Introduction to Language Adaptation of Large Language Models

24/11/2026 - COM3110/4115: Text Processing

Atsuki Yamaguchi

University of Sheffield

A World TOP 100 University

Self-introduction



Atsuki Yamaguchi

Sheffield NLP Group

Research Fellow & final-year PhD

Research Interests

Cross-lingual transfer - how to specialise an LLM to a specific language(s)
Efficient domain adaptation - how to efficiently specialise an LLM to a specific domain(s)
Efficient language modelling - how to efficiently create an LLM

- NLP Researcher @ Hitachi, Ltd. (2021-2023)
- MSc Computer Science with Speech and Language Processing @ Sheffield (2019-2020)

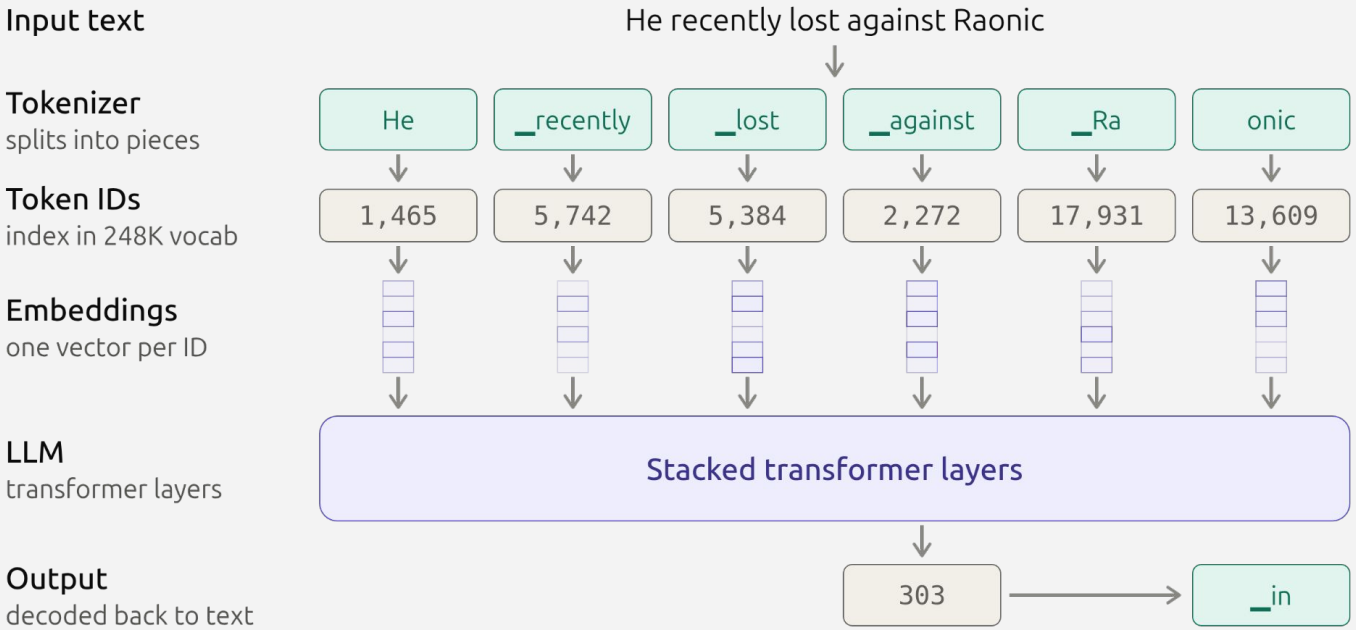
Lecture Structure

1. Background
2. What is Language Adaptation?
3. Tokeniser Adaptation
4. Continual Training
5. Discussion & Future Directions

University of Sheffield

1. Background

Recap: How LLMs represent words



- A tokeniser splits text into tokens: words, or pieces of words
- It has a fixed vocabulary (like a dictionary), learned from its training text
- Words not in the vocabulary are broken into smaller pieces, down to single bytes

Let’s tokenise parallel text!

- The same sentence in Amharic needs 4.3× more tokens.
- This is called: *overfragmentation*
- **What does this mean?**





Let's tokenise parallel text!

