

Large Language Models for African Languages

Dr. Jan Buys

Senior Lecturer

Department of Computer Science

University of Cape Town



UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

UCT Natural Language Processing research group

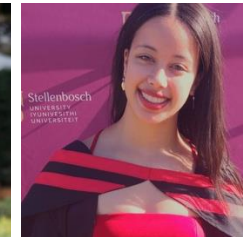
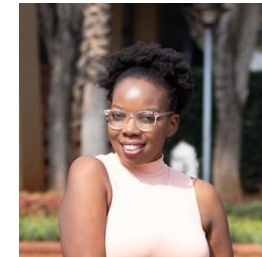
Principle Investigator: Dr. Jan Buys

Co-Investigator: Dr. Francois Meyer

4 PhD students, 5 Masters students

We perform research in natural language processing and machine learning. Research topics include:

- Language modelling and machine translation for low-resource languages
- Modelling knowledge and linguistic structure with deep learning models
- Creating NLP datasets and models for South African language



















Large Language Models (LLMs)

- LLMs have become the dominant approach to Natural Language Processing
- The application of LLMs have expanded beyond traditional language understanding and generation tasks to general-purpose AI systems, including e.g. coding, search, reasoning and deep research
- However due to a lack of sufficient data and computational resources work on LLMs for African languages, and LLMs developed in Africa, has been limited relative to efforts for other major languages
- This tutorial will give an overview of the technical foundations of LLM development, covering both general principles and their application to the particular challenges of African languages

Large Language Models (LLMs)

- Examples of LLM capabilities:
- Not limited to one of a fixed number of tasks

	Q&A Answer questions based on existing knowle...		Grammar correction Corrects sentences into standard English.
	Summarize for a 2nd grader Translates difficult text into simpler concep...		Natural language to OpenAI API Create code to call to the OpenAI API usin...
	Text to command Translate text into programmatic commands.		English to other languages Translates English text into French, Spanish...
	Natural language to Stripe API Create code to call the Stripe API using nat...		SQL translate Translate natural language to SQL queries.
	Parse unstructured data Create tables from long form text		Classification Classify items into categories via example.
	Python to natural language Explain a piece of Python code in human un...		Movie to Emoji Convert movie titles into emoji.
	Calculate Time Complexity Find the time complexity of a function.		Translate programming languages Translate from one programming language ...
	Advanced tweet classifier Advanced sentiment detection for a piece o...		Explain code Explain a complicated piece of code.

Large Language Models (LLMs)

Maths

Prompt: Can you write a proof that there are infinitely many primes, with every line that rhymes?

GPT-4:

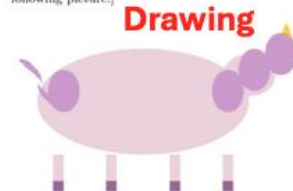
Yes, I think I can.
Though it might take a clever plan.
I'll start by noting Euclid's proof,
Which shows that primes aren't just aloof.
Assume we have a finite list,
Of primes, and that none have been missed.
Multiply them all together,
And add one, just to be clever.
The result will be a number,
That has no prime factors, oh wonder!
But every number has a prime,
That divides it, it's just a matter of time.
So we've found a contradiction,
And our finite list needs eviction.
There must be infinitely many primes,
And that's the end of my rhyming lines.

Prompt: Can you generate a pyplot for the following data: $x = [1, 3, 5, 6, 8]$, $y1 = [2, 3, 5, 18, 1]$, $y2 = [3, 5, 6, 8, 1]$, $y3 = [5, 1, 2, 3, 4]$, $y4 = [9, 7, 2, 3, 1]$. I want $y1, y2$ to be in the same plot, but $y3$ is in another plot next to that plot, $y4$ is in below. I want the legend of $y1$ to be "bob", $y2$ to be "alice", $y3$ to be "bilbo", $y4$ to be "allie". I want the x-axis to be labeled with "time" and y axis to be labeled with "money". I want to add a 10%-40% random error bar to each curve, through all times (including non-integers). I want smoothed curves to show the plot, and smoothed error bar. Do not use linear interpolation, use smooth interpolation! I want to also add some small zig-zag to the smoothed curve to make it look more real. I want to put a baseline as the mean of each line. I want to put a pie chart below indicating the fraction of the four people in each time step. I also want an animation to show how the fractions are changing in the pie chart in "continuous time". Interpolate the missing fractions! I also want vertical line animation in other three plots to match the pie chart. I want the fanciest plot. Please add as many fancy things as possible.

Here is a possible code to generate a pyplot for the following data, with some comments to explain the steps:

Prompt: Draw a unicorn in TikZ.

GPT-4: [Produces \LaTeX compiling to following picture.]

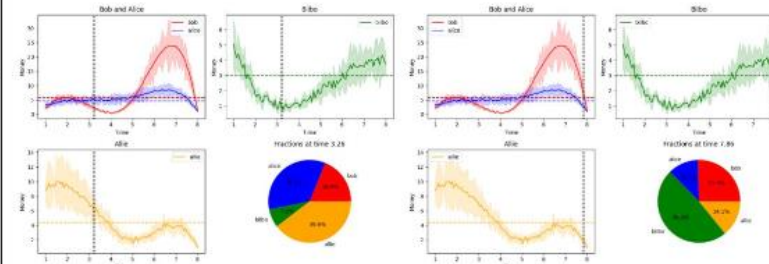


Plotting

Prompt: Can you generate a pyplot for the following data: $x = [1, 3, 5, 6, 8]$, $y1 = [2, 3, 5, 18, 1]$, $y2 = [3, 5, 6, 8, 1]$, $y3 = [5, 1, 2, 3, 4]$, $y4 = [9, 7, 2, 3, 1]$. I want $y1, y2$ to be in the same plot, but $y3$ is in another plot next to that plot, $y4$ is in below. I want the legend of $y1$ to be "bob", $y2$ to be "alice", $y3$ to be "bilbo", $y4$ to be "allie". I want the x-axis to be labeled with "time" and y axis to be labeled with "money". I want to add a 10%-40% random error bar to each curve, through all times (including non-integers). I want smoothed curves to show the plot, and smoothed error bar. Do not use linear interpolation, use smooth interpolation! I want to also add some small zig-zag to the smoothed curve to make it look more real. I want to put a baseline as the mean of each line. I want to put a pie chart below indicating the fraction of the four people in each time step. I also want an animation to show how the fractions are changing in the pie chart in "continuous time". Interpolate the missing fractions! I also want vertical line animation in other three plots to match the pie chart. Please add as many fancy things as possible.

Here is a possible code to generate a pyplot for the following data, with some comments to explain the steps:

GPT-4: [We give two snapshots from the animation resulting from the produced code]



GPT-4: Explanations

This is a conversation between a user and an artificial intelligence agent that follows instructions or answers questions. The AI always follows the instructions correctly.

User: Can you compose a short tune (say four to eight bars) using ABC notation?
AI: There are many possible ways to compose a short tune in abc notation, but here is one example:
X:1
T:Example Tune
M:4/4
L:1/8
K:C
|:G2AB c2BA | G2AB c2c2 | edcB c2BA | G4-
G2c2 :|
|:c2de f2ed | c2de f2f2 | g2fe d2cB | c4-
c2c2 :|
This tune has two four-bar phrases, each repeated once, forming an A part and a B part. The key is C major.

GPT-4:

User: After the first bar G2AB c2BA, you chose G to start the next bar. Give an example of a change to the notes in the first bar that would have caused you to pick D instead of G as the first note in the second bar.
AI: One possible change to the notes in the first bar that would have caused me to pick D instead of G as the first note in the second bar is:

|:G2AB c2Bc | ...

This change creates a stronger cadence on the C note at the end of the first bar, which is the fifth degree of the scale. This suggests a modulation to the dominant key of G major, which has D as its fifth degree. Therefore, starting the second bar with D would create a smooth transition to the new key and imply a harmonic progression of V-I.

Prompt:

```
struct mystruct_A { char a; int b; char c; } x;  
struct mystruct_B { int b; char a; char c; } y;  
  
#include <stdio.h>  
int main(void) {  
    printf("Size of x: %zu bytes\n", sizeof(x));  
    printf("Size of y: %zu bytes\n", sizeof(y));  
    return 0;  
}
```

Possible Output:

GPT-4:

There are different ways to implement and align structures in memory, so the exact output may vary depending on the compiler, platform, and options used. However, one possible output is:

Size of x: 12 bytes
Size of y: 8 bytes

This output assumes that the compiler uses 4-byte alignment for structures, meaning that each structure member starts at an offset that is a multiple of 4 bytes from the beginning of the structure. This may result in some padding bytes being inserted between or after the members to maintain alignment. For example, `int b` in `mystruct_A` may be padded to 4 bytes, and `char c` in `mystruct_B` may be padded to 4 bytes. The size of a structure is the sum of the sizes of its members plus any padding bytes.

Reasoning

Why is 31 a prime number?

Reasoned about number 31 being prime for 4 seconds

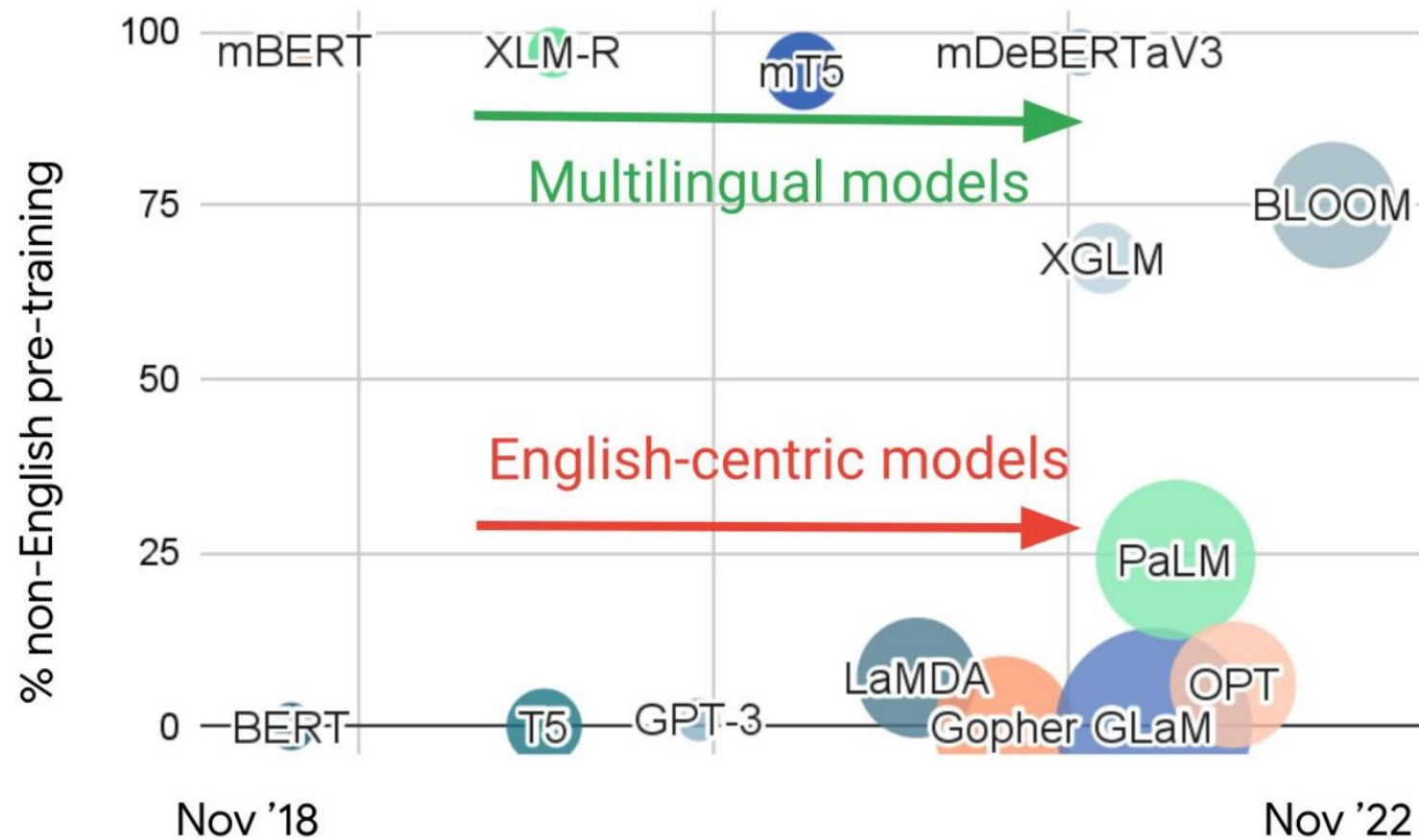
Alright, let's go through this. A prime number is an integer greater than 1, divisible only by 1 and itself. So why is 31 prime? Its only divisors are 1 and 31. I check for divisibility — it's not divisible by 2 ($31/2 = 15.5$, remainder 1), not by 3 ($31/3 = 10.33$, remainder 1), and not divisible by 5 ($31/5 = 6.2$, remainder 1). Since no other primes divide it, 31 is prime!

Since 31 has no divisors other than 1 and itself, it is prime. To determine this, we check possible divisors up to $\sqrt{31}$, approximately 5.6. The divisors to check are 2, 3, and 5, and none divide evenly into 31 — $31/2 = 15.5$, $31/3 \approx 10.33$, $31/5 = 6.2$. Since 31 isn't divisible by these, it confirms that 31 is prime. The fundamental theorem of arithmetic also supports that 31 cannot be factored further.

Understanding Programs

Large Language Models (LLMs)

- Many current pretrained models are multilingual



Large Language Models (LLMs)

- The benefits of LLMs are currently mostly limited to English, and to some extent other high-resource languages

Language	Cat.	ChatGPT	
		(en)	(spc)
English	H	70.2	70.2
Russian	H	60.8	45.4
German	H	64.5	51.1
Chinese	H	58.2	35.5
French	H	64.8	42.2
Spanish	H	65.8	47.4
Vietnamese	H	55.4	44.8
Turkish	M	57.1	37.1
Arabic	M	55.3	22.3
Greek	M	55.9	54.5
Thai	M	44.7	11.5
Bulgarian	M	59.7	44.6
Hindi	M	48.8	5.6
Urdu	L	43.7	6.3
Swahili	X	50.3	40.8

XNLI

Language	Code	Cat.	ChatGPT	
			(en)	(tgt)
English	en	H	75.0	75.0
Russian	ru	H	50.2	53.5
German	de	H	52.6	61.0
Chinese	zh	H	50.2	42.5
Japanese	jp	H	41.9	43.0
French	fr	H	50.5	61.7
Spanish	es	H	53.3	62.5
Italy	it	H	50.6	55.9
Dutch	nl	H	52.9	60.4
Polish	pl	H	35.2	51.1
Portugese	pt	H	49.5	59.2
Vietnamese	vi	H	42.3	47.9
Arabic	ar	M	49.4	47.3
Hindi	hi	M	41.1	38.6
Urdu	ur	L	34.7	24.5
Swahili	sw	X	35.6	46.6
Average			47.8	51.9

X-CSQA

Lang.	ChatGPT		NLLB	
	BLEU	chrF	BLEU	chrF
srp_Cyrl	1.36	3.26	43.4	59.7
kon_Latn	0.94	8.50	18.9	45.3
tso_Latn	2.92	15.0	26.7	50.0
kac_Latn	0.04	2.95	14.3	37.5
nso_Latn	3.69	16.7	26.5	50.8
jpn_Jpan	28.4	32.9	20.1	27.9
nno_Latn	37.1	58.7	33.4	53.6
zho_Hans	36.3	31.0	26.6	22.8
zho_Hant	26.0	24.4	12.4	14.0
acm_Arab	28.2	44.7	11.8	31.9

Machine translation

Large Language Models (LLMs)

- Why do LLMs lag behind for other languages?

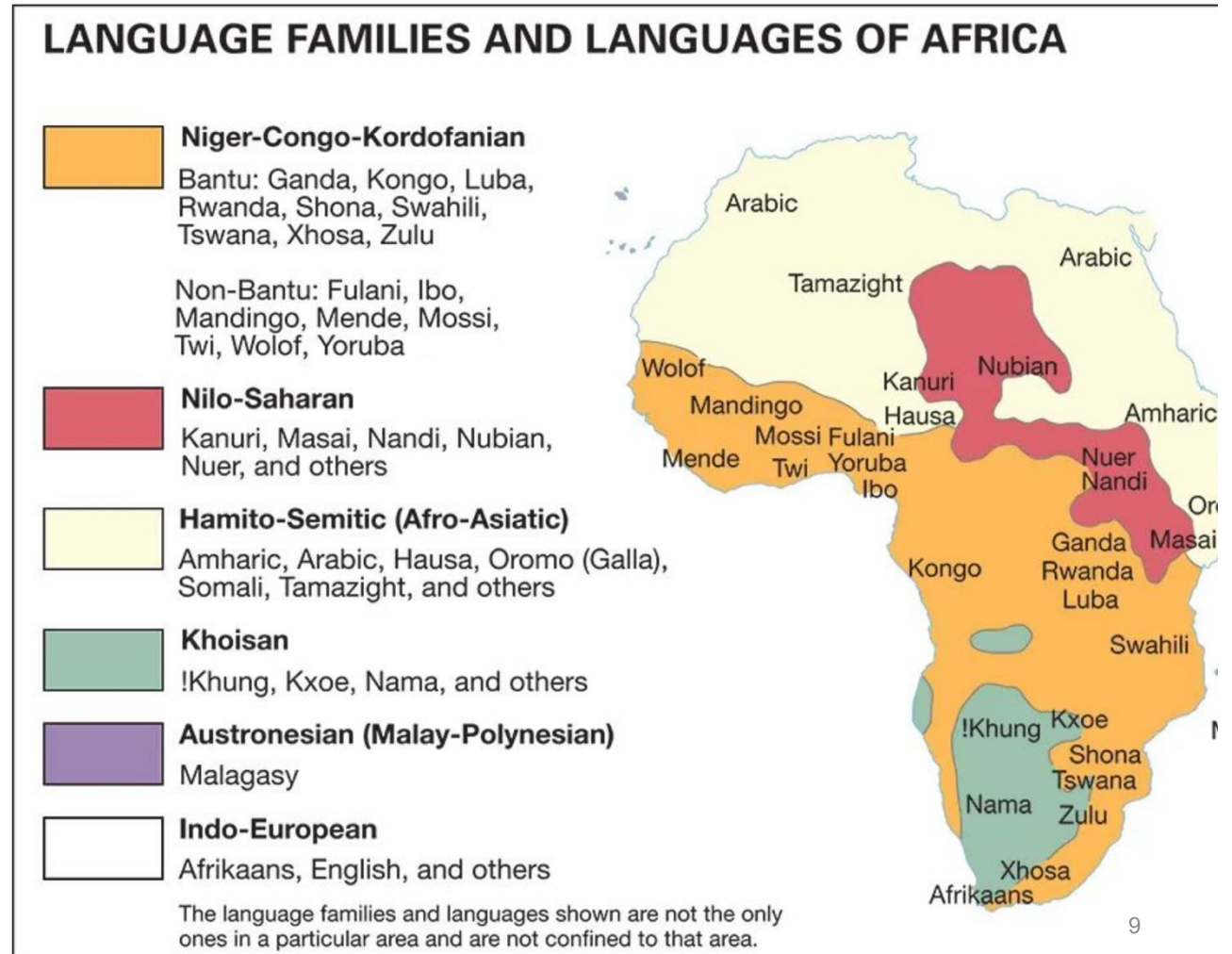
Lack of:

- Pretraining data
- Token representation
- Instruction tuning data
- Human preference data
- Reasoning data
- Limited transfer from English

For most African languages, the availability of data in relation to the number of language speakers is extremely low

