	75.4	86.4	55.7	67.0 / 49.0	64.6 / 45.0	57.2 / 41.2
mT5-Large	81.1	88.9	58.5	77.8 / 61.5	71.2 / 51.7	69.9 / 52.2
mT5-XL	82.9	89.6	65.5	79.5 / 63.6	73.5 / 54.5	75.9 / 59.4
mT5-XXL	**85.0**	**90.0**	69.2	**82.5 / 66.8**	**76.0 / 57.4**	**80.8 / 65.9**
Translate-train (models fine-tuned on English data plus translations in all target languages)						
XLM-R	82.6	90.4	-	80.2 / 65.9	72.8 / 54.3	66.5 / 47.7
FILTER + Self-Teaching	83.9	91.4	-	82.4 / 68.0	76.2 / 57.7	68.3 / 50.9
VECO	83.0	91.1	-	79.9 / 66.3	73.1 / 54.9	75.0 / 58.9
mT5-Small	64.7	79.9	-	64.3 / 49.5	56.6 / 38.8	48.2 / 34.0
mT5-Base	75.9	89.3	-	75.3 / 59.7	67.6 / 48.5	64.0 / 47.7
mT5-Large	81.8	91.2	-	81.2 / 65.9	73.9 / 55.2	71.1 / 54.9
mT5-XL	84.8	91.0	-	82.7 / 68.1	75.1 / 56.6	79.9 / 65.3
mT5-XXL	**87.8**	**91.5**	-	**85.2 / 71.3**	**76.9 / 58.3**	**82.8 / 68.8**
In-language multitask (models fine-tuned on gold data in all target languages)						
mBERT	-	-	89.1	-	-	77.6 / 68.0
mT5-Small	-	-	83.4	-	-	73.0 / 62.0
mT5-Base	-	-	85.4	-	-	80.8 / 70.0
mT5-Large	-	-	88.4	-	-	85.5 / 75.3
mT5-XL	-	-	90.9	-	-	87.5 / 78.1
mT5-XXL	-	-	**91.2**	-	-	**88.5 / 79.1**

mT5: Model Capacity

- Model capacity is the key to improving cross-lingual performance.
- We can see from the figure That as the size of the model Increases the performance Increases.

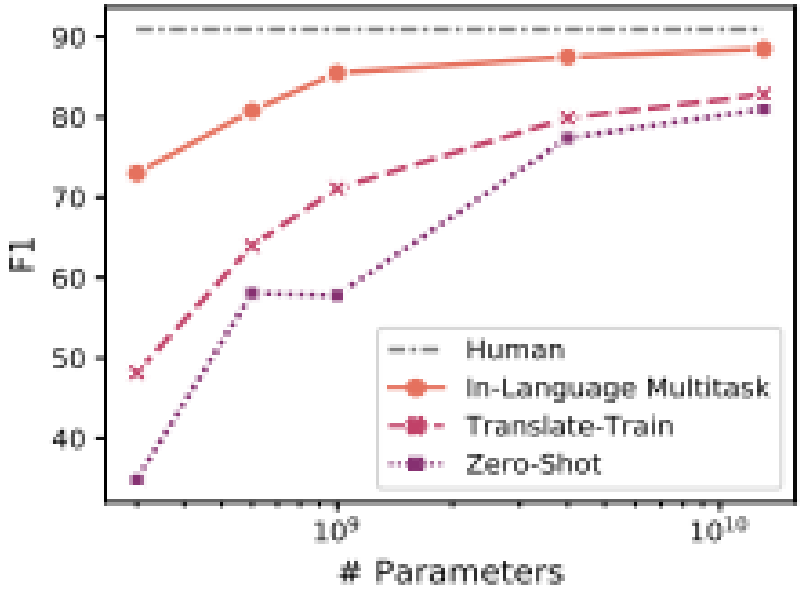


Figure 2: Average F1 on the TyDi QA GoldP task across languages. Performance improves with increasing model capacity. The importance of in-language training data (whether gold In-Lanugage Multitask or synthetic Translate-Train) decreases with model scale, as seen by Zero-Shot closing the quality gap.

mT5: Comparison with similar sized dedicated model

- Compare the performance of mT5 and T5 when fine-tuned on the SQuAD.
- Small and base mT5 fall short Of English Counterpart T5 but Larger model closed the gap

	T5	mT5
Small	87.2 / 79.1	84.7 / 76.4
Base	92.1 / 85.4	89.6 / 83.8
Large	93.8 / 86.7	93.0 / 87.0
XL	95.0 / 88.5	94.5 / 88.9
XXL	96.2 / 91.3	95.6 / 90.4

Table 3: Comparison of T5 vs. mT5 on SQuAD question answering (F1/EM).

mT5: Accidental translation

seis años	six years	Translated from Spanish
Zweiten Weltkrieg	the Second World War	Translated from German
新英格兰爱国者队	New英格兰爱国者队	Partially translated Chinese “New England Patriots”
хлоропласт	chloroпласт	Partially translated Russian “chloroplast”

- As English-only finetuning proceeds, the model’s assigned likelihood of non-English tokens presumably decreases, eventually reaching the point where English becomes the most likely answer to any question.
- simply mix in our unsupervised multilingual pre-training task during fine-tuning.

---

mT5: Student Reviews(Pros)

- mT5 also addresses the issue of "accidental translation," which occurs when a portion of the model's predictions is translated into a different language in a zero-shot setting.(Shivangi)
- Sampling during training has also been done in proportion to the available data which is very important and a hyperparameter was introduced to control how much boost is to be given to low resource languages - to prevent overfitting and underfitting. (Shreya)
- For the two largest mT5 models, zero-shot and translate-train perform similarly, showing that machine translations of the monolingual dataset do not improve performance a lot as model size increases. Thus one qualitative result that the paper shows is that one can avoid costly steps of annotating data in more than one language when using large models.(Jai)

mT5: Student Reviews(Pros)

- uses prediction of text of label, which reduces hyper parameters for training multiple downstream tasks. (Rohit)

mT5: Student Reviews(Cons)

- It would have been interesting to see effect a particular language has on mT5 performance. For example, if we remove hindi from mC4 data and then train the mT5, will it affect models performance on related language like marathi. This analysis is missing.(Vishal)
- Training the mT5 model is compute and resource heavy task, thus reproducing its pre-training process is not easily possible. (Shivangi)
- The paper also has little in terms of innovation in strategy. It just uses a already published model T5 and trains it on multiple languages. (Though the other part of accidental translation is quite interesting) (Jai)
- Results can be shown for more languages in the paper. Only English results are offered, which does not show the totality of this model.(Seshank)

mT5: Student Reviews(Extensions)

- One can do knowledge distillation on mT5 so that smaller models can be made which are computationally cheaper to train and test with.(Jai)
- They wanted to keep text to text interface, which is understandable, but could have added task specific tuning mechanisms and compare the results and check if proposed methods are beating them.(Rohit)
- I am quite interested to know if we can do some pruning to see how many parameters do we actually need for a particular language, or for a particular performance. LM's keep getting bigger, but it would be nice to have such an experiment, partially motivated by previous class on Knowledge distillation, lottery ticket hypothesis etc. (Harman)

Thank You