- The same sentence in Amharic needs 4.3× more tokens.
- This is called: *overfragmentation*
- **What does this mean?** Higher inference costs:
  - **Slower**: the model generates one token at a time
  - **More expensive**: APIs charge per token
  - **Less room**: the context window fills up faster



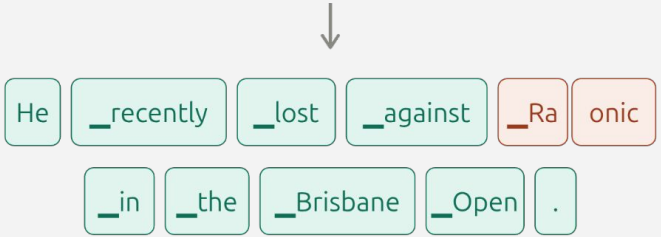


Do All Languages Cost the Same? Tokenization in the Era of Commercial Language Models (Ahia et al., EMNLP 2023)
Language Model Tokenizers Introduce Unfairness Between Languages (Petrov et al., NeurIPS 2023)

Overfragmentation happens across languages

English

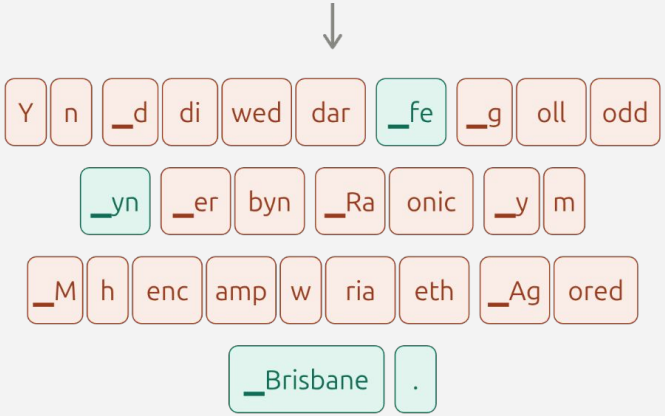
He recently lost against Raonic in the Brisbane Open.



11 tokens

Welsh

Yn ddiweddar fe gollodd yn erbyn Raonic ym Mhencampwriaeth Agored Brisbane.



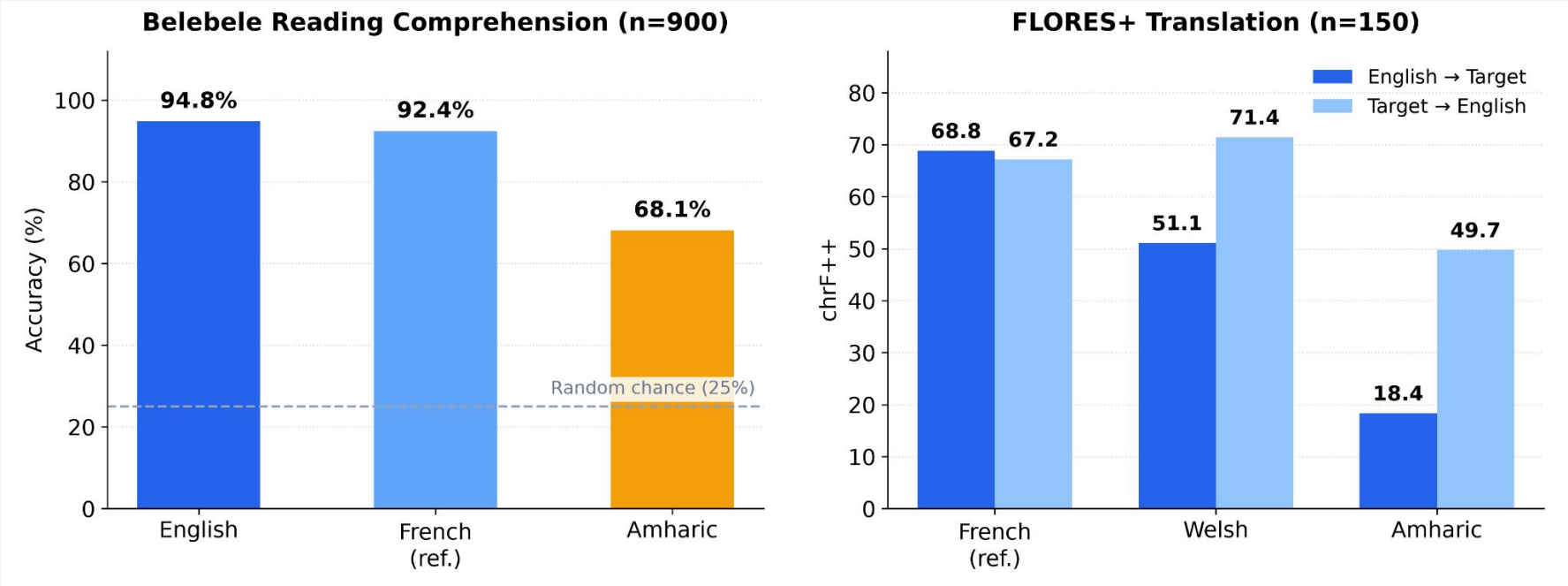
28 tokens (2.5× the cost)