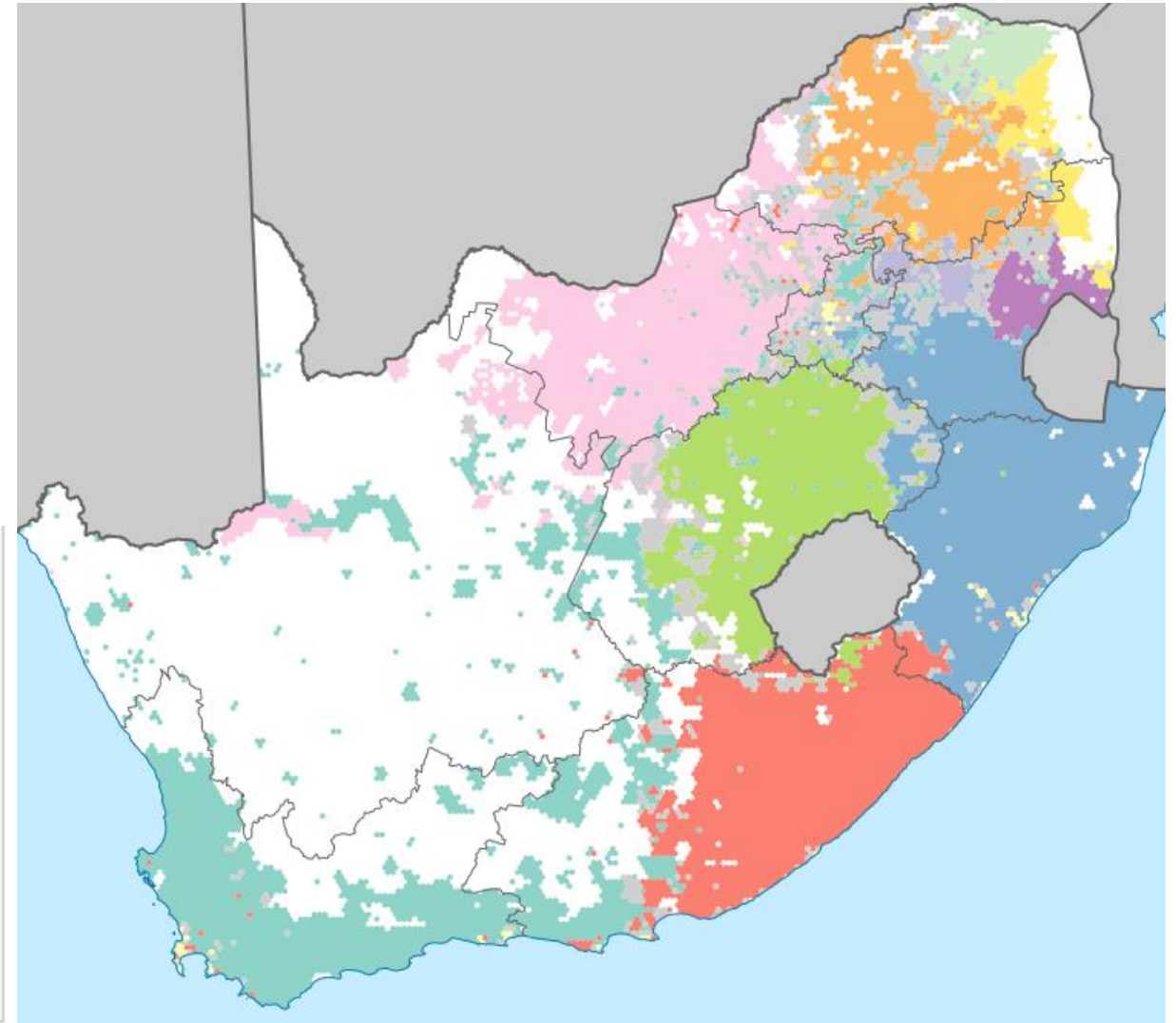
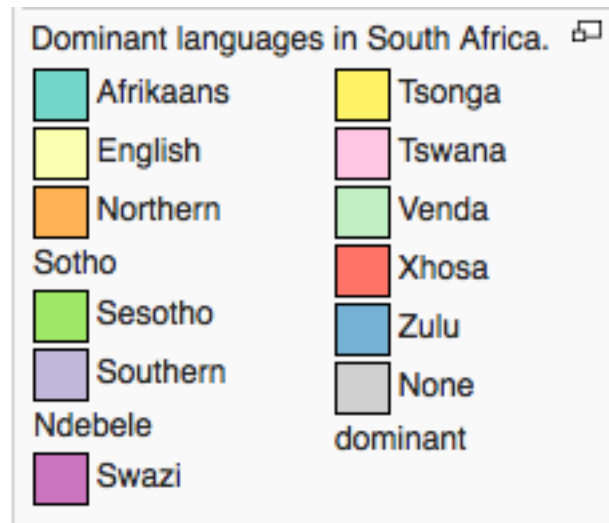
African languages

- 54 countries
- >2000 languages
- 33% of the world's languages
- 16% of the world's population
- While African languages have rich oral histories, many developed as written languages at a relatively late stage and for historical and political reasons most are not used as widely for educational and official purposes
- Therefore the amount of written text available online and offline is much smaller than for most Western and Asian languages with similar number of speakers



South African Languages

- 12 Official languages
- Two largest language groups:
- Nguni languages (28M)
 - Sotho/Tswana languages (17M)



Tutorial Overview

1. Brief introduction to Large Language Models

- Language modelling
- Model architectures
- Application to NLP

2. LLM Pretraining

- Pretraining data
- Tokenization
- Continual pretraining

3. LLM Post-training

- Task-specific fine-tuning
- Instruction fine-tuning
- Reinforcement Learning
- Evaluation

Part 1: Brief introduction to Large Language Models

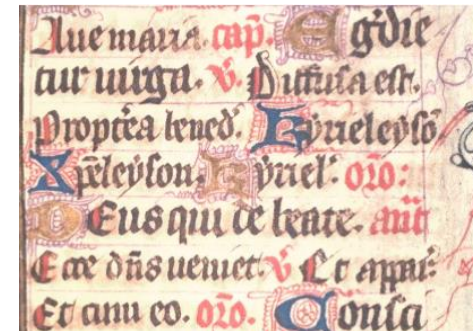
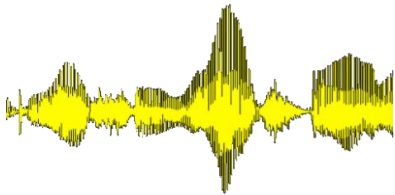
Scope

- We will focus on the technical foundations of Large Language Models, not on how to use ChatGPT to do task X
- We will only cover text-based models, not multimodal models including images/video/audio
- We'll cover model architectures and machine learning details fairly briefly and mostly focus on the other aspects of LLM development
- While there has been an increase in research on LLMs for African languages, a lot of work remains to be done
- This is a very rapidly evolving field: it isn't possible to cover all relevant methods and current best practices may change
- The ethics and safety of LLMs is an important topic that we can't fully cover here, and more research has to be done on the ethics/safety of African LLMs

Natural Language

What is special about human language?

- A human language is a system specifically constructed to convey the speaker/writer's meaning
- A human language is a **discrete/symbolic/categorical signaling system**
 - rocket = 🚀 ; violin = 🎻
 - With very minor exceptions for expressive signaling (e.g. "I loooove it.", "Whoomppaaa")
- The categorical symbols of a language can be encoded as a signal for communication in several ways: Sound, gesture, images (writing)
 - The **symbol** is **invariant** across different encodings



Natural Language Processing

Natural Language Processing is the study of systems that process human language and enable computers to perform useful tasks involving human language

- The fundamental goal is *deep understand* of *general-purpose* language, not just string processing or keyword matching

Main components of NLP:

- Analysis/Understanding ($NL \rightarrow \mathcal{R}$)
- Generation ($\mathcal{R} \rightarrow NL$)
- Acquisition of the representation (\mathcal{R}) from knowledge and data

\mathcal{R} is a representation that is useful for a computer. It might or might not be interpretable to humans or based on scientific theories.

- Traditional NLP often used explicit linguistic representations (Parts-of-Speech, syntactic trees, semantic graphs)
- Current approaches are based on vector representations of letters, words, or text passages

Natural Language Processing

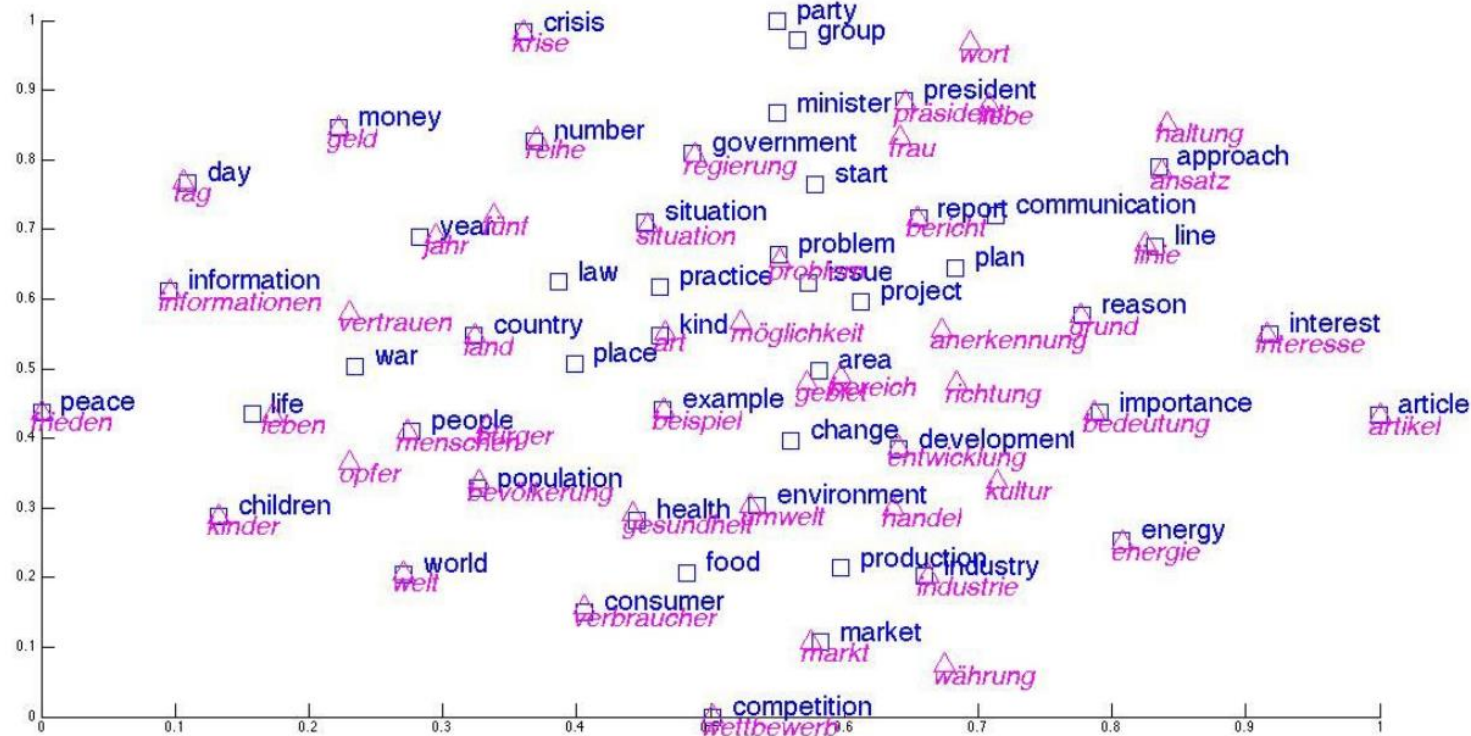
- Word vectors: similar words have similar vectors



Two-dimensional (t-SNE)
projection of embeddings

Natural Language Processing

- Multilingual modelling: Train word embeddings multiple languages. Encodes cross-lingual word meaning (e.g. similar meanings).



Language modelling

- Suppose we have a text passage or document consisting of n (e.g. 1 000) word **tokens** in language that where V is the set of word **types** – the vocabulary (e.g. $|V| = 10\,000$)
- A language model is a probabilistic model of a document

$$P(w_1, w_2, \dots, w_n; \theta)$$

- We can use the **chain rule** to decompose the distribution:

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$$

- An **autoregressive language model** predicts the next token conditioned on the previous tokens

Language modelling

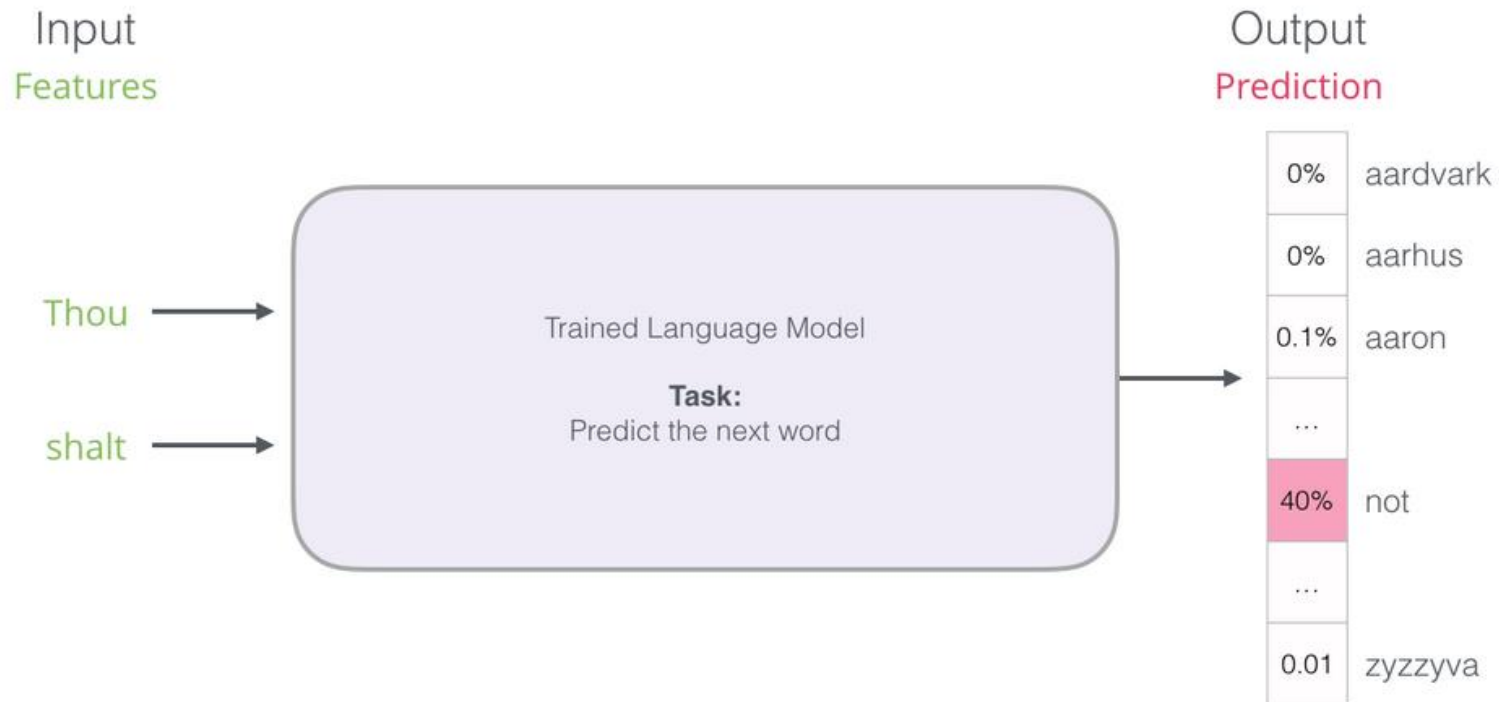
- Predict the next word in a word sequence



<https://jalammar.github.io/illustrated-word2vec/>

Language modelling

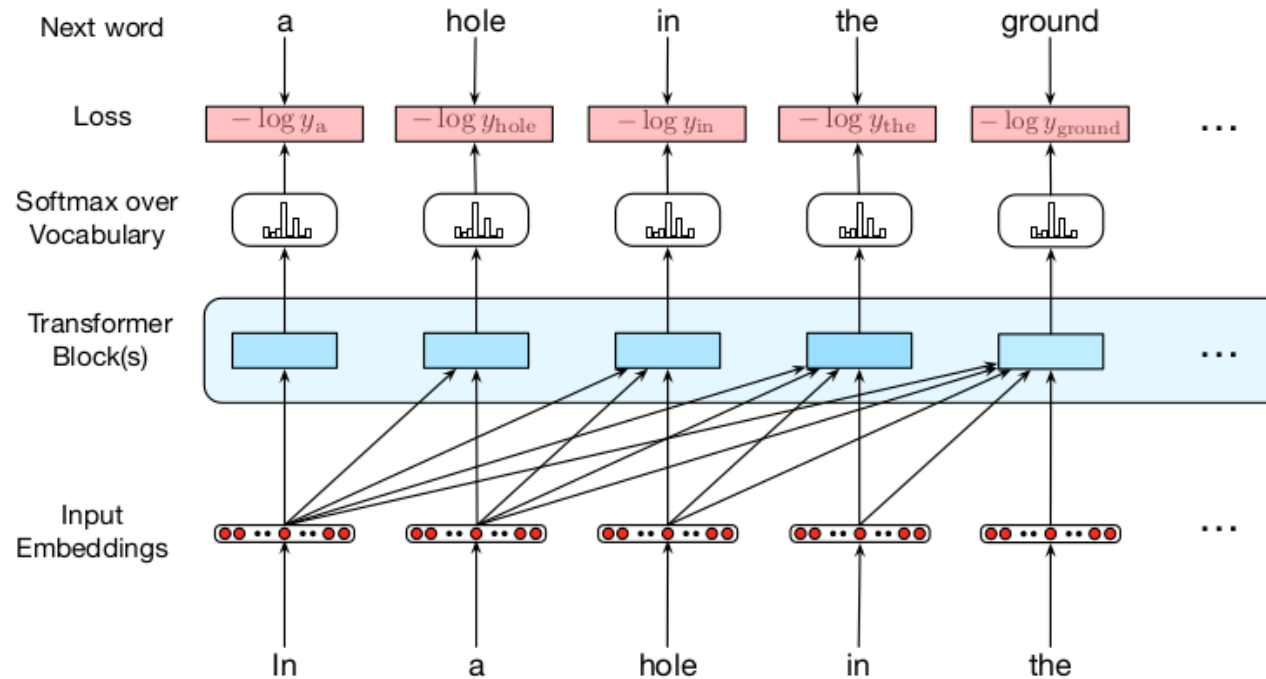
- Predict the next word in a word sequence by assigning a probability to each word in the vocabulary



Language modelling

- **Autoregressive** language modelling: predict the next word repeatedly

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$$



Language modelling

Classical view: Language models are used to model fluency, can disambiguate possible outputs generated by another model

- E.g. $P(\text{"recognize speech"}) > P(\text{"wreck a nice beach"})$

Modern view:

- Language models are used directly to generate text
- Language models can recall factual knowledge from the training data (learnt via the next word prediction objective)
- Language models can perform reasoning
- Language models can execute instructions when prompted

Language modelling architectures

- The first generation of language models were ***n*-gram** models, which condition the next word probability on a fixed number of previous words (typically $n \approx 5$), using formulas based on how many times each *n*-gram appear in the training data

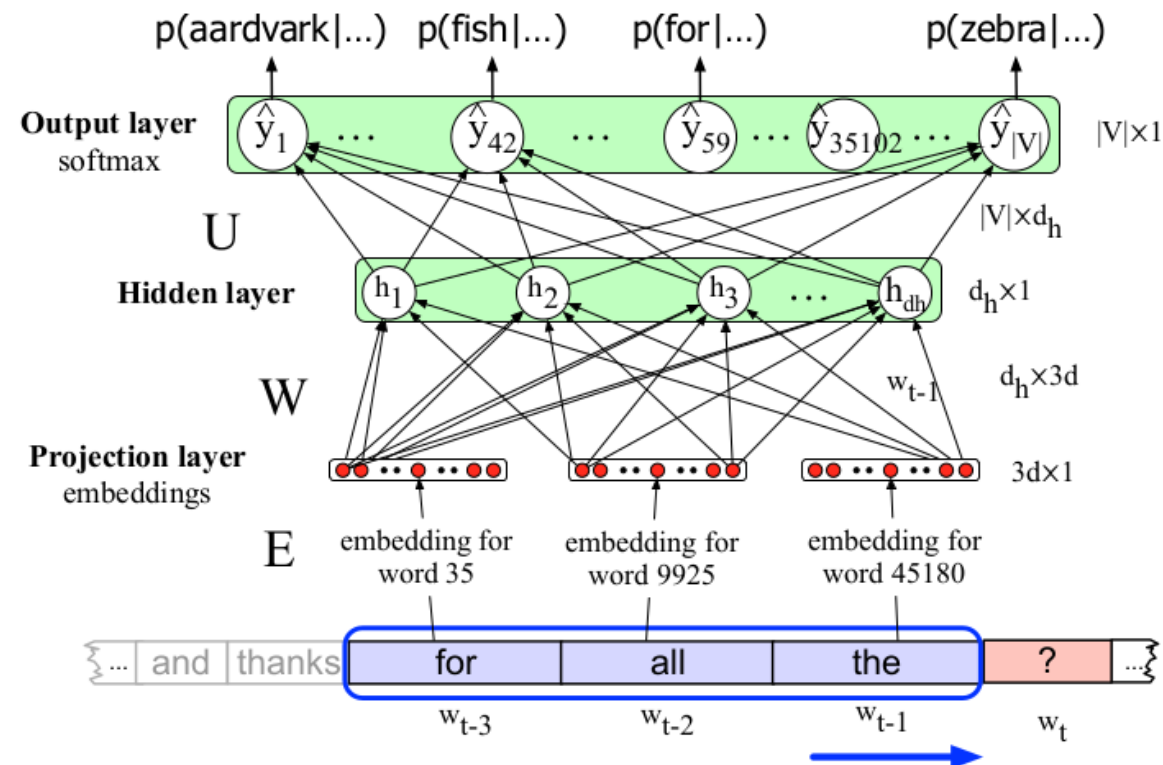
$$P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-k} \dots w_{i-1})$$

- These models are efficient for modelling fluency when used together with another model, but due to the limited context length cannot model long-distance dependencies between words and cannot be used as text generation models independently

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Language modelling architectures

Feedforward neural networks can also be used to estimate LM probabilities with a relatively small fixed context, but suffer from many of the same limitations as count-based n -gram models

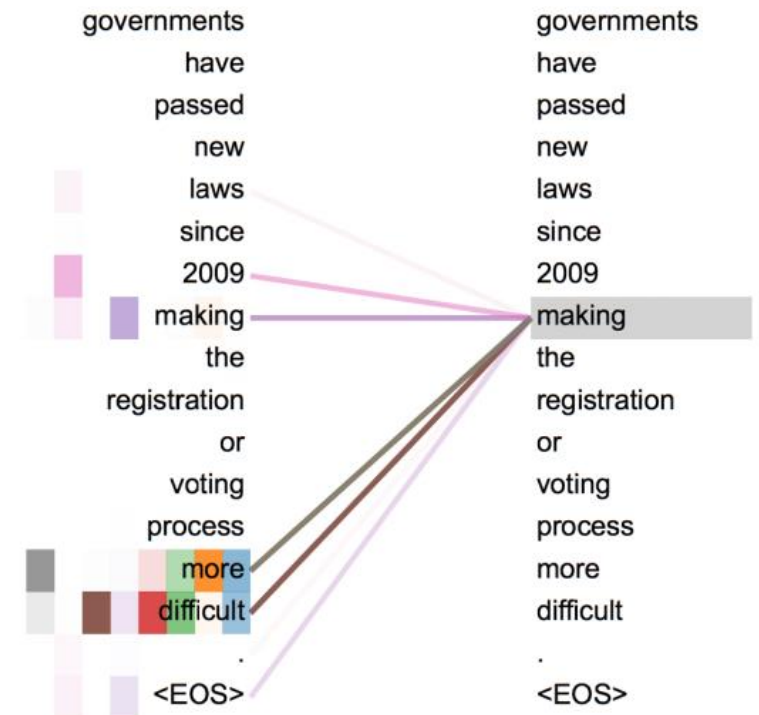


Language modelling architectures

- Recurrent neural networks model sequences directly without an explicit context length limitation. LSTM recurrent neural networks were the first widely used deep learning-based language models.
- Research showed that LSTMs can be pre-trained as language models and then applied to downstream applications, but there were fundamental scalability issues