Whole word = 1 token

Word split into pieces

How about task performances across languages?

Qwen3.5 9B performance across languages (Zero-shot, Non-thinking)



Performances on “low-resource” languages tend to be lower compared to “high-resource” languages!

Why do these gaps occur?

- LLMs learn from web text, often from Common Crawl (free web snapshots)

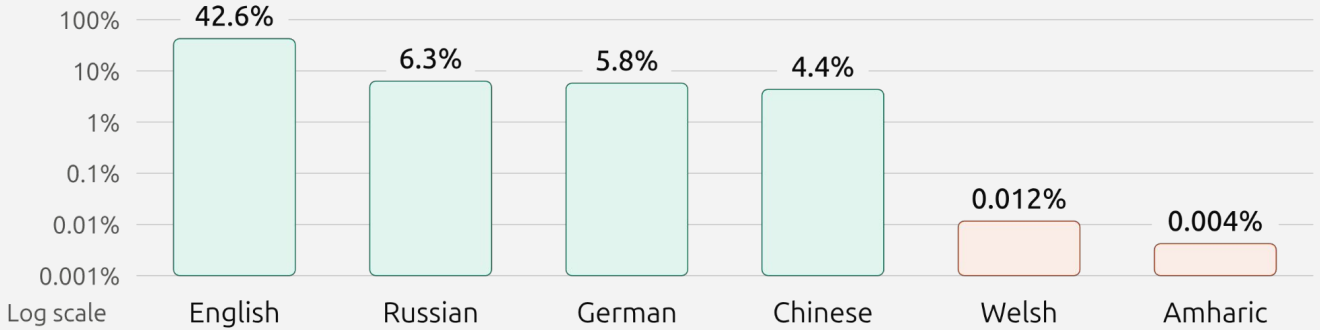


Why do these gaps occur?

- LLMs learn from web text, often from Common Crawl (free web snapshots)



- And the web is mostly English:



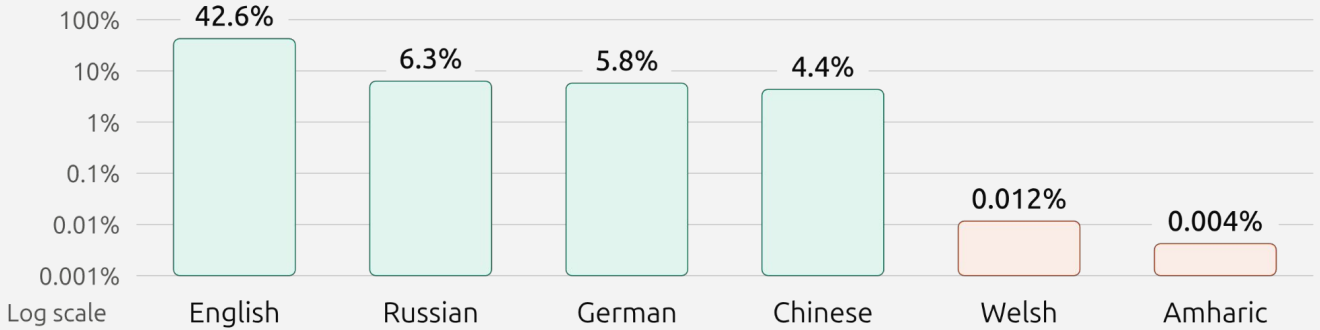
Source: https://github.com/commoncrawl/cc-crawl-statistics/blob/master/plots/languages.csv

Why do these gaps occur?

- LLMs learn from web text, often from Common Crawl (free web snapshots)



- And the web is mostly English:



Tokenisers and LLMs learn what is frequent in their data
→ Imbalanced web data = an inherited bias towards English
→ Overfragmentation + lower performance for other languages

Source: https://github.com/commoncrawl/cc-crawl-statistics/blob/master/plots/languages.csv

University of Sheffield

2. What is Language Adaptation?

How can we fix these problems?

Problem: Overfragmentation + lower performance

Option 1 Train a new LLM from scratch on balanced data

Pros: Straightforward
Cons:

- Too costly
- Balanced data for many languages does not exist!

Option 2 Adapt an existing LLM to a target language(s)
:= **Language adaptation**

Pros:

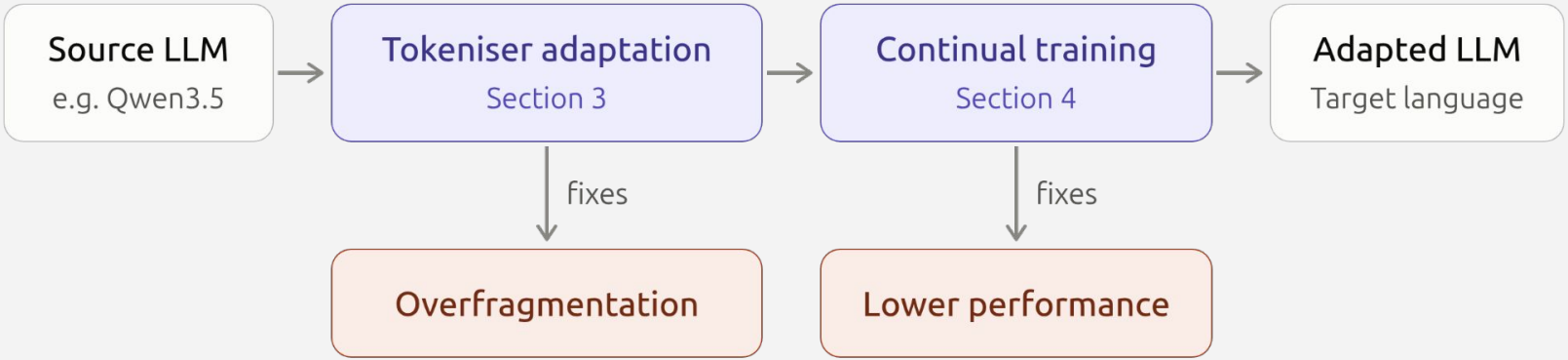
- Less data required:
  Reuses facts & reasoning learned from English + other languages
  → Only needs to learn new words & grammar

Cons:

- Side-effects like forgetting

The adaptation pipeline

How can we do language adaptation?



- **Tokeniser adaptation**: Change the vocabulary to suit the target language
  → fewer tokens for the same text
- **Continual training**: Training the model on target-language text
  → better understanding and generation of the language