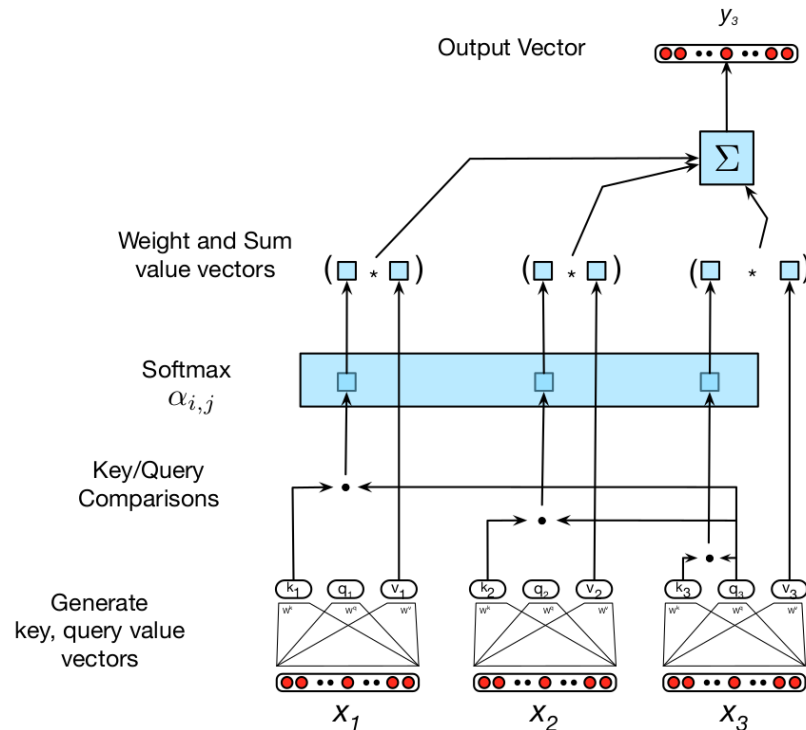
Language modelling architectures

- The **Transformer** architecture became the dominant neural network architecture for Natural Language Processing, and the basis for most Large Language Models
- The Transformer is based on the concept of *self-attention* where the relationship between each pair of input elements are modelled and the relative "importance" of the relationship between each pair of elements is determined.
- The attention mechanism computes a contextual representation of each input element as a weighted average of all input representation
- Attention is computed multiple times (with multiple attention "heads") and in multiple stacked layers
- While the context size has to be fixed, in practice Transformers can model much longer contexts than LSTMs

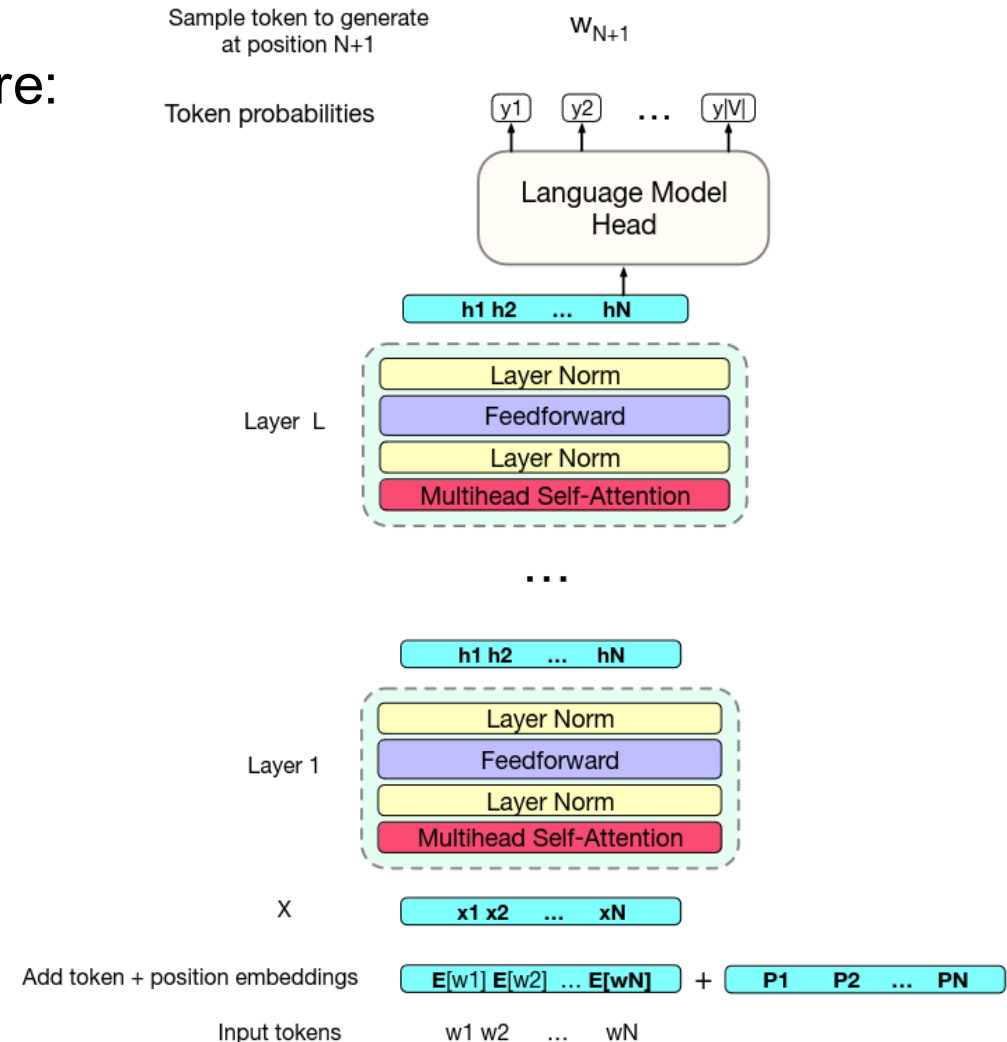


Language modelling architectures

- Complete Transformer architecture:

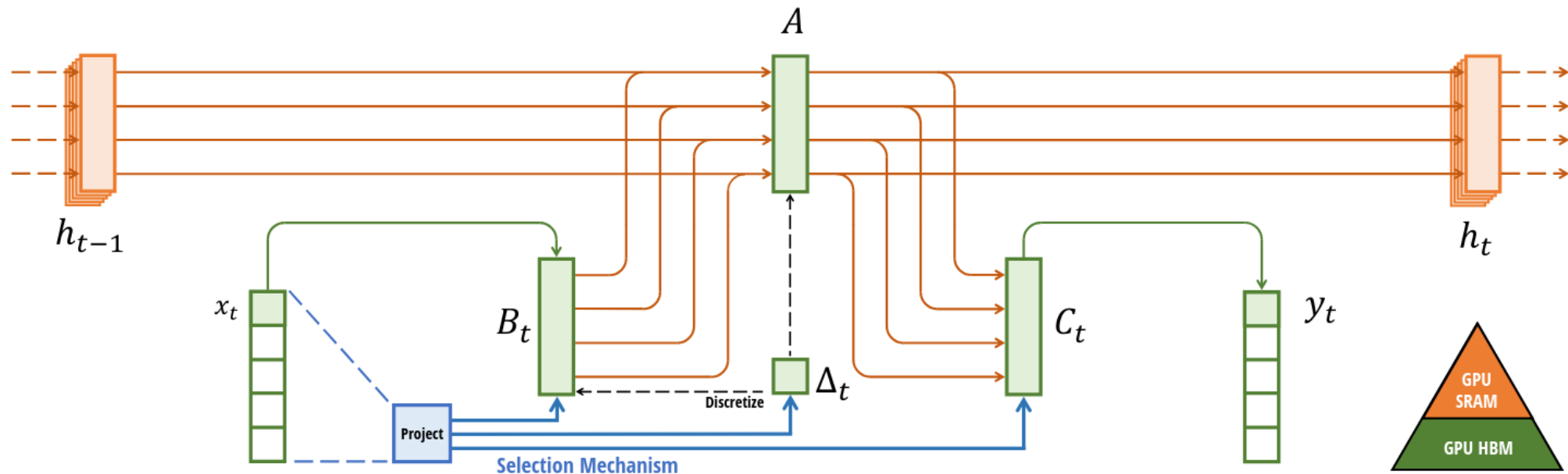


Self-attention computation using x_3 as query



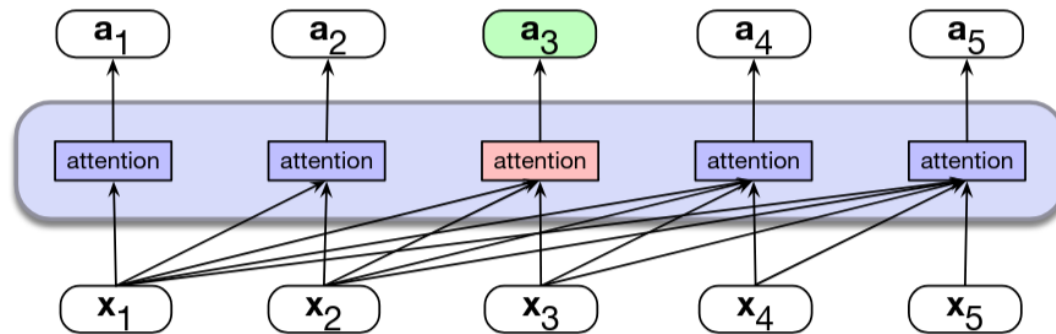
Language modelling architectures

- While the Transformer architecture remains dominant in Large Language Models, research continues into alternative architectures
- The Mamba architecture, which is a selective state space model, is currently the most promising alternative

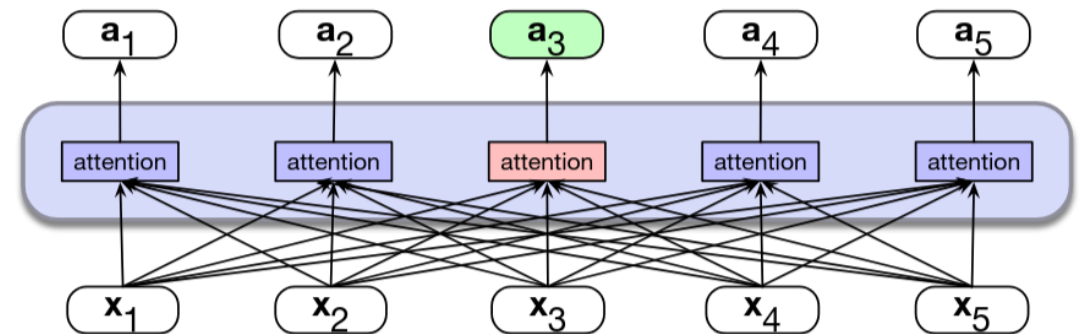


Language modelling architectures

- **Decoder** language models: Autoregressive language models can only encode context from previous tokens in the sequence at each time step
 - “Causal” self-attention in the Transformer
- **Encoder** language models: When we don’t need to generate text, we don’t need this restriction and can instead use both the past and the future as context
 - Bidirectional self-attention in the Transformer



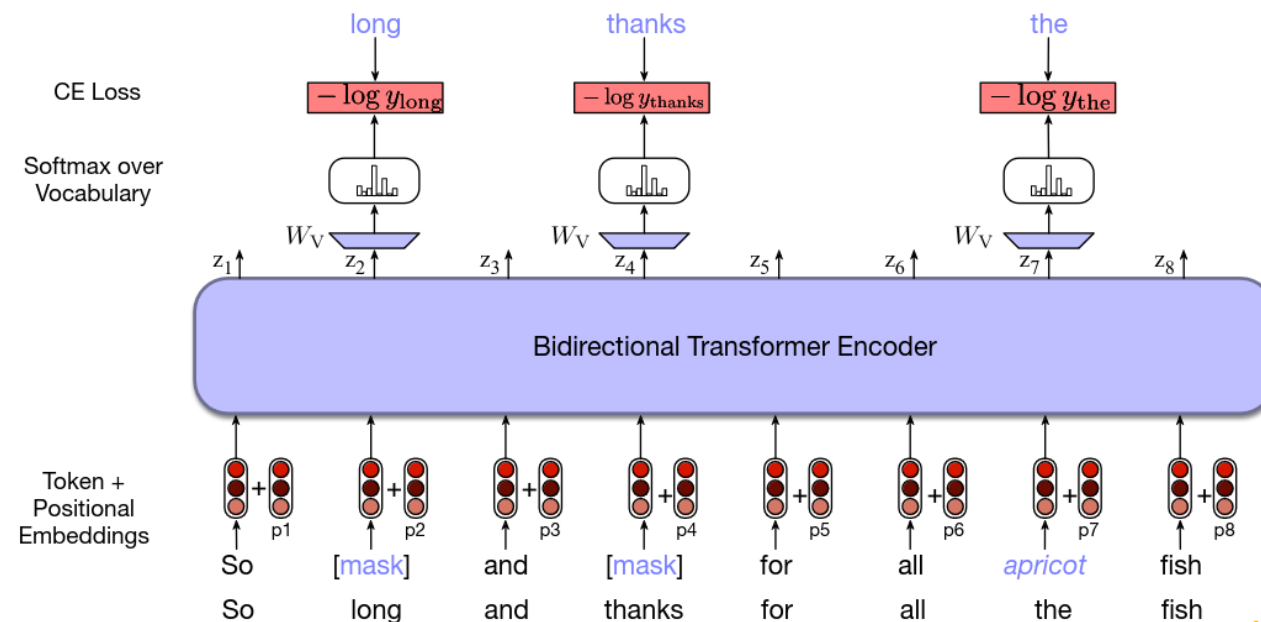
a) A causal self-attention layer



b) A bidirectional self-attention layer

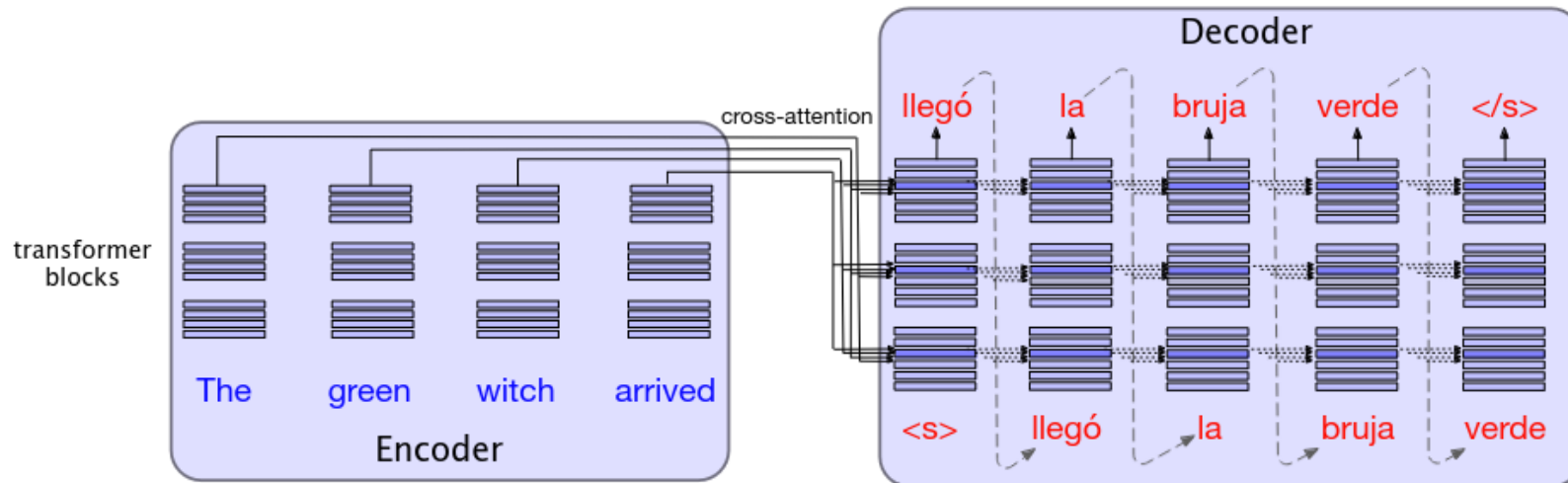
Language modelling architectures

- Encoder language models cannot be pretrained with a next-word prediction objective function
- Instead they are pretrained for masked language modelling: Some input tokens are replaced with a mask (placeholder) and the task is to re-predict these tokens
- Most prominent encoder language model is BERT



Language modelling architectures

- The final configuration is the **encoder-decoder** architecture
- An input sequence is encoded by the encoder, and an output sequence is generated by the decoder
- Originally applied for machine translation, but can also be pretrained with a variant of masked language modelling
- Uses cross-attention between the encoder and decoder



Language modelling applications

How can we apply language models to NLP tasks and leverage the representations learnt from pretraining?

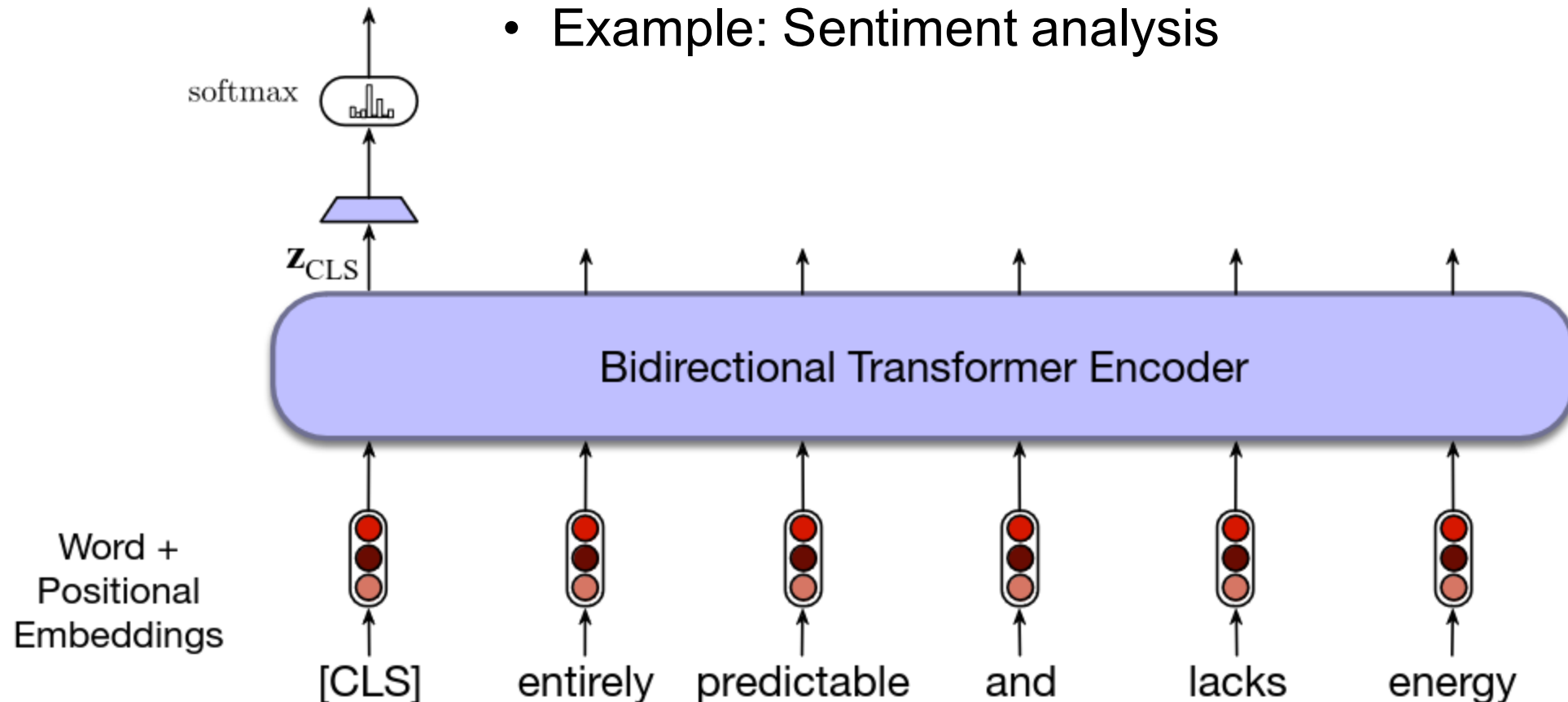
- Fine-tune models for a specific task by adding an output layer on top of the LM, leveraging the contextual representations provided by the LM encoding the input
 - Can fine-tune all or some of the LM parameters, or only the output layer
- Leverage the LM's next word prediction or fill-in-the-blank capabilities directly
 - Reformulate tasks as text completion

Task-specific fine-tuning: Sequence classification

$$\mathbf{y} = \text{softmax}(\mathbf{W}_C \mathbf{z}_{CLS})$$

y_{CLS}

- Add an output classification layer and fine-tune
- Example: Sentiment analysis

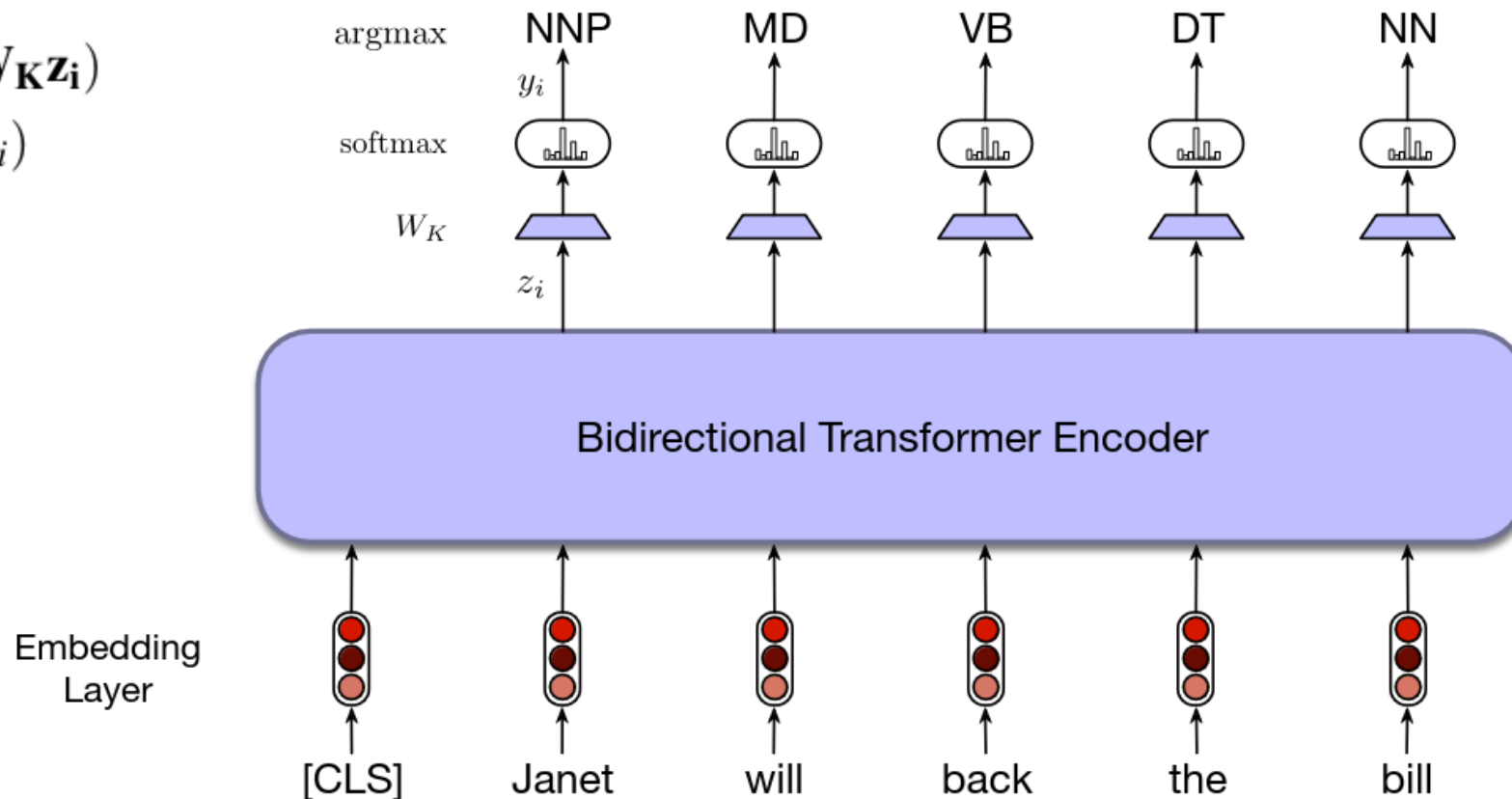


Task-specific fine-tuning: Sequence labelling

- Add an output classification layer per token and fine-tune
- Example: POS tagging

$$\mathbf{y}_i = \text{softmax}(\mathbf{W}_K \mathbf{z}_i)$$

$$\mathbf{t}_i = \text{argmax}_k(\mathbf{y}_i)$$



In-context learning

Apply the LM to a task by constructing a *prompt* and asking the LM to predict the output as the continuation of the input sequence

- Zero-shot: Give the task description directly
- Few-shot: using only a few examples
- Example for zero-shot Question Answering:

Context → Q: Who played tess on touched by an angel?

A:

Target Completion → Delloreese Patricia Early (July 6, 1931 { November 19, 2017), known professionally as Della Reese

In-context learning

- Example for few-shot sentiment analysis:

Instruction Tell me the sentiment of this review

Example The movie begins The plot is engaging, thoroughly enjoyable.
The movie is **great**

Oh, how can such a fine cast produce such a terrible performance..... A total waste of time.
The movie is **pathetic**

Prompt It is just a rehash of old movies
The movie is <MASK>

What kinds of things does pretraining learn?

- Stanford University is located in _____, California. [Trivia]
- I put ____ fork down on the table. [syntax]
- The woman walked across the street, checking for traffic over ____ shoulder. [coreference]
- I went to the ocean to see the fish, turtles, seals, and _____. [lexical semantics/topic]
- Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was _____. [sentiment]
- Iroh went into the kitchen to make some tea. Standing next to Iroh, Zuko pondered his destiny. Zuko left the _____. [some reasoning – this is harder]
- I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, _____ [some basic arithmetic]

Part 2: Large Language Model Pretraining

Pretraining data

- Pretraining data is the basis for developing LLMs
- “Scaling laws” show that model performance is a function of the pretraining data size, model size, and training time
- Large open-source LLMs are now typically trained on >15 trillion tokens of text, but even smaller LLMs use >1 trillion tokens
- In order to understand and generate text in African languages, models need pretraining data in these languages

Pretraining data

- The most common approach for constructing publicly available pretraining datasets is to use Common Crawl dumps as a starting point
 - There are web crawls of publicly available websites
- Then various filtering and preprocessing steps are performed to extract text in the target language(s)
- The first pretraining datasets focussed mostly on English, but there has been an increase in efforts to build multilingual pretraining datasets
- Example datasets:
 - mC4, CC100 (older)
 - Glot500
 - CulturaX
 - Fineweb2
 - HPLT

Pretraining data

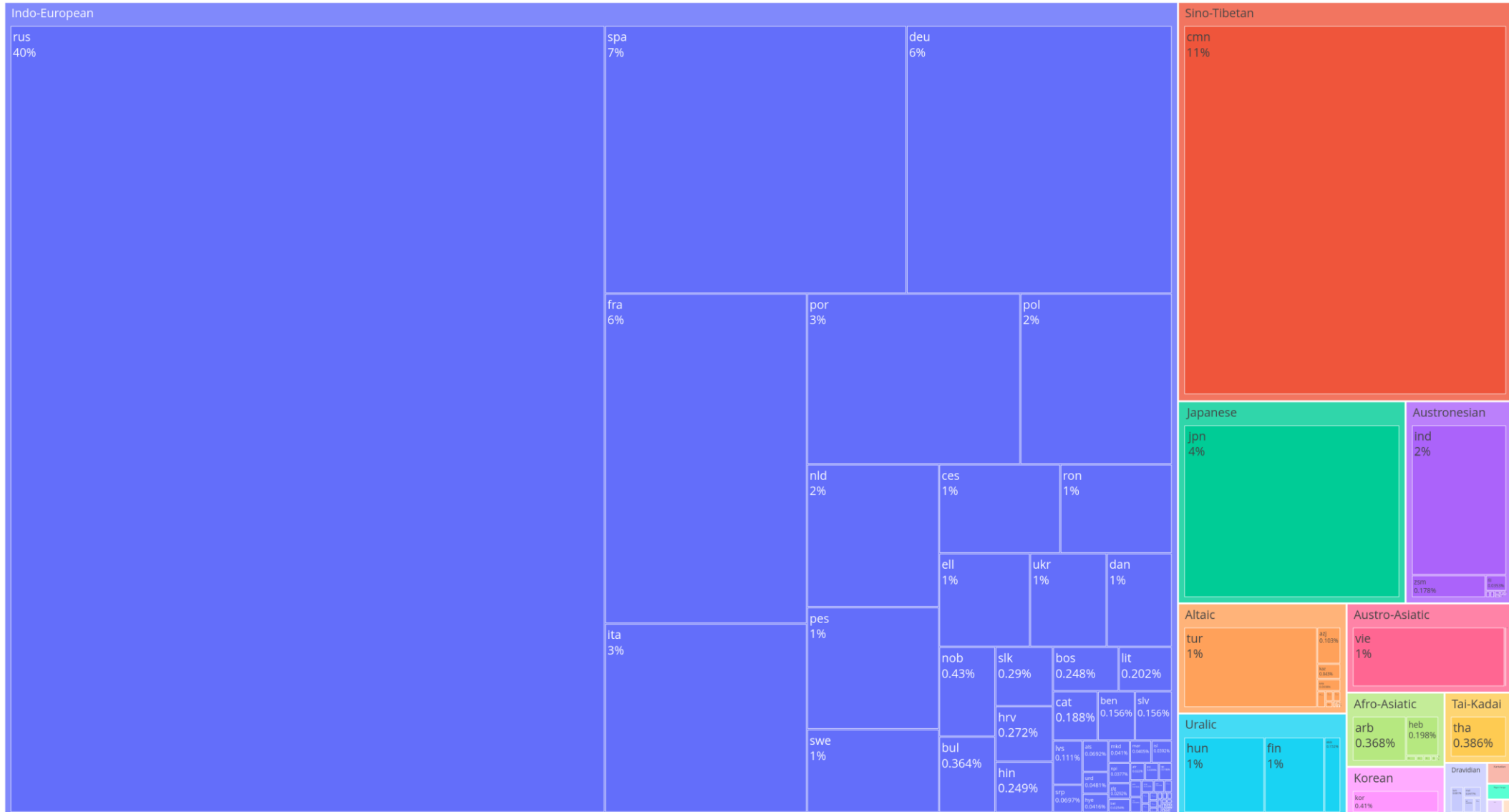
- Multilingual Pretraining: Pretraining a language model on a dataset of text in multiple languages.

da-DK	hvem producere flest pistacienødder i verden
de-{DE,AT,CH}	Wer produziert weltweit die meisten Pistazien
es-ES	¿Quién produce la mayor cantidad de pistachos del mundo?
fi-FI	Kuka tuottaa eniten pistaasipähkinöitä maailmassa
fr-FR	Qui produit le plus de pistaches dans le monde
he-IL	בעולם פיסטוקים הרבה הכי מייצר מי
hu-HU	Ki termeli a legtöbb pisztáciát a világon?
it-IT	Chi produce più pistacchi al mondo
ja-JP	世界で一番ピスタチオを生産しているのは誰ですか
km-KH	អ្នកណាផលិត pistachios ច្រើនជាងគេបំផុតនៅលើពិភពលោក?
ko-KR	전 세계에서 누가 가장 많은 피스타치오를 생산하나요
ms-MY	siapa menghasilkan pistachios paling banyak di dunia
nl-NL	wie produceert de meeste pistachio nootjes ter wereld
nb-NO	hvem lager mest pistasjnøtter i verden
pl-PL	kto produkuje najwięcej pistacji na świecie
pt-BR	quem produz mais pistaches no mundo

Pretraining dataset sizes

- Most pretraining datasets have very limited representation of African languages
- For example, CulturaX has 6.3 trillion tokens in 167 languages
 - The 10 largest languages have > 100 billion tokens
 - Afrikaans is ranked 57 in dataset size with 1.1 billion
 - Malagasy is rank 74 with 142 million
 - Swahili is rank 81 with 30 million tokens
 - No other African languages has a substantial representation (>1M tokens)
- mC4 has around 12 African languages included
 - Afrikaans, Malagasy, Swahili and Somali has >1B tokens, next is Zulu with 200M, and 4 of the bottom 5 languages are African
 - Quality issues have been identified with the African language data in this corpus

HPLT 3.0: 13T non-English tokens



Pretraining data preparation

- A set of best practices have been developed in cleaning web text for pretraining

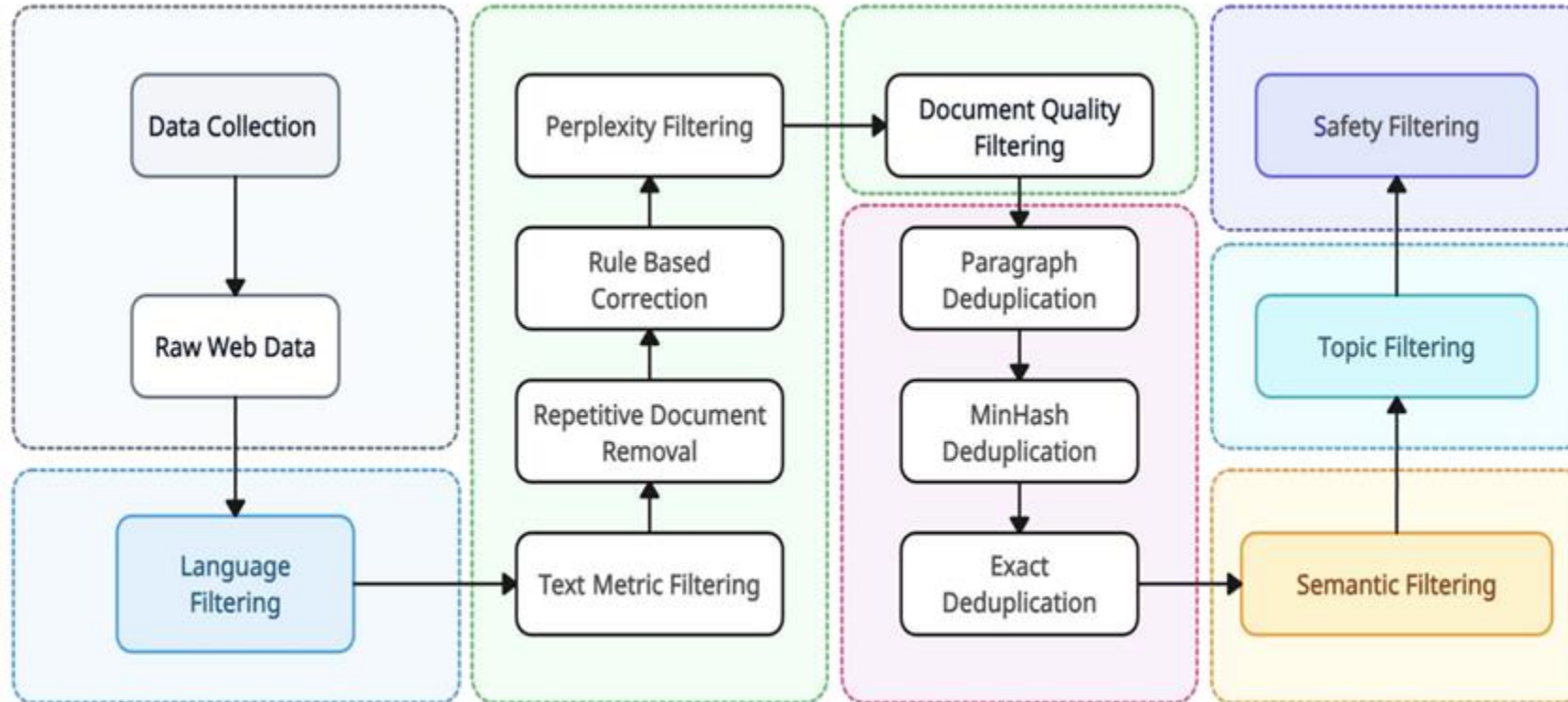
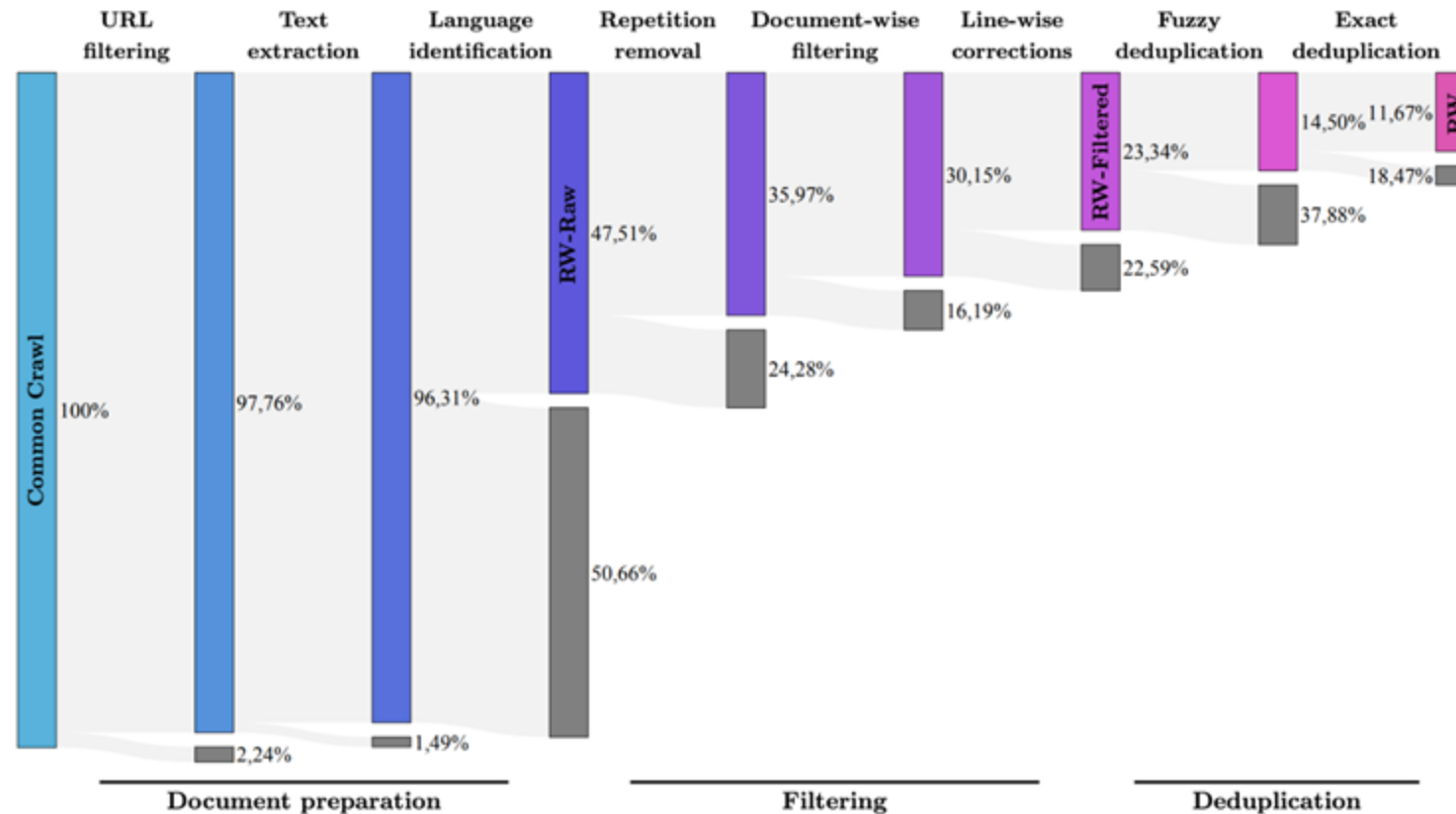


Figure 1: Yi’s pretraining data cleaning pipeline.