University of Sheffield

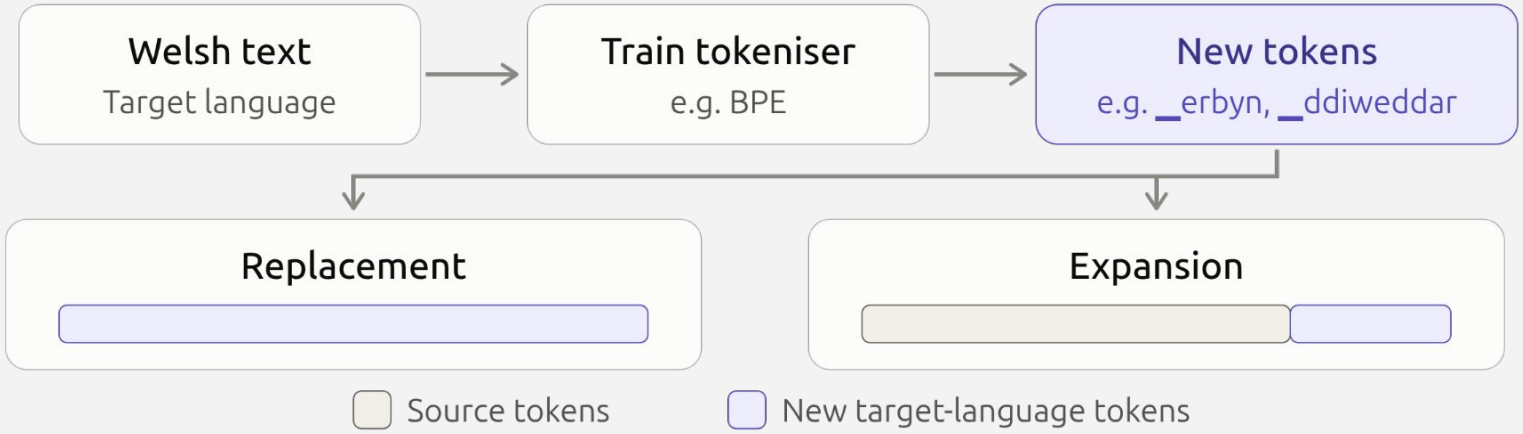
3. Tokeniser Adaptation

If you want to learn more on tokenisation, check out “Tokenization: A Survey for Modern NLP” and “Tokenization in Large Language Models” (EMNLP 2026 Tutorial)

Vocabulary expansion & replacement

Train a tokeniser on target-language text, then either:

1. **Replacement**: replace the model's vocabulary with the new one
   (e.g., WECHSEL, FOCUS)
2. **Expansion**: add the new tokens to the model's existing vocabulary
   (e.g., Chinese-LLaMA, Swallow)



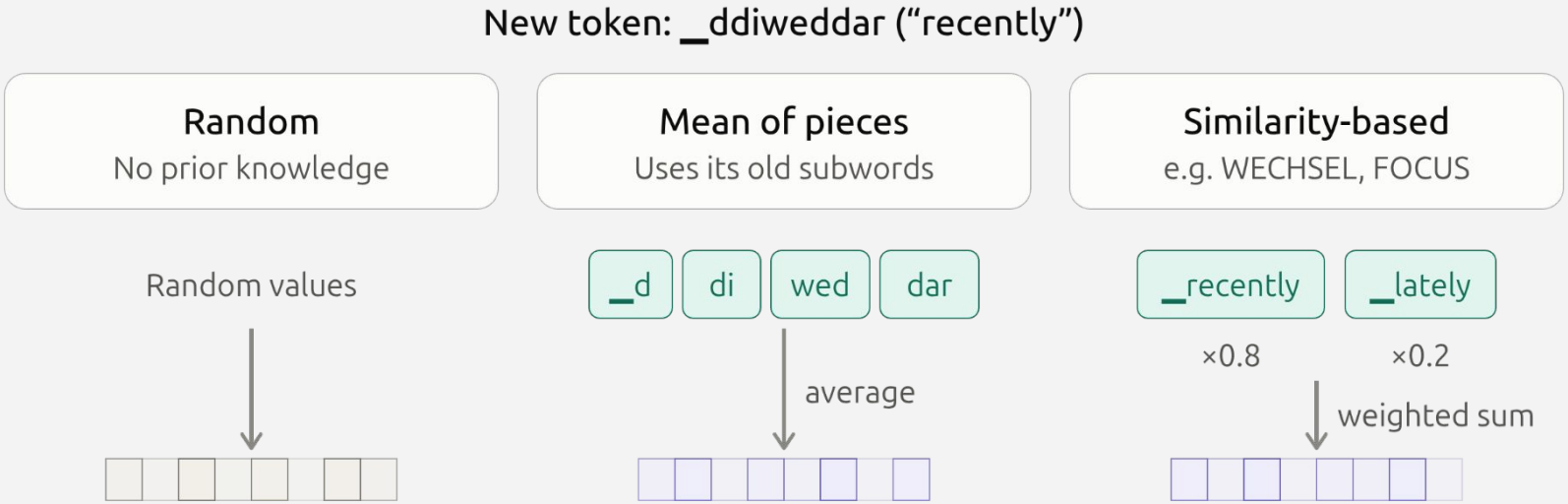
Replacement: WECHSEL (Minixhofer et al., NAACL 2022); FOCUS (Dobler & de Melo, EMNLP 2023) etc.
Expansion: Chinese-LLaMA (Cui et al., arXiv 2023); Swallow (Fujii et al., COLM 2024) etc.

New tokens need new embeddings

- The model has never seen the new tokens → their embeddings must be initialised
- **Replacement**: every embedding is new
  Expansion: only the added tokens → expansion is now common for LLMs
- **How do we initialise them?**

New tokens need new embeddings

- The model has never seen the new tokens → their embeddings must be initialised
- **Replacement**: every embedding is new
  Expansion: only the added tokens → expansion is now common for LLMs
- **How do we initialise them?** → Many studies have looked at this!



Do they work?

Gains

Fewer tokens → faster, cheaper, more room in the context window

Needs

good initialisation + some target-language training to keep performance



Before: 4 tokens

After: 1 token

However, every new language(s) needs a new tokeniser *and* new embeddings to train
→ *Can we avoid this?*

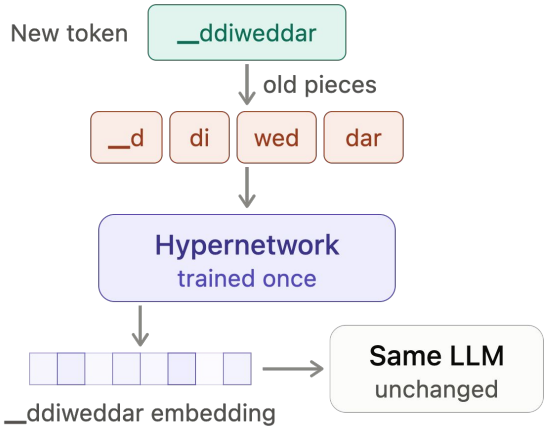
An Empirical Study on Cross-lingual Vocabulary Adaptation for Efficient Language Model Inference (Yamaguchi et al., EMNLP Findings 2024)
How Can We Effectively Expand the Vocabulary of LLMs with 0.01GB of Target Language Text? (Yamaguchi et al., Computational Linguistics 2026)

Making tokeniser transfer cheaper

Zero-shot tokeniser transfer (ZeTT)

Train a hypernetwork once to predict embeddings for *any* tokeniser

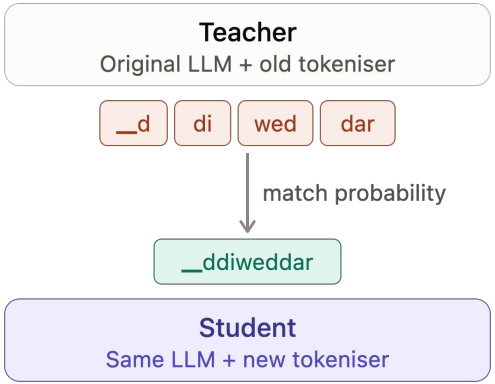
Training: Hypernetwork is trained on many randomly sampled tokenisers



Self-distillation (ALM)

The original model teaches itself to use the new tokeniser

Training: the student learns to match the teacher's predictions on the same text



Zero-Shot Tokenizer Transfer (Minixhofer et al., NeurIPS 2024)
Universal Cross-Tokenizer Distillation via Approximate Likelihood Matching (Minixhofer et al., NeurIPS 2025)

Going below tokens: byte-level models

Can we get rid of the tokeniser altogether?