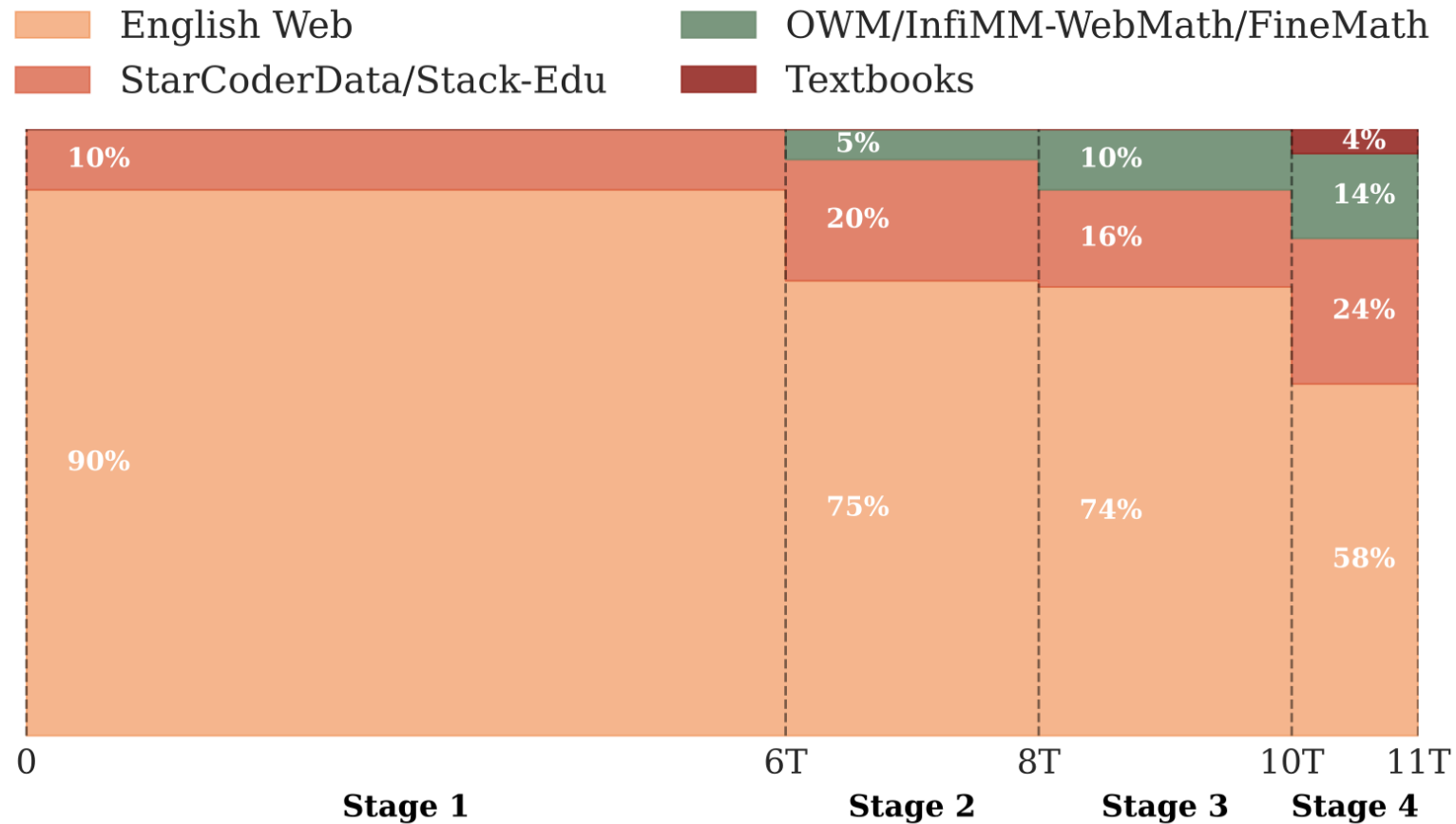
Pretraining data preparation

- Data cleaning reduces the amount of data kept by 90%



Pretraining data preparation

- Typical data-mix for open-source LLM pretraining: SmolLM2



Pretraining data preparation

- Typical data-mix for open-source LLM pretraining: Olmo3

Source	Type	9T Pool		6T Mix		150B Mix	
		Tokens	Docs	Tokens	Docs	Tokens	Docs
Common Crawl	Web pages	8.14T	9.67B	4.51T (76.1%)	3.15B	121B (76.9%)	84.5M
olmOCR Science PDFs	Academic documents	972B	101M	805B (13.6%)	83.8M	19.9B (12.6%)	2.25M
StackEdu (Rebalanced)	GitHub code	137B	167M	409B (6.89%)	526M	11.1B (7.06%)	14.3M
arXiv	Papers with LaTeX	21.4B	3.95M	50.8B (0.86%)	9.10M	1.29B (0.82%)	247K
FineMath 3+	Math web pages	34.1B	21.4M	152B (2.56%)	95.5M	4.10B (2.60%)	2.57M
Wikipedia & Wikibooks	Encyclopedic	3.69B	6.67M	2.51B (0.04%)	4.24M	64.6M (0.04%)	119K
Total		9.31T	9.97B	5.93T (100%)	3.87B	157B (100%)	104M

Pretraining data for South African languages

- We collected text in all 11 written South African languages from multiple pretraining corpora, and then applied quality filtering and deduplication using the Datatrove package from Huggingface (Lombard et al., 2025, work in progress)

Source	Before processing	After deduplication	After filtering	Percent retained
WURA	997,742,420	988,157,747	879,523,389	88.2
mC4	1,008,467,039	979,623,283	824,839,355	81.8
CulturaX	702,050,710	695,083,559	676,035,198	96.3
Glott500	191,154,885	167,600,993	79,244,672	47.3
Inkuba	234,941,750	196,457,594	63,114,818	26.9
CC100	23,822,691	20,291,392	16,922,824	71.1
ParaCrawl	287,212,175	262,079,888	10,139,616	3.5
Corpora	13,473,354	11,372,194	9,840,573	73.0

Pretraining data for South African languages

- Tokens counts per language after filtering:

Language	Train		Validation		Test	
	Tokens	%	Tokens	%	Tokens	%
afr	2,475,913,822	64.96%	1,865,255	14.42%	1,875,605	14.24%
eng	740,994,679	19.44%	1,813,651	14.02%	1,821,803	13.83%
nbl	818,549	0.02%	106,224	0.82%	143,458	1.09%
nso	6,697,358	0.18%	685,425	5.30%	778,656	5.91%
sot	97,558,939	2.56%	2,315,298	17.90%	2,316,170	17.59%
ssw	1,932,989	0.05%	196,247	1.52%	225,810	1.71%
tsn	10,082,930	0.26%	1,216,539	9.41%	1,413,473	10.73%
tso	3,013,408	0.08%	510,463	3.95%	319,496	2.43%
ven	1,852,481	0.05%	191,495	1.48%	243,315	1.85%
xho	152,212,403	3.99%	2,016,503	15.59%	2,012,000	15.28%
zul	320,224,015	8.40%	2,017,406	15.60%	2,021,343	15.35%
TOTAL	3,811,301,573	100.00%	12,934,506	100.00%	13,171,129	100.00%

Pretraining data for South African languages

- In recent work, FineWeb2 and HPLT 3, applies cleaning pipelines directly to CommonCrawl dumps:

Language	MzansiText	Fineweb2 (#M Words)	HPLT3 (#M words)
Afrikaans	2 4775	1 598	1 700
isiZulu	320	71	168
isiXhosa	152	115	131
Sesotho	97	79	128
Setswana		6	12
Sepedi	6	6	9
Ndebele	0.8	1.7	-
SiSwati	1.9	1.4	2.1
Tsonga	3	6.7	12
Venda	1.8	3.3	-

Pretraining data for other Southern/East African languages

Language	FineWeb2 (#M words)	HPLT3 (#M words)
Swahili (swh)	569.5	1,100.0
Kinyarwanda (kin)	127.5	120.0
Chichewa / Nyanja (nya)	62.6	106.0
Shona (sna)	51.9	93.0
Rundi (run)	22.3	88.0
Ganda (lug)	12.3	22.0
Lingala (lin)	11.2	16
Bemba (bem)	1.4	6.2
Tumbuka (tum)	-	5.8

Pretraining data

How can we develop LLMs for African languages given the huge data disparity?

- Develop small models with a more focussed set of capabilities
- Adapt existing English/multilingual LLMs to African languages to leverage cross-lingual transfer
- Collect or create more data

Tokenization

- For language model training text has to be represented as a sequence of tokens from a finite vocabulary
- The simplest approach is to treat each word as a token, but that has a number of limitations such as dealing with rare or unknown words and controlling the vocabulary size
- Instead text is represented a sequence of **subword** tokens, which are sub-parts/pieces of whole words

e.g.

This is a newly spoken sentence.

->

Th is is a new ly spok en se ntence .

Subword tokenization

Why?

- Handle unknown words as a sequence of known subwords.
 - e.g. `newwebsite.com` -> `new website .com` instead of [UNK]
- Compose the meaning of words from subwords (morphemes)
 - e.g. If “`dog`” has only been seen in singular form in training, but “`-s`” has been seen with other plural words, “`dogs`” can be composed as “`dog`” + “`s`”.
- Some languages are morphologically complex - subword units are the fundamental units of meaning.
 - e.g. “`Ndiyabulela`” in isiXhosa = “I am grateful”
 - Ndi : I ya : am bulela : grateful

Subword tokenization

The most widely used tokenization algorithm is **Byte-Pair Encoding (BPE)** (Sennrich et al., 2016)

Type learner:

- Start with a vocabulary consisting of all individual characters and represent the corpus as sequences of items from this vocabulary
= {A, B, C, D, ..., a, b, c, d, ...}
- Repeat until k merges have been done:
 - Choose the two symbols that are most frequently adjacent in the training corpus (say 'A', 'B')
 - Add a new merged symbol 'AB' to the vocabulary
 - Replace every adjacent 'A' 'B' in the corpus with 'AB'

Segmenter algorithm (apply on dataset other than training corpus):

- Run each merge learned from the training data **greedily, in the order** they were learned (test frequencies don't play a role)

Subword tokenization

- Another subword tokenization algorithm is the Unigram Language Modeling (ULM) tokenizer (Kudo, 2018)
- Start with a large vocabulary of substrings from the training corpus
- Assign a (unigram) probability to each vocabulary item based on its frequency
 - This can be used to assign a probability to any possibly tokenization of a word into subwords
 - The most likely (highest probability) tokenization can be found with the Viterbi algorithm
- To train the tokenizer, calculate which tokens' removal will have the least negative effect on the overall probability of the corpus according to the Unigram model
- Iteratively remove tokens until the desired vocab size is reached
- The Unigram tokenizer has been shown to lead to better performance than BPE in lower resource settings in particular, and to produce tokenizations that are closer to languages' morphological structure

Tokenization for African languages

Most South African languages are Niger-Congo B languages

- Agglutinative languages with a rich morphology: words may consist of multiple small meaningful units (morphemes)
- In some languages the morphemes are space-separated (disjunctive, e.g. Sesotho), in others not (conjunctive, e.g. isiXhosa)
- Other Niger-Congo languages have similar challenges

They are sponsored by departments of government (that are) various
Baxhaswe yiminyango kahulumeni eyinhlobonhlobo

POS: PRON VERB AUX NOUN ADP NOUN PRON AUX NOUN

NounClass: B2

B4

B1a

B4

Tokenization example

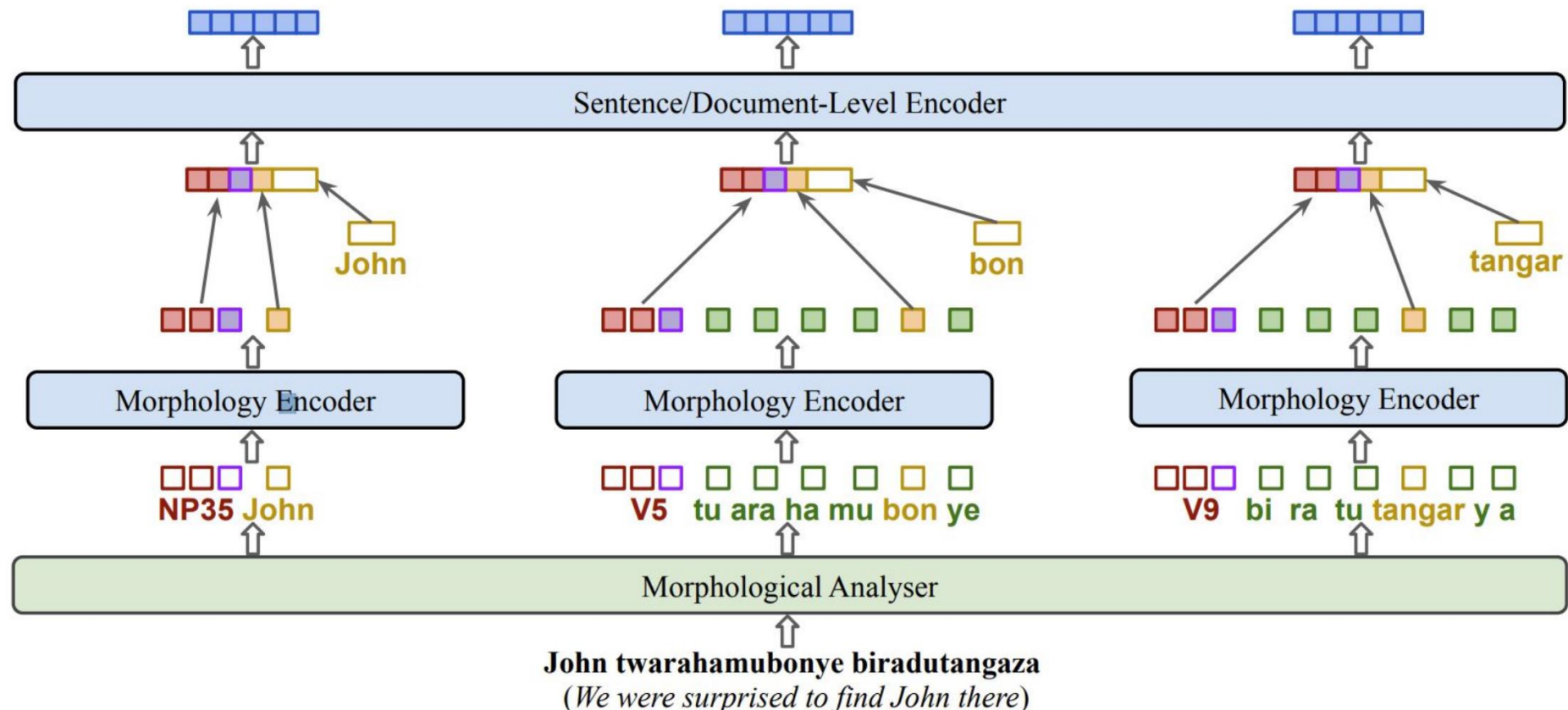
Morphemes	se-si-hamb-e
BPE	sesi-ha-mbe
Unigram LM	se-si-hambe
Morfessor	se-s-ihambe

Tokenization

- Small data sizes and agglutinative language structures both make it harder to learn good tokenizers in a data-driven approach
- An alternative is to use morphological knowledge of the language directly to make tokenization more consistent and meaningful
- One can first apply (supervised) morphological segmentation and then subword tokenization, but that hasn't lead to consistent performance improvements
- An alternative is BPE-knockout (Bauwens and Delobelle, 2024) which eliminates BPE subwords that violate morphological boundaries based on some frequency threshold
- Other approaches modifies the language modelling architecture to incorporate morphological information or to aim to improve tokenization quality

Language modelling for Agglutinative Languages

KinyaBERT: incorporate morphological analysis directly into the language model



Language modelling for Agglutinative Languages

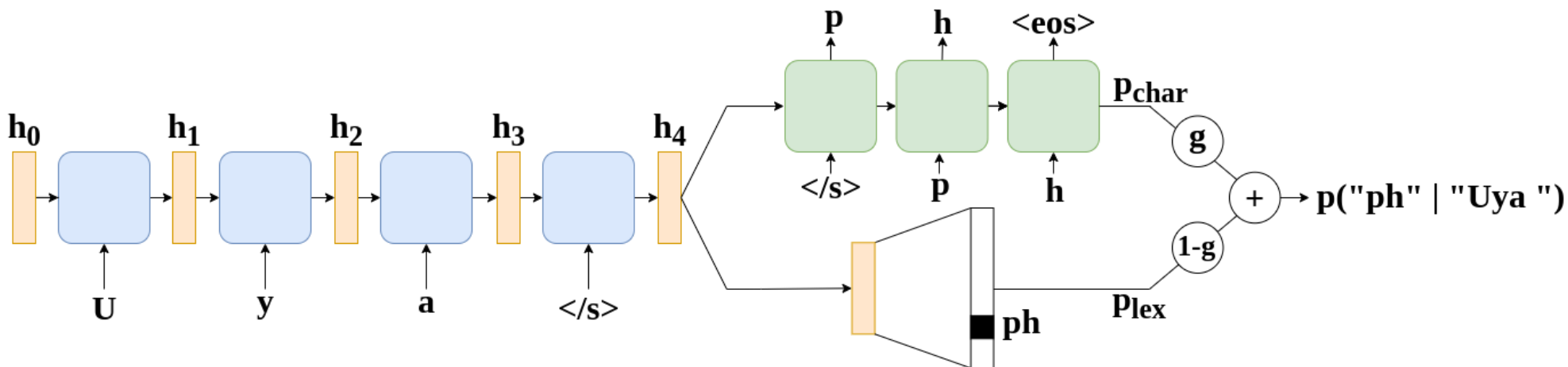
Can a language model's performance be improved by learning subword segmentations that are adapted to optimize the language model's objective and are closer to the morphological structure of the language?

Subword Segmental Language Model (SSLM) (Meyer and Buys 2022)

- Predicts the subword segmentation jointly with the next word in the sequence (i.e., joint segmentation and language modelling objective)
- The model learns the segmentation that will optimize the language model's performance
- Encoder-decoder version uses standard encoder and SSLM decoder (e.g. for machine translation)

Subword Segmental Language Model

- The SSLM generates a sequence of words $w = w_1, w_2, \dots, w_n$. Each word w_i is a sequence of subwords $s_i = s_{i1}, s_{i2}, \dots, s_{i|s_i|}$.



$$p(s_{ij} | s_{\leq i, < j}) = g_k p_{char}(s_{ij} | h_k) + (1 - g_k) p_{lex}(s_{ij} | h_k)$$

- Each segment probability is mixture of the subword lexicon p_{lex} and a character LSTM p_{char}

Subword Segmental Language Model

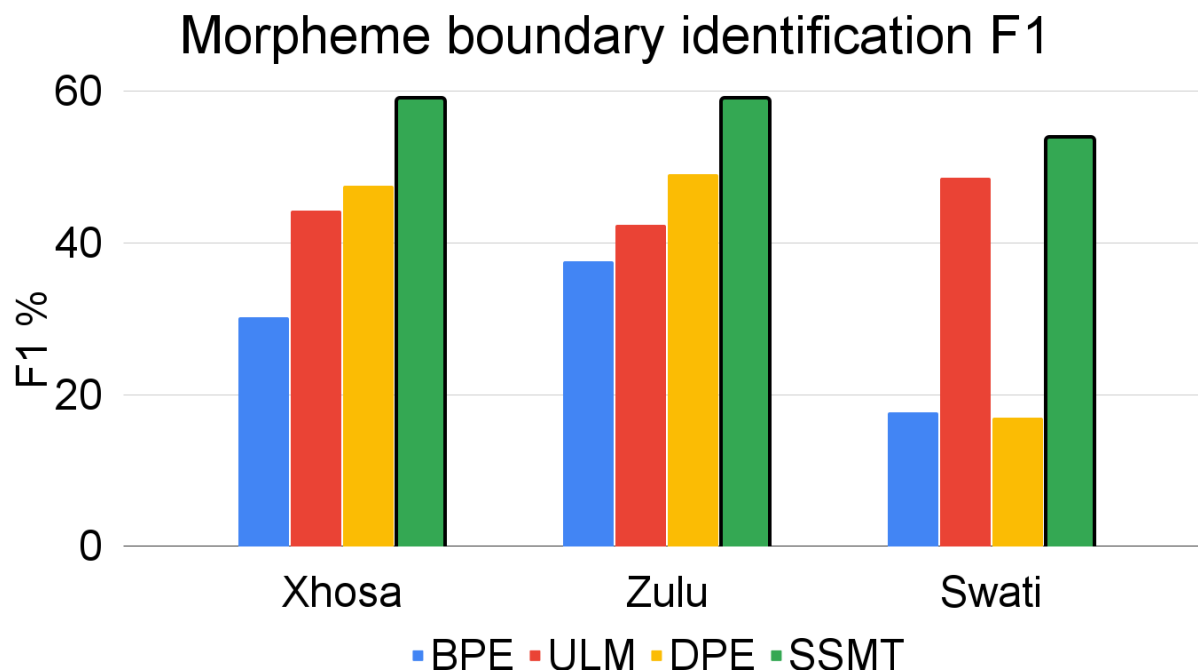
- During training the model marginalizes over all possible word segmentations:

$$p(w) = \sum_{s:\pi(s)=w} \prod_{i=1}^{|w|} \prod_{j=1}^{|s_i|} p(s_{ij} | s_{\leq i, < j})$$

- Semi-Markov assumption: Condition on characters before current segment, not on any previous segment boundaries
- Dynamic programming is used to compute this effectively

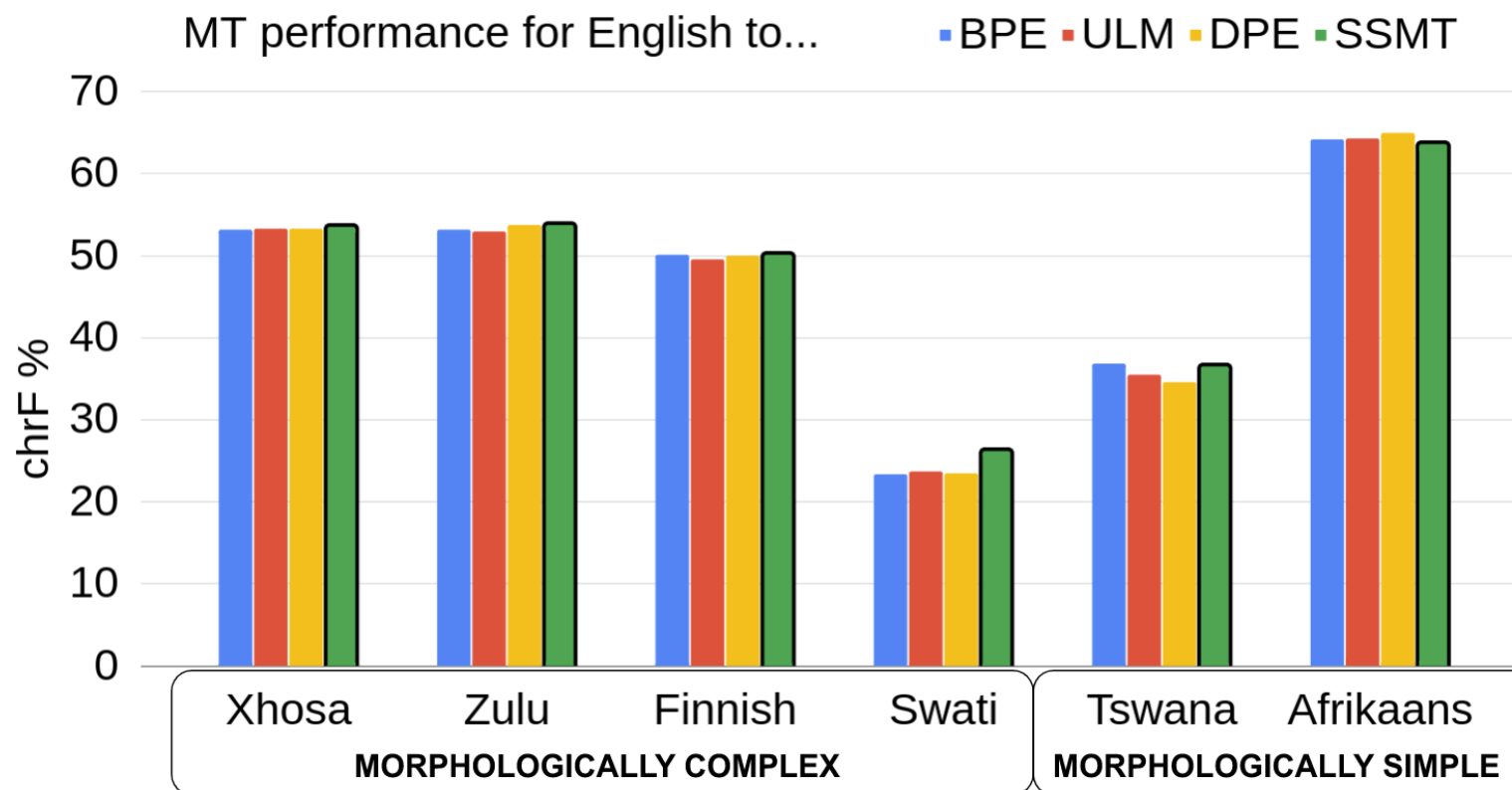
Subword Segmental Language Model

- SSLM learns subwords that are closer to the morphological structure of the language



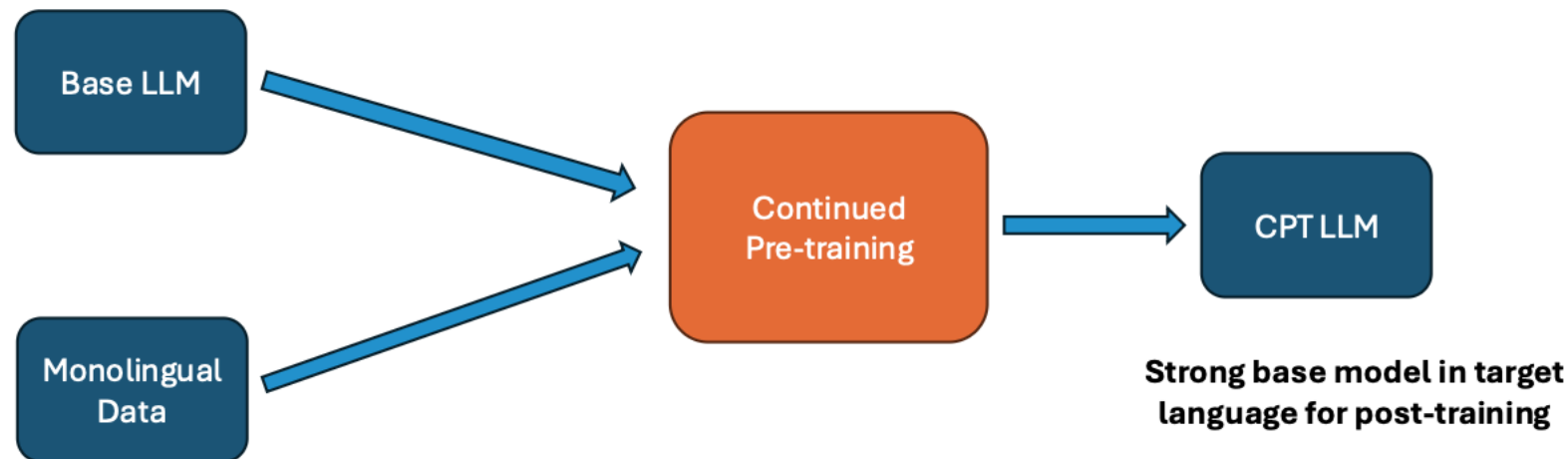
Subword Segmental Language Model

- Better performance in low-resource, morphologically complex settings



Continual pretraining

- We can train language models using African language data only (with some English/French/Portuguese added due to local relevance) but due to the small data size the model is unlikely to have general-purpose capabilities
- The alternative is to take an English-centric or multilingual LLM as starting point and to train it further to adapt or specialize it for one or more target languages
- This can leverage the ability of multilingual models to transfer (some) knowledge across languages



Continual pretraining

- Helps to improve fluency in the target language
- Improves alignment between English and the target language, which lead to better transfer from English
- LLMs are better at using in-language knowledge than knowledge from cross-lingual transfer
- Incorporate cultural-specific knowledge captured in target language corpora only

Continual pretraining for African languages

- Some older multilingual LLMs still perform relatively well on African languages compared to big recent models (requires fine-tuning)
 - Multilingual T5 (mT5): encoder-decoder model trained on mC4 dataset
 - ByT5: Byte-level version
 - XLMR: encoder model (masked LM pretraining)
- AfroXLMR continues XLM-R masked language model pretraining on a corpus of 17 African languages
- Similarly AfriMT5 and AfriByT5 are adaptations of their base models

Continual pretraining

- Continual pretraining for the South African Nguni languages (Meyer et al., 2024)

Existing language models:

- mT5: 101 languages, trained on mC4
- XLM-R: 100 languages, cleaned CommonCrawl
- ByT5: similar to mT5, but byte-level text representation

- AfroLM: 23 African languages
- Afro-XLMR: Adapt XLM-R to 17 African languages
- Afri-ByT5: similar but with ByT5

Continual pretraining

Nguni-XLMR and Nguni-ByT5

- Adapt XLMR and ByT5 on all data from the 4 Nguni languages (only)
- Train models for both NL Understanding (XLM-R) and NL Generation (T5)

Language	xh	zu	nr	ss
Speaker statistics				
L1	8m	12m	2.3m	1.1m
L2	22m	16m	2.4m	1.4m
Pretraining corpus size (tokens)				
XLM-R	13m	0	0	0
ByT5	60m	200m	0	0
Adaptation corpus size (tokens)				
Afro-XLMR	60m	200m	0	0
Afri-ByT5	60m	200m	0	0
Nguni-XLMR/ByT5	60m	200m	450k	500k

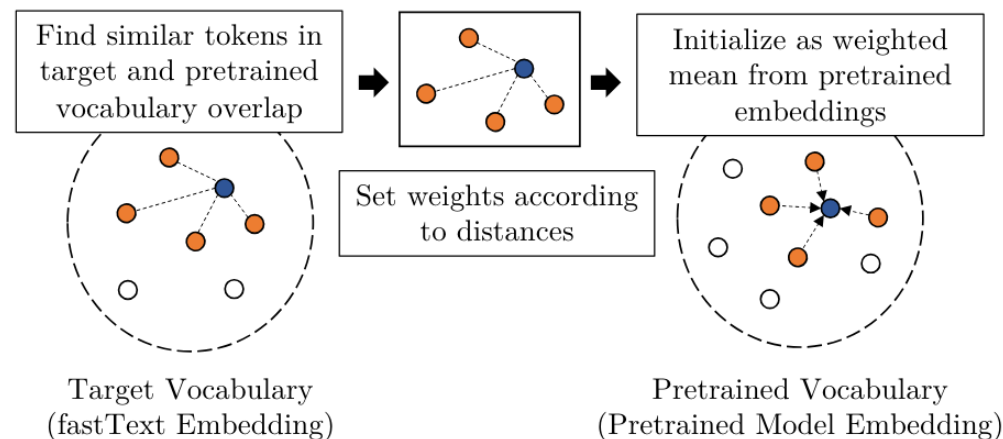
Vocabulary adaptation

Vocabulary adaptation for continual pretraining

- The vocabulary of multilingual LLM cover words in multiple languages
- Tokenizers are trained on pretraining data, so if a language is under-represented or not represented in the pretraining data, it is also going to be underrepresented in the vocabulary
- This can lead to:
 - Higher fertility (average number of tokens per word) -> more computation, lower effective context length
 - Greater inconsistency between the tokenization and language structure
- Several approaches have been proposed to deal with this

Vocabulary adaptation

- For extending the vocabulary to a new language, the best approach is to initialize the embeddings of the new language's tokens in a way that aims to enable cross-lingual transfer
- WECHSEL: target language token embeddings are initialized as a weighted average of source language token embeddings
 - Need multilingual embeddings as starting points
- FOCUS: similar transfer to extend a multilingual vocabulary, based on anchoring overlapping tokens



3. Large Language Modelling Post-training

Fine-tuning and evaluation datasets

- We need datasets that can be used to fine-tune LLMs to perform classical NLP tasks, evaluate the knowledge that they have acquired, and ideally test if they can be used as general instruction following / chat models
- Again, there are limited datasets available to do this for African languages

Fine-tuning and evaluation: NGLUEni Benchmark

NGLUEni: Datasets for fine-tuning and evaluating language models for various understanding and generation tasks in Nguni languages

- Understanding tasks include Named Entity Recognition, Part-of-Speech tagging and topic classification

Task	Dataset	xh	zu	nr	ss	Size
Natural language understanding (NLU)						
NER	MasakhaNER	✓	✓			5783
	SADiLaR NER	✓	✓	✓	✓	6520
POS tagging	MasakhaPOS	✓	✓			753
	NLAPOST	✓	✓	✓	✓	2717
Classification	MasakhaNEWS	✓				1032
	ANTC	✓				2961
	NCHLT Genre	✓	✓	✓	✓	1919
Phrase chunk	NCHLT PC	✓	✓	✓	✓	848
Natural language generation (NLG)						
Data-to-text	T2X	✓				3859
Headline generation	MasakhaNEWS	✓				1032
	Vuk'uzenzele	✓	✓	✓	✓	149

Fine-tuning datasets for SA languages

Task	Language	Train		Validation		Test	
		Ex.	Tokens	Ex.	Tokens	Ex.	Tokens
<i>AfriHG</i>	Xho	10,440	5,892,814	1,305	750,506	1,305	734,845
	Zul	14,209	7,495,625	1,777	952,525	1,776	944,750
<i>T2X</i>	Xho	3,859	346,518	460	44,208	378	44,296
<i>INJONGO Intent</i>	Eng	1,779	398,170	622	139,312	622	139,312
	Sot	2,240	501,824	320	71,720	640	143,370
	Xho	2,240	509,035	320	72,795	640	145,383
	Zul	2,240	508,096	320	72,624	640	145,139
<i>MasakhaNER 2.0</i>	Tsn	3,489	609,722	499	87,620	996	173,044
	Xho	5,718	920,599	817	142,658	1,633	274,254
	Zul	5,848	922,761	836	134,701	1,670	266,979
<i>MasakhaNEWS</i>	Eng	3,309	2,594,818	472	369,539	948	752,019
	Xho	1,032	607,254	147	84,010	297	173,229
<i>SIB-200</i>	Afr	701	78,995	99	10,851	204	22,691
	Eng	701	77,191	99	10,635	204	22,267
	Nso	701	91,974	99	12,468	204	26,812
	Sot	701	87,709	99	12,064	204	25,394
	Xho	701	81,956	99	11,281	204	23,645
	Zul	701	81,336	99	11,198	204	23,429
<i>MasakhaPOS</i>	Tsn	754	462,757	150	88,279	602	342,468
	Xho	752	364,051	150	69,079	601	281,026
	Zul	753	349,628	150	68,687	601	269,956
TOTAL	—	62,868	22,982,833	8,939	3,216,760	14,573	4,974,308

Data-to-Text Dataset

New dataset: **Triples-to-isiXhosa** (T2X)

- Based on triples in the WebNLG data-to-text dataset
- Translated/verbalised from the English WebNLG into isiXhosa text
 - Annotated by 6 postgraduate African language students
- Covers 15 DBPedia categories, 286 relation types

	Train	Valid	Test
WebNLG 1-triples	3 114	392	388
T2X triples	2 413	391	378
T2X verbalisations	3 859	600	888

T2X Text Generation Examples

Data	(a) (South Africa , capital, <i>Cape Town</i>)
Ref	Ikomkhulu lo Mzantsi Afrika li <i>Kapa</i> .
SSPG	I- <i>Cape Town</i> likomkhulu lase South Africa .
PG	U <i>Cape Town</i> likomkhulu lase- Afrika .
BPEMT	Ikomkhulu lo Mzantsi Afrika yi <i>Kapa</i> .
Data	(c) (Ethiopia , leaderName, <i>Mulatu Teshome</i>)
Ref #1	UMulatu Teshome yinkokheli yase- Ethiopia .
Ref #2	Igama lenkokheli e- Ethiopia ngu <i>Mulatu Teshome</i> .
SSPG	UMulatu Teshome yinkokeli yase- Ethiopia .
PG	Inkokeli yase- Ethiopia ngu <i>Mulatu Teshome</i> .
BPEMT	UMulatu Teshome yinkokeli yase- Ethiopia .

Instruction fine-tuning

Instead of fine-tuning a model separately for each task, we want it to be able to follow instructions specifying the task, while leveraging knowledge from pretraining

- How to we get a language model to follow instructions?
- Using a pretrained language model directly is not enough

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION

GPT-3

Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.

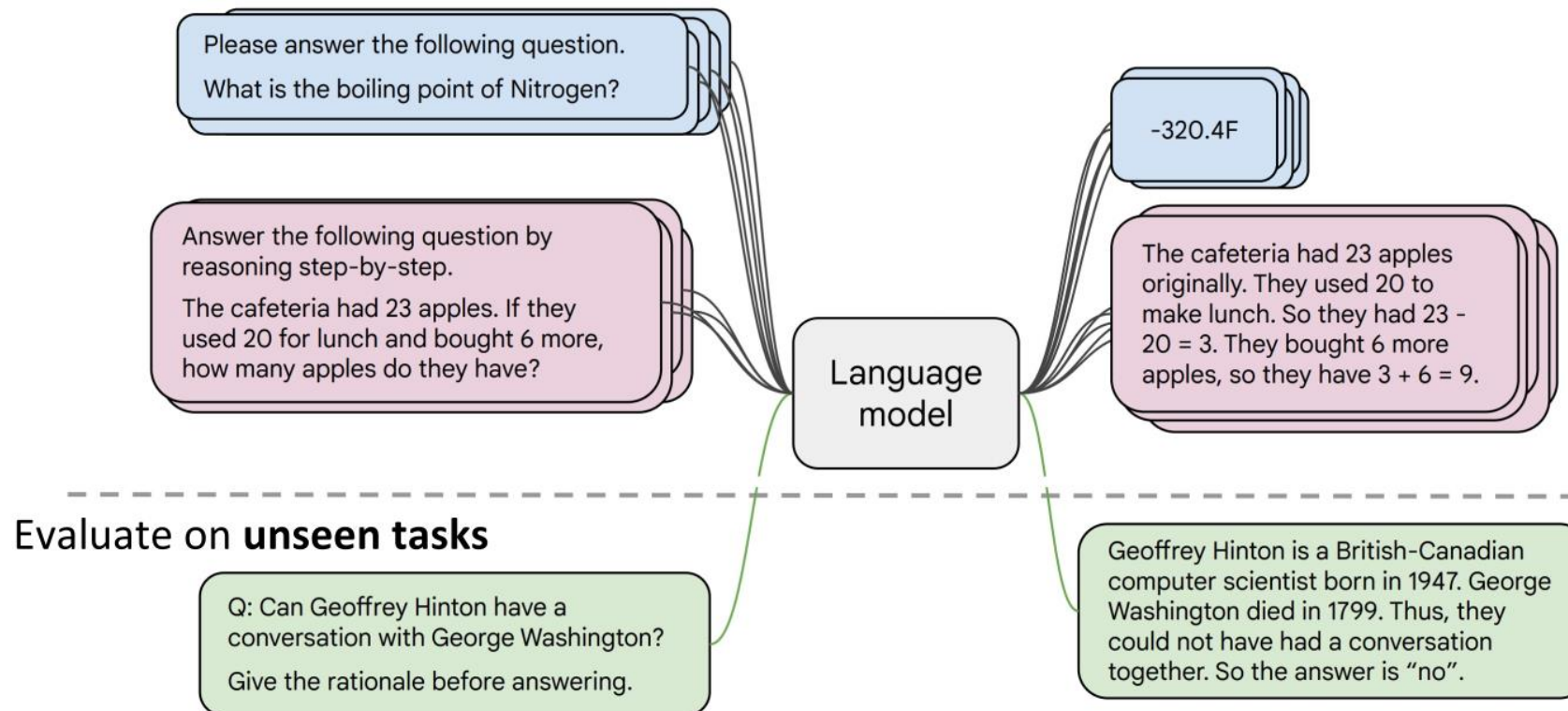
Instruction fine-tuning

Without fine-tuning the language model is not going to have any particular goal in mind, it will just generate high-probability words, which can lead to various problems:

- Generating factually incorrect outputs
- Generating obscene, biased or harmful statements
- No control over how specific or sensible the output is
- Not “understanding” a user’s request because it didn’t appear in this format in the training data
- Lack of “alignment” with human values

Instruction fine-tuning

- Collect examples of (instruction, output) pairs across many tasks and fine-tune an LM



Instruction fine-tuning

Instruction tuning datasets have been created with large numbers of tasks and examples:

- xP3: 17 tasks, 46 languages
- Aya dataset: 65 languages, 204K instances
- SmolTalk: synthetic instruction data (English)

Very limited coverage of African languages

Instruction fine-tuning example

Model input (Disambiguation QA)

Q: In the following sentences, explain the antecedent of the pronoun (which thing the pronoun refers to), or state that it is ambiguous.

Sentence: The reporter and the chef will discuss their favorite dishes.

Options:

- (A) They will discuss the reporter's favorite dishes
- (B) They will discuss the chef's favorite dishes
- (C) Ambiguous

A: Let's think step by step.

Before instruction finetuning

The reporter and the chef will discuss their favorite dishes.

The reporter and the chef will discuss the reporter's favorite dishes.

The reporter and the chef will discuss the chef's favorite dishes.

The reporter and the chef will discuss the reporter's and the chef's favorite dishes.

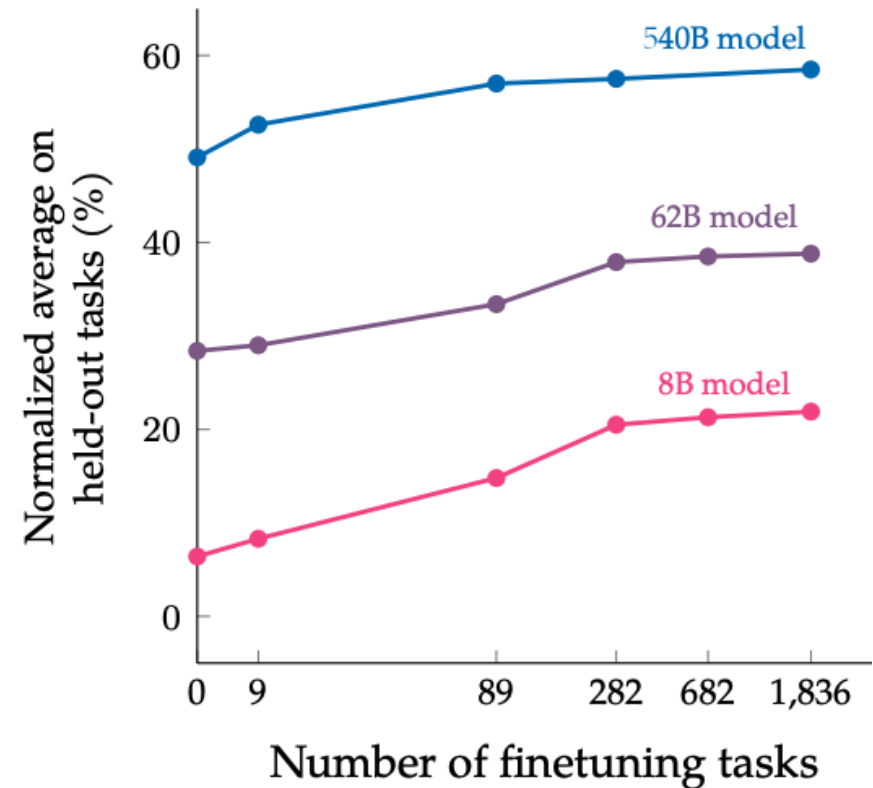
✗ (doesn't answer question)

After instruction finetuning

The reporter and the chef will discuss their favorite dishes does not indicate whose favorite dishes they will discuss. So, the answer is (C). ✓

Scaling instruction fine-tuning

- Instruction finetuning improves performance by a large margin compared to no finetuning
- Increasing the number of finetuning tasks improves performance
- Increasing model scale by an order of magnitude (i.e., 8B \rightarrow 62B or 62B \rightarrow 540B) improves performance substantially for both finetuned and non-finetuned models



Instruction tuning

- Most work on LLMs for African language focus on continual pretraining rather than instruction tuning
- AfrilInstruct did investigate instruction tuning for African languages, however most gains still came from continual pretraining rather than actual instruction tuning
- There are still limited instruction tuning datasets available for African languages: Data that has been used either cover a limited number of tasks and not general instruction following, or are automatically translated

Source Data	Task	of Tokens	of Prompts	of Languages
MasakhaNEWS	News Topic Classification	6,154,176	90,890	eng, fra, amh, hau, ibo, orm, sna, som, swa, tir, xho, yor
MasakhaPOS	Part-of-Speech Tagging	1,780,578	6,879	hau, ibo, kin, nya, sna, swa, xho, yor, zul
AfriSenti	Sentiment Analysis	19,201,035	235,225	amh, hau, ibo, yor, por, kin, swa
NollySenti	Sentiment Analysis	1,213,691	15,100	hau, ibo, eng, yor
xP3	xP3 - Multitask	640,745,532	7,773,312	eng, ara, ibo, hau, kin, nya, sna, sot, swa, xho, yor, zul
xP3	xP3 - Question Answering	146,758,736	541,630	eng, ara, ibo, hau, kin, nya, sna, sot, swa, xho, yor , zul
FLORES	Translation	5,692,402	72,324	eng, fra, afr, amh, ara, hau, ibo, kin, nya, por, som, sna, sot, swa, tir, xho, yor, zul
MAFAND	Translation	4,467,767	66,234	eng, amh, hau, ibo, kin, nya, sna, swa, xho, yor, zul
MasakhaNER2.0	Named Entity Recognition	12,935,191	58,667	hau, ibo, kin, nya, sna, swa, xho, yor, zul
MENYO	Translation	1,225,883	16,703	eng, yor
XL-Sum	Summarization	32,814,291	72,124	eng, amh, ara, hau, ibo, orm, por, swa, tir, yor

Instruction tuning

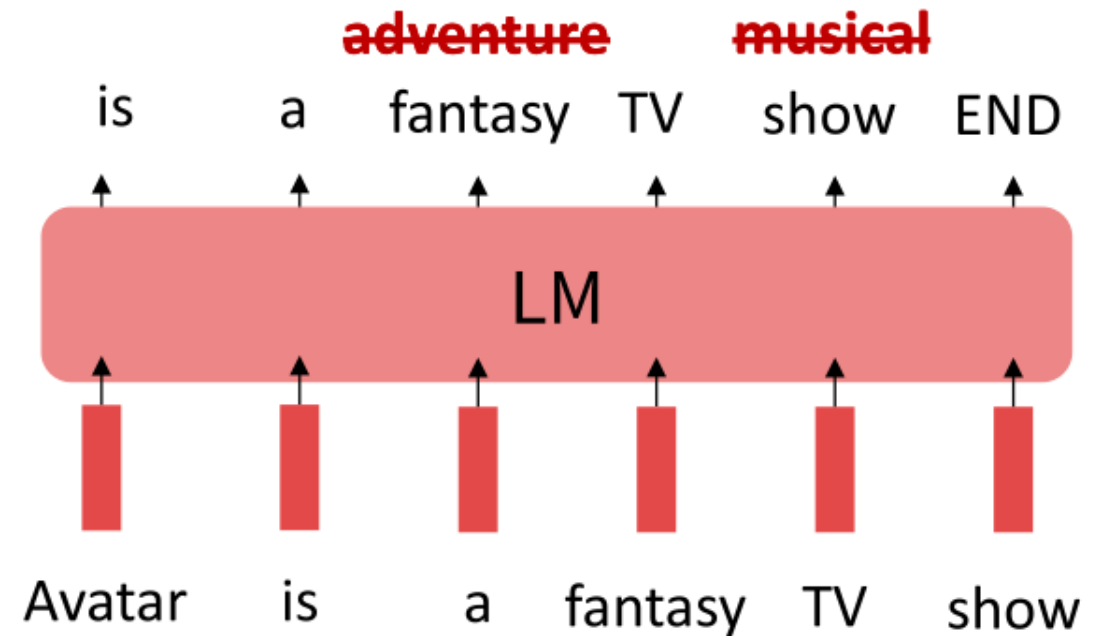
Prompt Templates for AfrInstruct (Uemura et al., 2024)

Task	Prompt
Machine Translation	Translate the following text from {source language} to {target language}. {source language}:{source texts}. {target language}:
Named Entity Recognition	Study this taxonomy for classifying named entities:- LOC (Location or physical facilities)- ORG (Organizations, corporations or other entities)- PER (Names of people)- DATE (Date or time)Identify all named entities in the following tokens:{split tokens} Additionally, you should add B- to the first token of a given entity and I- to subsequent ones if they exist. For tokens that are not named entities, mark them as O.Answer:
News Topic Classification	Which of these labels best describes this news article:{topic candidates}{target sentence} Label:
Part-of-Speech Tagging	Study this taxonomy for classifying parts of speech:- X: Other- ADJ: Adjective- ADP: Adposition- ADV: Adverb- AUX: Auxiliary verb- CCONJ: Coordinating conjunction- DET: Determiner- INTJ: Interjection- NOUN: Noun- NUM: Numeral- PART: Particle- PRON: Pronoun- PROPN: Proper noun- PUNCT: Punctuation- SCONJ: Subordinating conjunction- SYM: Symbol- VERB: VerbPerform Part-of-Speech (POS) tagging on the following tokens: {split tokens} Answer:
Sentiment Analysis	Analyze the sentiment expressed in the following tweet' { text } 'Options: positive, negative, neutral
Summarization	{ passage } Write a summary of the text above in { target language}:

Instruction fine-tuning

Limitations of instruction fine-tuning:

- Expensive to collect annotations
- Some tasks, e.g. open-ended creative generation, have no right answer
- Some errors are worse than others, but the LM objective cannot capture this
- Can we explicitly attempt to satisfy human preferences?



Optimizing for human preferences

- For every output sample s of a LM on some task, obtain a human reward $R(s)$ – higher is better
- Example: Summarization

SAN FRANCISCO,
California (CNN) --
A magnitude 4.2
earthquake shook the
San Francisco
...
overturn unstable
objects.

An earthquake hit
San Francisco.
There was minor
property damage,
but no injuries.

$$s_1$$
$$R(s_1) = 8.0$$

The Bay Area has
good weather but is
prone to
earthquakes and
wildfires.

$$s_2$$
$$R(s_2) = 1.2$$

Reinforcement Learning from Human Feedback (RLHF)

1. Collect instruction / response data and train a supervised model (policy)
 - Supervised instruction fine-tuning
2. Collect human preference data ranking or scoring multiple possible responses to instructions
 - This data is used to train a reward model to emulate the human judgements
3. Optimize a policy against the reward model using reinforcement learning
 - Generate responses using the current policy (the LLM)
 - Score responses with the reward model
 - Use reinforcement learning to update the policy

Reinforcement Learning from Human Feedback (RLHF): Reward modelling

- Human-in-the-loop is expensive: rather model human preferences as a separate problem
- Train a reward model LM to predict human preferences from an annotated dataset, then use RL to optimize the policy (main LM) for the reward model instead
- Pairwise comparisons can be more reliable than asking for direct ratings

An earthquake hit
San Francisco.
There was minor
property damage,
but no injuries.

>

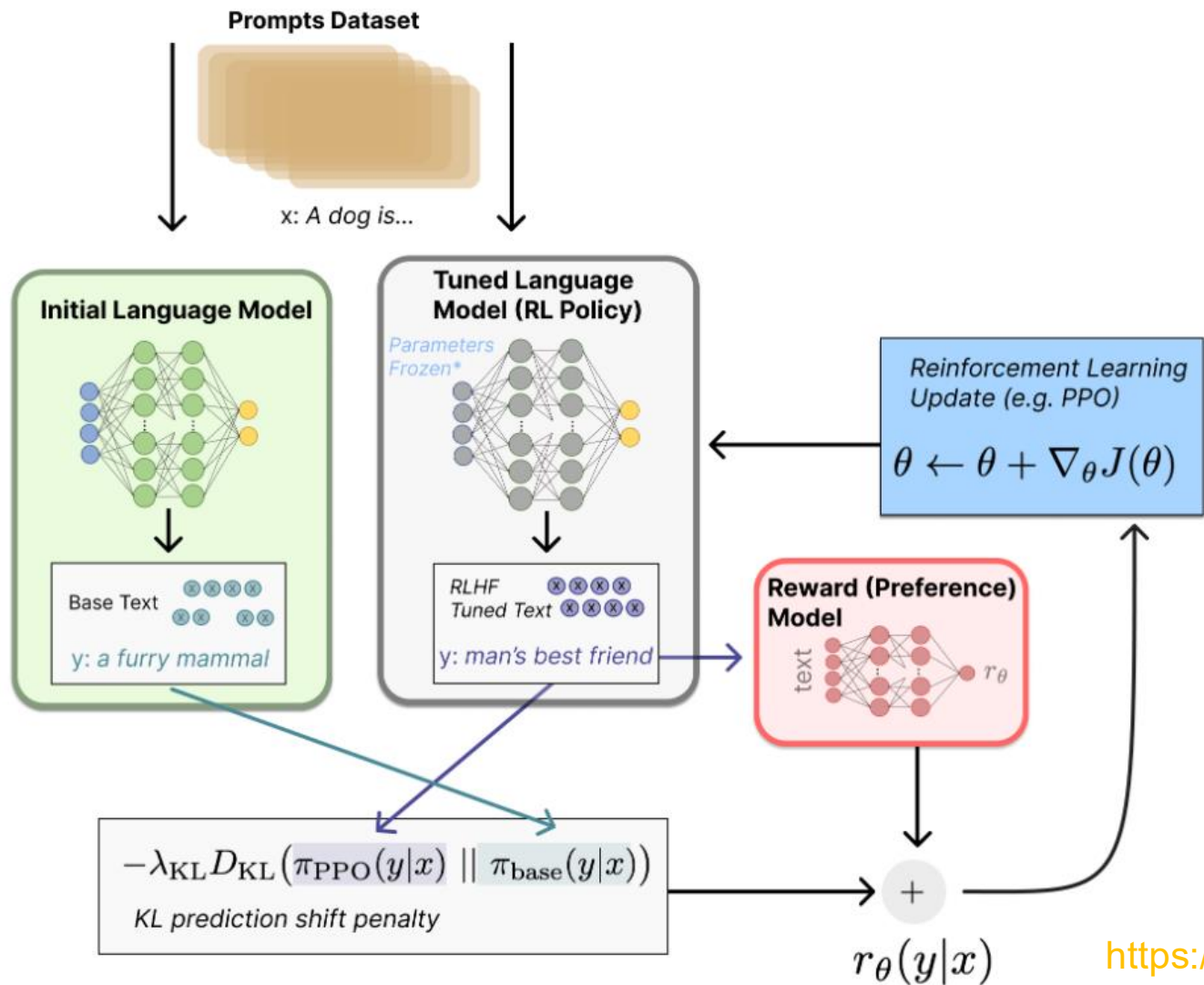
A 4.2 magnitude
earthquake hit
San Francisco,
resulting in
massive damage.

>

The Bay Area has
good weather but is
prone to
earthquakes and
wildfires.



Reinforcement Learning from Human Feedback (RLHF): RL fine-tuning



Reinforcement Learning from Human Feedback (RLHF)

- Stylistic changes after RLHF

Instruction: What are the five most common causes of stress among people?

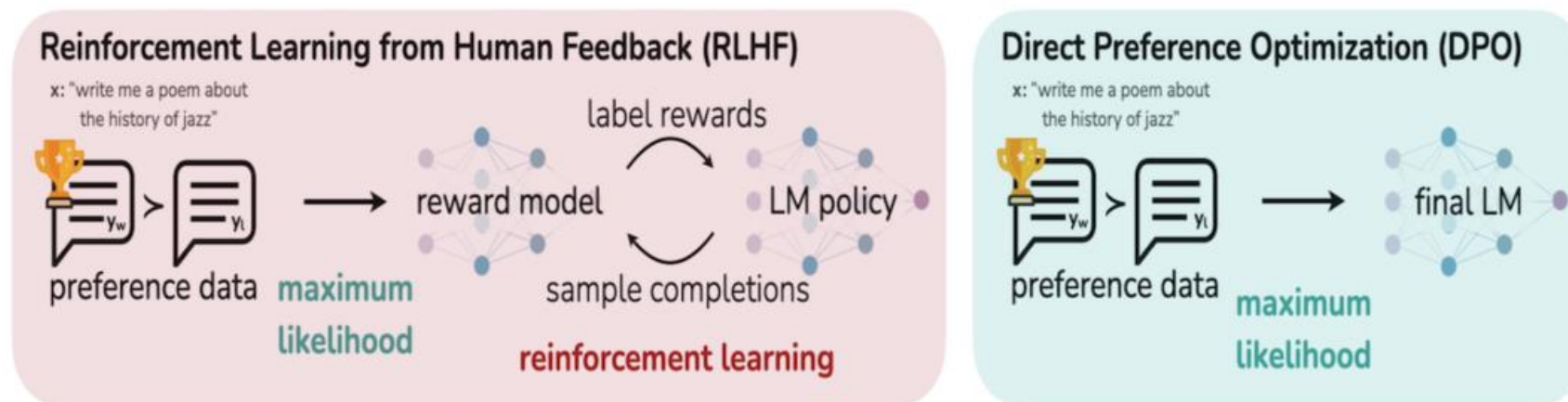
Alpaca (Finetuned w/ 10k data): The five most common causes of stress among people are work, money, relationships, health, and family

Alpaca + PPO: The five most common causes of stress among people are:

1. Pressure at work: Whether it is due to tight deadlines, high expectations, or an overwhelming workload, pressure at work can be a major source of stress
2. Money problems: Financial issues can be a significant source of stress, from not being able to make ends meet to struggling with debt.
3. Unhealthy lifestyle: Poor diet, lack of exercise, and sleep deprivation can all contribute to stress.
4. Unhappiness at home: Problems in relationships, conflicts with family members, and other domestic issues can be a major source of stress.
5. Uncertainty: Excessive worry and anxiety about the future, health issues, or other unknowns can be a major cause of stress

Direct Preference Optimization (DPO)

- Replace the complex RL part with a simple weighted MLE objective

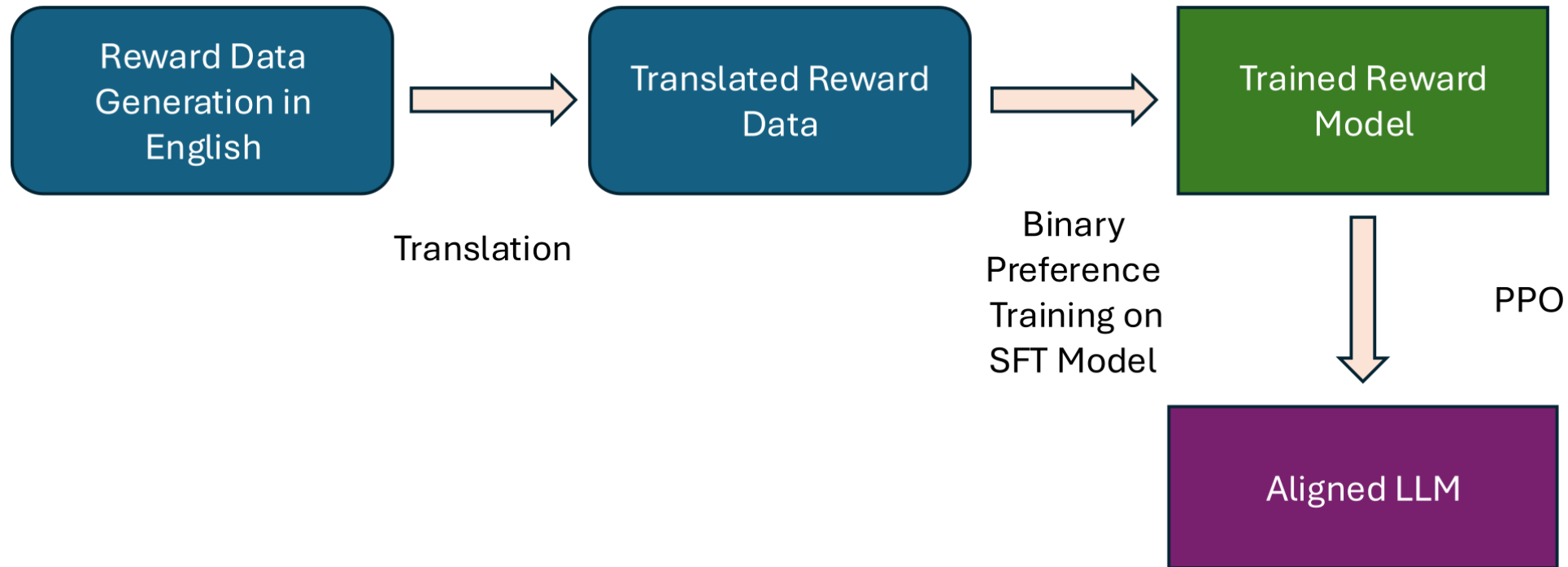


$$\nabla_{\theta} \mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) = -\beta \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\underbrace{\sigma(\hat{r}_{\theta}(x, y_l) - \hat{r}_{\theta}(x, y_w))}_{\text{higher weight when reward estimate is wrong}} \left[\underbrace{\nabla_{\theta} \log \pi(y_w | x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_{\theta} \log \pi(y_l | x)}_{\text{decrease likelihood of } y_l} \right] \right],$$

$$\hat{r}_{\theta}(x, y) = \beta \log \frac{\pi_{\theta}(y|x)}{\pi_{\text{ref}}(y|x)}$$

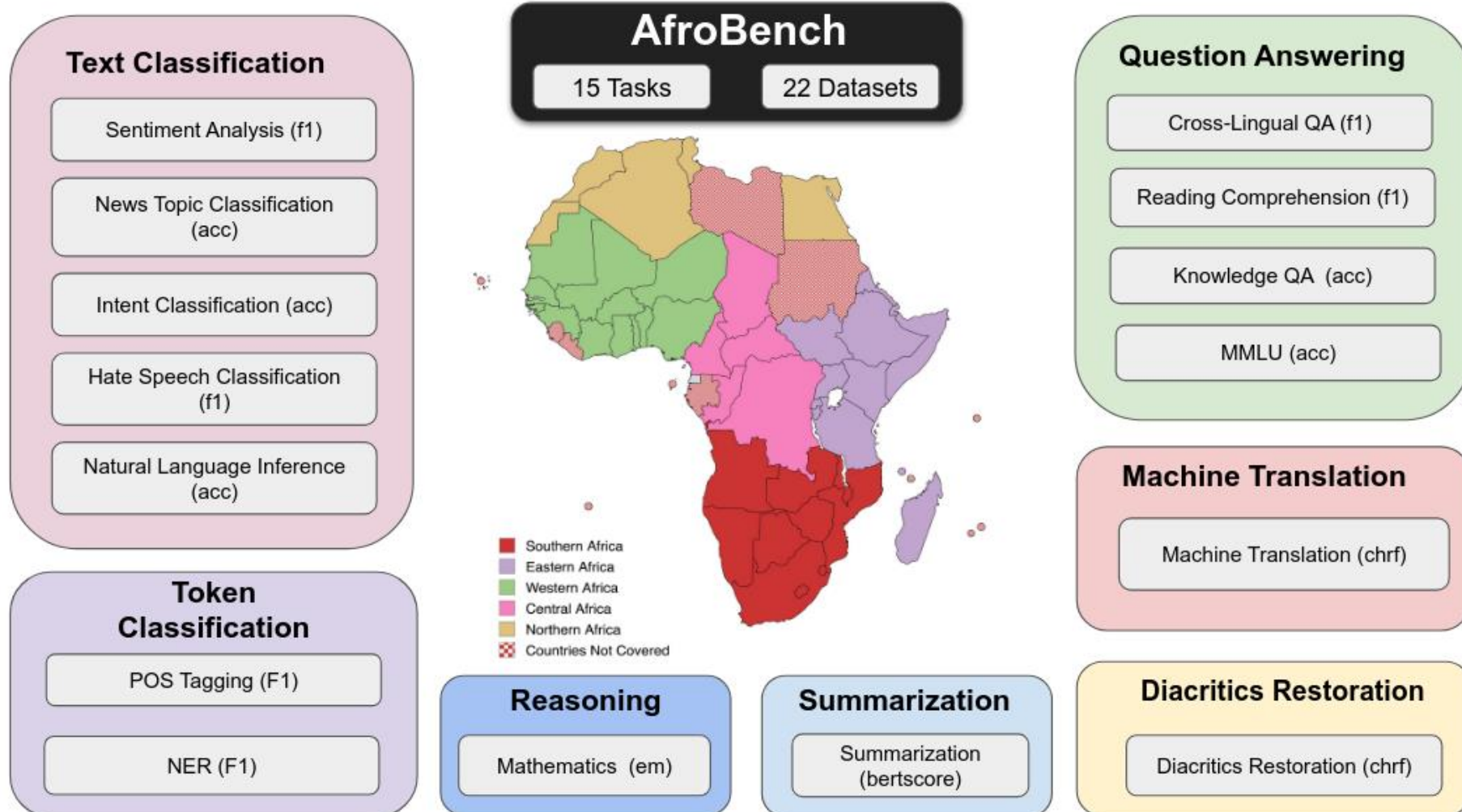
Reinforcement Learning from Human Feedback (RLHF)

- RHLF with translated data



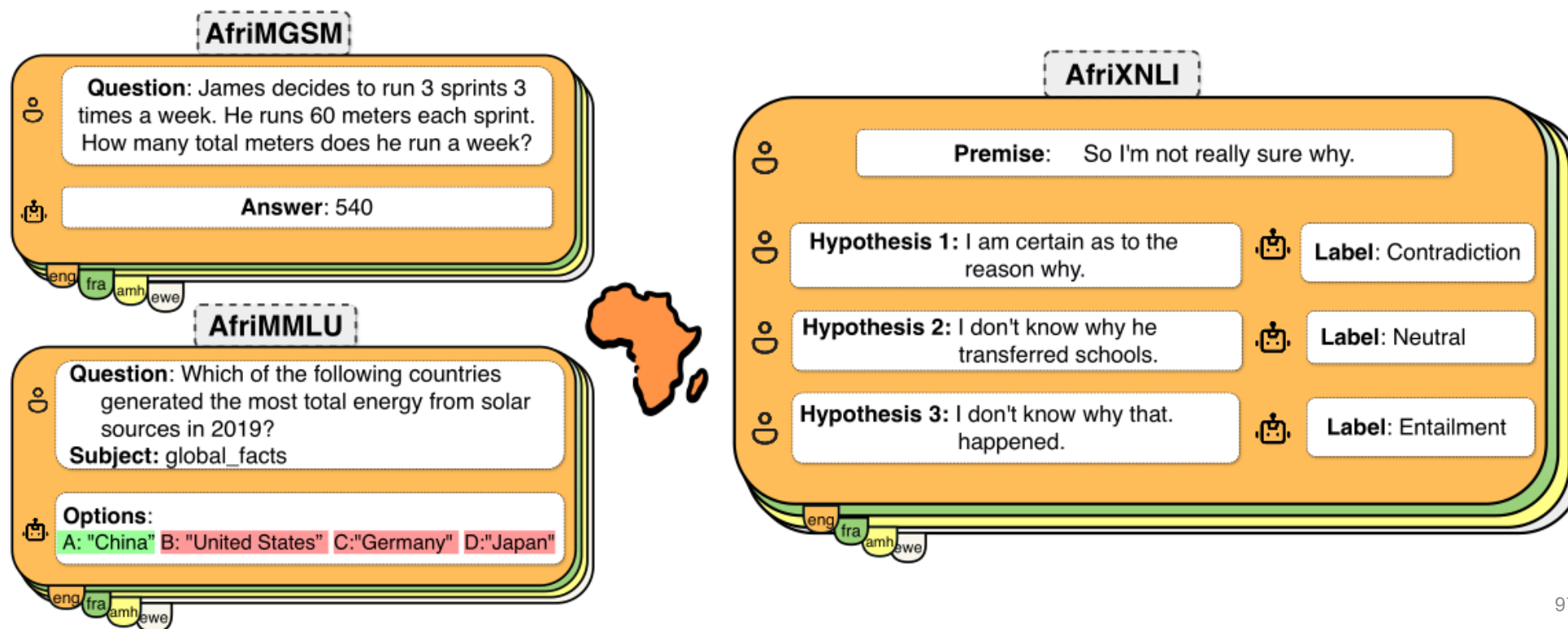
AfroBench (Oja et al.)

Evaluation benchmark covering 64 African languages



Irokobench (Adelani et al., 2024)

Evaluation benchmark for 16 African languages

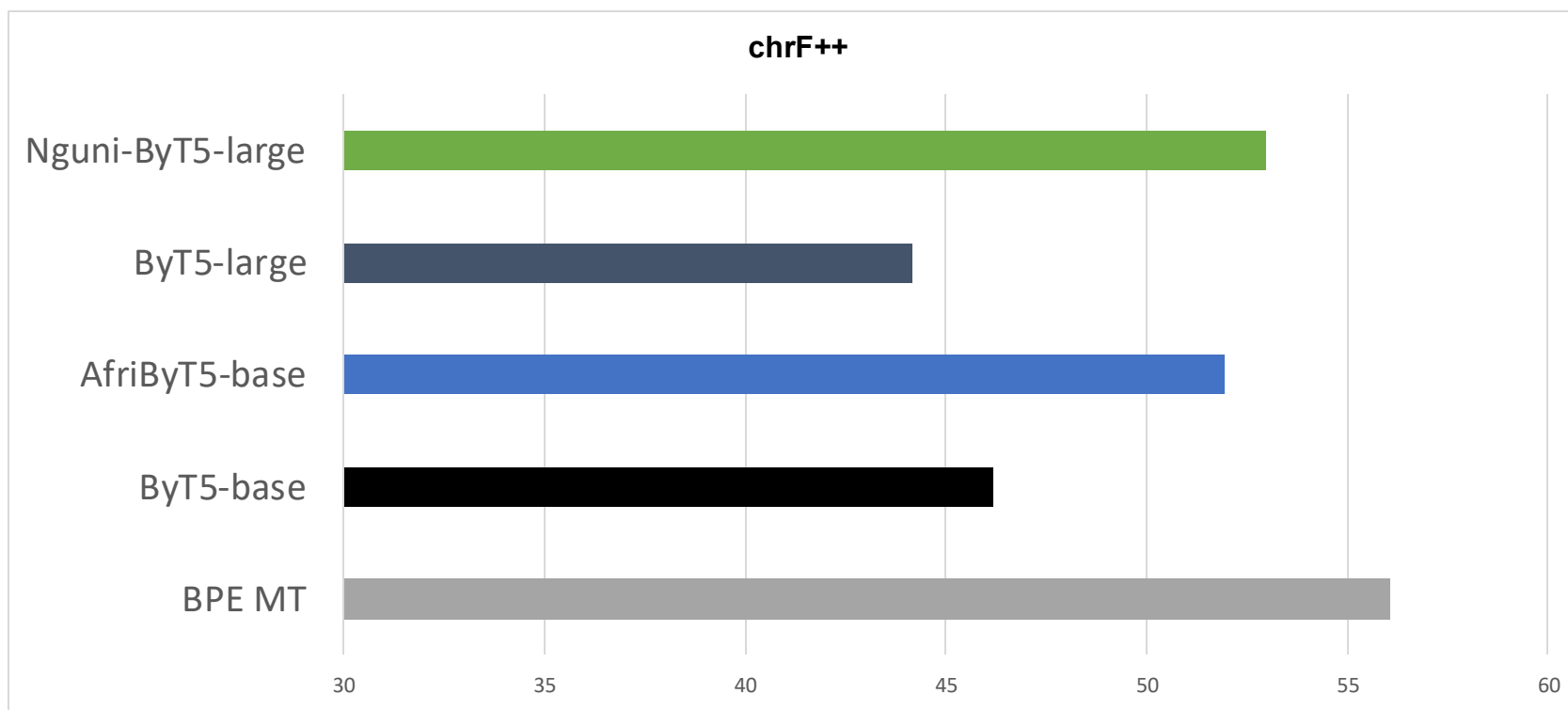


LLMs for African languages: Nguni-XLMR and Nguni-ByT5

Task	Dataset	lang	XLM-R-large	Afro-XLMR-large	Nguni-XLMR-large
NER	MasakhaNER	xh	88.1	89.9	90.4 ± 0.004
		zu	86.7	90.6	91.8 ± 0.006
	SADiLaR NER	xh	74.8 ± 0.7	76.3 ± 0.9	77.3 ± 0.5
		zu	73.6 ± 0.3	74.1 ± 0.6	74.3 ± 0.4
		nr	78.6 ± 0.2	79.4 ± 0.4	79.1 ± 0.7
		ss	71.8 ± 0.6	72.8 ± 0.4	74.1 ± 0.7
POS	MasakhanePOS	xh	88.1	88.7	88.3 ± 0.1
		zu	89.4	90.1	90.1 ± 0.1
	NLAPOST	xh	97.1 ± 0.1	97.8 ± 0.1	97.9 ± 0.1
		zu	92.5 ± 0.2	92.9 ± 0.2	93.3 ± 0.1
		nr	90.3 ± 0.1	90.5 ± 0.1	90.6 ± 0.2
		ss	90.9 ± 0.3	91.0 ± 0.1	91.6 ± 0.3
Classification	MasakhaneNEWS	xh	89.2	97.3	98.2 ± 0.5
	ANTC	zu	78.7	81.6 ± 1.4	86.8 ± 0.6
	NCHLT Genre	xh	89.1 ± 0.9	89.0 ± 1.0	88.8 ± 0.6
		zu	82.8 ± 1.4	84.9 ± 1.2	86.5 ± 1.7
		nr	96.4 ± 2.6	94.9 ± 0.6	95.2 ± 0.6
		ss	96.3 ± 1.4	96.7 ± 0.8	96.0 ± 0.6

T2X Data-to-Text Generation Results

- Pretrained T5 language models vs fine-tuning a machine translation model (BPE MT)



LLMs for African languages: Language adaptation

- AfroLlama: Continual fine-tuning from Llama3
- Lugha-Llama: Continual fine-tuning: Adds high-quality educational documents and translate them into Swahili
 - Recent trend to use synthetic data for pretraining

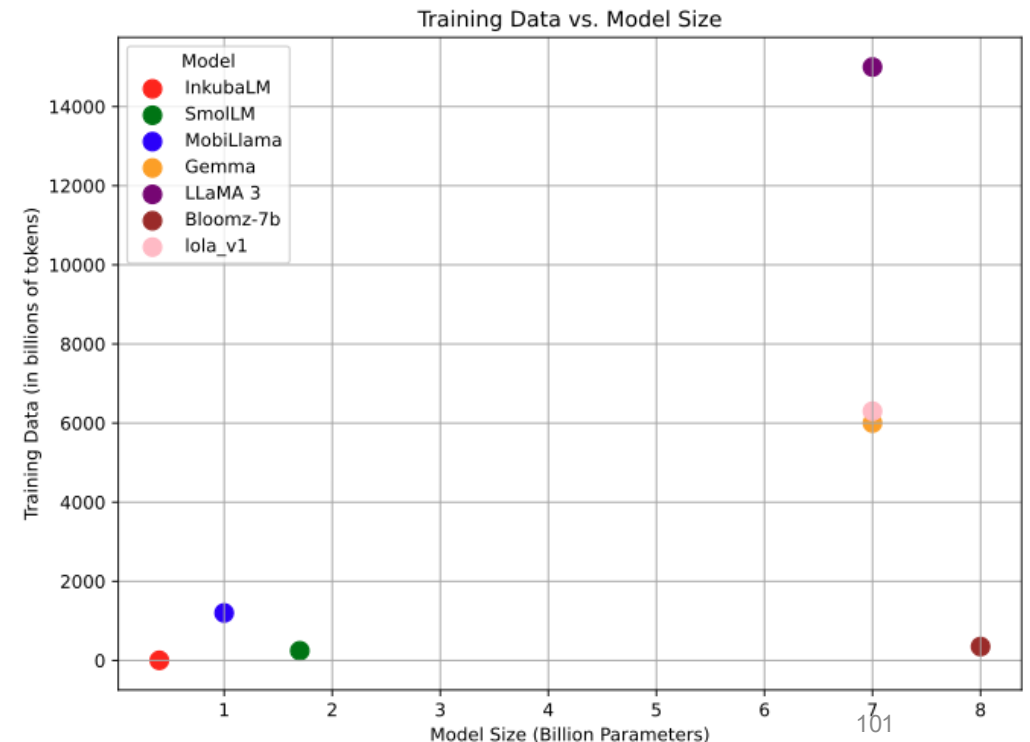
Model	Size	Average Score
Llama-3.1	8B	20.1
Lugha-Llama	8B	34.2
Lugha-Llama-Edu	8B	30.3
Lugha-Llama-Math	8B	37.7
AfroLlama-V1	8B	19.0
AfriInstruct	7B	21.9

Training Data	AfrimMLU		
	eng	swa	Avg [†]
100% WURA _{swa}	64.4	41.0	30.6
60% WURA _{swa} + 40% FW-Edu	66.6	42.6	31.6
60% WURA _{swa} + 40% FW-Edu _{swa}	65.4	46.0	31.5
100% FW-Edu _{swa}	66.2	43.8	31.5

LLMs for African languages from scratch

- Serengeti, Cheetah: Use large datasets but not publicly available
- InkubaLM: Trained on 2.4B tokens covering 5 African languages

Model	swa	hau	yor	AVG
<i>Prompt LLMs in English Language</i>				
InkubaLM-0.4B	42.47	22.25	28.08	30.93
SmolLM-1.7B	26.09	31.97	28.36	28.80
MobiLlama-1B	37.2	34.53	32.89	34.87
Gemma-7B	14.42	36.16	26.17	25.58
LLaMa 3-8B	19.48	32.44	29.77	27.23
BLOOMZ-7B	17.26	33.81	32.99	28.02
lola_v1-7.4B	14.4	26.71	28.16	22.42



LLM for South African languages from scratch

- MzansiLM: Train 125M parameter decoder model on data from South African languages only (3.8B tokens)

Model / Variant	Size	Eng	Xho
MzansiLM			
<i>Base 0-shot</i>	0.125B	38.3	39.1
<i>mono-masakhanews-ft</i>	0.125B	63.5	73.2
<i>multi-masakhanews-ft</i>	0.125B	60.8	78.5
<i>general-ft</i>	0.125B	49.2	50.9
<hr/>			
<i>Decoder-Only</i>			
InkubaLM-0.4B	0.4B	20.3	7.4
AfroLlama-V1	8B	67.1	50.2
Llama-3.1-70B-Instruct	70B	83.3	68.4
<hr/>			
<i>Encoder-Decoder</i>			
Aya-101	13B	87.1	94.6
<hr/>			
<i>Encoder-Only</i>			
AfriBERTa	0.126B	88.9	87.0
AfroXLMR-base	0.270B	<u>92.2</u>	<u>94.7</u>
AfroXLMR-large	0.550B	93.1	97.3

MasakhaNEWS topic
classification

Model / Variant	Size	BLEU	chrF	ROUGE
T2X (isiXhosa)				
MzansiLM				
<i>Base 0-shot</i>	0.125B	0.00	0.03	3.29
<i>Base 1-shot</i>	0.125B	0.00	0.03	4.08
<i>Base 3-shot</i>	0.125B	0.00	0.00	0.74
<i>mono-t2x-ft</i>	0.125B	20.65	31.56	41.19
<i>general-ft</i>	0.125B	0.00	0.00	2.05
<hr/>				
<i>Encoder-Decoder</i>				
mT5-base	0.58B	<u>16.8</u>	<u>28.7</u>	<u>38.7</u>
Aya	13B	8.9	22.1	33.9

T2X data-to-text generation

LLM for South African languages from scratch

- MzansiLM: Train 125M parameter decoder model on data from South African languages only (3.8B tokens)

Model / Variant	Size	Eng	Xho	Zul	Sot	Nso	Afr
MzansiLM							
<i>Base 0-shot</i>	0.125B	32.3	28.9	31.2	18.0	17.3	43.4
<i>mono-sib200-ft</i>	0.125B	28.0	39.1	22.0	36.8	36.6	54.4
<i>multi-sib200-ft</i>	0.125B	33.3	40.4	47.2	34.7	30.5	29.6
<i>general-ft</i>	0.125B	28.0	39.1	47.2	36.8	33.9	54.4
<hr/>							
<i>Decoder-Only</i>							
InkubaLM-0.4B	0.4B	9.0	8.4	8.2	5.3	6.4	5.3
AfroLlama-V1	8B	6.4	6.5	6.4	39.7	38.7	6.4
Llama-3.1-70B-Instruct	70B	88.3	65.0	57.3	54.4	55.8	85.6
<hr/>							
<i>Encoder-Decoder</i>							
Aya-101	13B	82.8	82.0	82.9	81.4	82.1	83.7
<hr/>							
<i>Encoder-Only</i>							
AfriBERTa	0.126B	–	70.7	73.5	55.9	54.8	89.8
AfroXLMR-base	0.270B	–	<u>83.1</u>	<u>84.9</u>	<u>83.7</u>	<u>80.7</u>	<u>90.4</u>
AfroXLMR-large	0.550B	–	84.0	85.8	<u>83.5</u>	83.3	91.1

SIB-200 topic classification

Model / Variant	Size	Xho	Zul	Tsn	Ssw	Sot	Eng	Afr
MzansiLM								
<i>Base 0-shot</i>	0.125B	27.8	28.0	28.3	27.8	27.1	27.3	27.3
<i>Base 1-shot</i>	0.125B	29.4	28.2	30.8	29.7	27.4	27.4	27.6
<i>Base 3-shot</i>	0.125B	28.8	28.2	29.0	28.8	27.3	27.6	27.4
<i>general-ft</i>	0.125B	21.9	25.1	21.9	23.6	22.0	29.6	31.1
<hr/>								
<i>Decoder-Only</i>								
InkubaLM-0.4B	0.4B	23.1	23.2	24.6	–	–	23.9	25.9
AfroLlama-V1	8B	24.8	28.2	26.8	22.9	22.6	25.3	24.7
Llama-3.1-8B-Instruct	8B	35.1	35.3	32.3	32.3	33.7	80.7	66.9
Meta-Llama-3-70B-Instruct	70B	41.3	42.9	<u>41.4</u>	42.9	<u>51.3</u>	93.2	88.9
<hr/>								
<i>Encoder-Decoder</i>								
Aya-101	13B	65.9	64.9	63.6	57.6	61.7	<u>86.1</u>	<u>81.7</u>
<hr/>								
<i>Encoder-Only</i>								
XLM-V large	–	<u>54.4</u>	<u>54.2</u>	–	<u>47.1</u>	32.7	77.8	72.3

Belebele reading comprehension

IrokoBench results

Model	size	AfriXNLI		AfriMMLU	
		in-lang.	translate test	in-lang.	translate test
AfroXLMR-76L	559M	65.7	63.6		
mT0-XXL-MT	13B	51.0	49.9	27.9	28.4
Aya-101	13B	51.5	50.2	29.7	31.1
BLOOMZ 7B	7B	39.4	47.6	24.1	27.9
LLaMa 3 8B	8B	35.4	38.2	28.1	31.8
LLaMa 3.1 8B	8B	36.6	43.6	31.1	41.1
LLaMaX 3 8B	8B	40.8	33.3	29.3	35.2
Gemma 2 9B	9B	40.3	43.3	35.4	44.7
Gemma 2 27B	27B	42.8	49.0	39.9	48.8
LLaMa 3.1 70B	70B	38.0	42.8	39.4	51.3
Command-R	35B	43.4	<u>57.0</u>	27.8	40.8
Claude Opus	UNK	58.1	56.4	43.0	47.6
Gemini-1.5-Pro	UNK	59.4	49.9	60.2	<u>53.1</u>
GPT-3.5-Turbo	UNK	42.1	45.5	38.1	46.8
GPT-4o-mini	UNK	54.2	56.7	45.5	50.2
GPT-4-Turbo	UNK	59.5	<u>57.0</u>	54.2	52.1
GPT-4o	UNK	<u>64.3</u>	52.1	<u>60.0</u>	54.1

AfroBench Leaderboard

Rank	Model	Score	NLU						
			POS	NER	Senti	Topic	Intent	Hate	NLI
1	GPT-4o (Aug)	59.6	62.8	40.7	68.0	75.0	74.0	63.0	64.3
2	Gemini 1.5 pro	58.5	60.8	41.8	68.3	76.8	74.3	61.7	62.0
3	Gemma2 27b	47.9	55.1	50.8	63.4	62.9	34.9	45.7	42.8
4	LLaMa3.1 70B	43.5	54.1	14.4	50.6	58.4	34.0	49.3	38.0
5	Gemma2 9b	43.1	51.9	40.3	60.0	56.4	31.7	30.1	40.3
6	Aya-101 13B	40.3	0.0	0.0	63.4	70.7	44.8	31.6	51.5
7	LLaMAX3 8B	30.1	41.5	0.0	51.9	49.9	5.6	29.2	40.8
8	LLaMa3.1 8B	29.5	47.1	11.5	52.8	47.5	6.0	23.6	36.5
9	Gemma1.1 7b	29.1	38.6	27.9	43.3	45.7	9.4	24.2	34.4
10	LLaMa3 8B	28.8	48.5	22.7	43.6	38.0	2.1	27.8	35.4
11	LLaMa2 7b	22.5	27.9	15.6	42.3	19.7	1.5	21.4	33.8
12	AfroLLaMa 8B	19.8	0.0	3.5	43.4	31.8	0.8	18.1	35.9

AfroBench Leaderboard

Rank	Model	Score	QA		Knowledge		Reasoning	NLG			
			XQA	Arc-E	RC	MMLU	MATH	MT (en/fr-xx)	MT (xx-en/fr)	SUMM	ADR
1	GPT-4o (Aug)	59.6	43.4	85.7	69.2	60.4	49.8	35.5	41.0	66.5	54.9
2	Gemini 1.5 pro	58.5	40.5	84.8	52.7	57.6	52.3	37.9	42.0	66.7	55.6
3	Gemma2 27b	47.9	50.5	56.3	53.9	40.5	27.0	28.3	33.2	66.4	55.1
4	LLaMa3.1 70B	43.5	44.0	57.5	49.7	39.9	23.2	25.6	38.3	67.6	51.7
5	Gemma2 9b	43.1	45.9	53.4	51.6	37.1	18.7	25.1	29.4	66.1	51.6
6	Aya-101 13B	40.3	62.5	60.0	60.7	30.9	4.4	23.9	38.2	52.4	50.4
7	LLaMAX3 8B	30.1	2.2	39.9	29.7	28.3	4.7	23.2	35.3	50.7	49.4
8	LLaMa3.1 8B	29.5	21.8	32.8	39.5	31.4	6.8	16.7	28.9	43.7	25.9
9	Gemma1.1 7b	29.1	17.4	32.2	38.1	28.6	4.6	11.6	9.6	49.1	50.8
10	LLaMa3 8B	28.8	12.6	32.0	27.6	27.4	5.1	16.4	28.1	66.2	27.8
11	LLaMa2 7b	22.5	13.7	23.3	24.3	25.6	2.0	10.8	20.7	46.9	30.4
12	AfroLLaMa 8B	19.8	21.8	37.2	24.1	25.8	0.3	8.5	9.5	50.8	5.2

AfroBench

- Fine-tuned models still sometimes do better than prompt-based (zero-shot)

Tasks Metrics	natural language understanding						
	POS acc	NER F1	SA F1	TC acc	Intent acc	Hate F1	NLI acc
<i>Fine-tuned baselines</i>							
AfroXLMR	89.4	84.6	72.1	74.4	93.7	77.2	61.4
mT5/AfriTeVa V2 1B							
NLLB 3.3B							
<i>Prompt-based baselines</i>							
<i>open models</i>							
Gemma 1.1 7B	38.6	27.9	43.3	45.3	9.4	24.3	34.0
LLaMa 2 7B	27.9	15.6	42.3	19.4	1.5	21.9	33.8
LLaMa 3 8B	48.5	22.7	43.6	37.0	2.1	27.8	35.4
LLaMaX 8B	41.6	0.0	51.9	49.8	5.6	28.6	40.8
LLaMa 3.1 8B	47.1	11.5	50.5	46.7	6.0	23.6	36.6
AfroLLaMa 8B	0.0	3.5	43.4	19.8	0.8	18.4	35.9
Gemma 2 9B	51.9	40.3	60.0	56.0	29.2	29.9	40.3
Aya-101 13B	0.0	0.0	<u>63.4</u>	70.3	42.4	31.0	<u>51.5</u>
Gemma 2 27B	<u>55.1</u>	<u>50.8</u>	<u>63.4</u>	62.4	33.0	45.5	42.8
LlaMa 3.1 70B	54.1	14.4	52.2	57.7	34.0	<u>49.0</u>	38.0
<i>proprietary models</i>							
Gemini 1.5 pro	60.8	41.8	68.3	76.7	74.3	62.1	62.0
GPT-4o (Aug)	<u>62.8</u>	40.7	68.0	74.8	74.0	63.5	64.3

Conclusions

- Data scarcity remains a limitation for training or adapting LLMs for African languages
- Decoder-only models still often under-perform
- Task-specific fine-tuning can work if sufficient task data is available

Dr Jan Buys. Email: jbuys@cs.uct.ac.za

- Research papers, models and code are available



For more details visit www.janmbuys.com