Yes, read text as raw bytes, with a vocabulary of just 256 symbols (1 byte = 8 bits → 2^8 = 256 possible values; all text is stored as bytes)

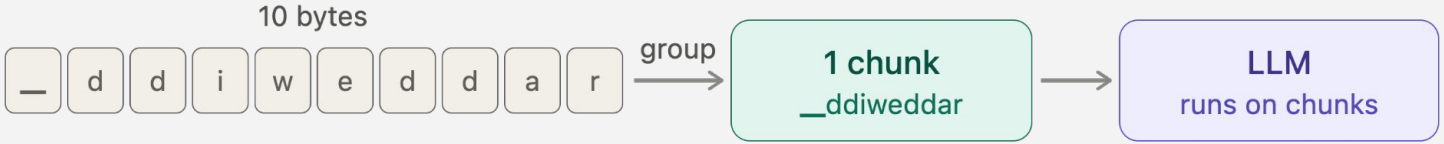
→ no tokeniser necessary



But: sequences get much longer



Fix: group bytes into chunks, then run the LLM on the chunks



Grouping requires training, but existing LLMs can even be converted to bytes, including for low-resource languages!

Dynamic Chunking for End-to-End Hierarchical Sequence Modeling (Hwang et al., ICLR 2026)
Retrofitting language models to operate over bytes (Minixhofer et al., Nature 2026)
When Tokenizers Fail: Byte-Level Chunking for Zero-Shot Transfer to Low-Resource Languages (Kumar et al., EMNLP 2026)

Comparing approaches

There is no single winner:

A trade-off between training cost and keeping what the model already knows

	Replacement	Expansion	ZeTT / Self-distillation	Byte-level
What changes	Whole vocabulary swapped	New tokens added	Tokeniser swapped cheaply	No tokeniser at all
Extra training	A lot: every embedding is new	Moderate: only new embeddings	Little per new tokeniser	Model must be converted to bytes
Limitation	Loses the original tokens	Vocabulary grows	Needs a hypernetwork or a distillation run	Longer sequences; newest and least mature

These mainly fix overfragmentation. What about lower performance?
→ Continual training

University of Sheffield

4. Continual Training

What is continual training?

Parameters: The numbers that a model learns during training (They are saved in “embeddings” and “layers”, see slide 5)

Continual training: Keep training an existing model on new data (here, target-language text)
→ like a student who keeps studying a new subject

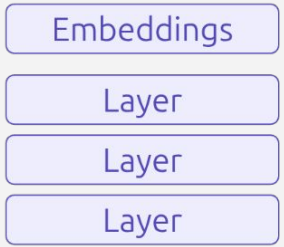
Problem: Updating every parameter is expensive, and new training can overwrite old knowledge (= forgetting old knowledge) as they share the same parameters

→ **Can we change fewer parameters while improving target performance?**

Which parameters do we update?

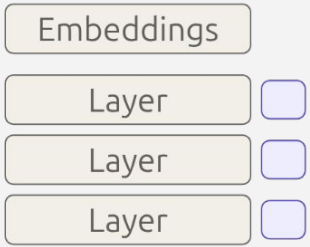
Old and new knowledge share the same parameters
→ What if we change fewer of them, or add new ones?

Update everything



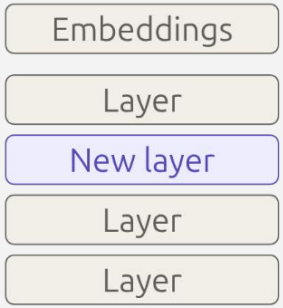
Rewrite the textbook
e.g. full training

Update a little



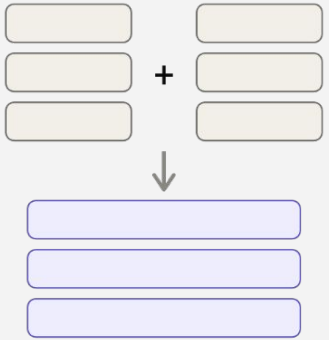
Add sticky notes
e.g. LoRA

Add new parts



Add a new chapter
e.g. expansion, MoE

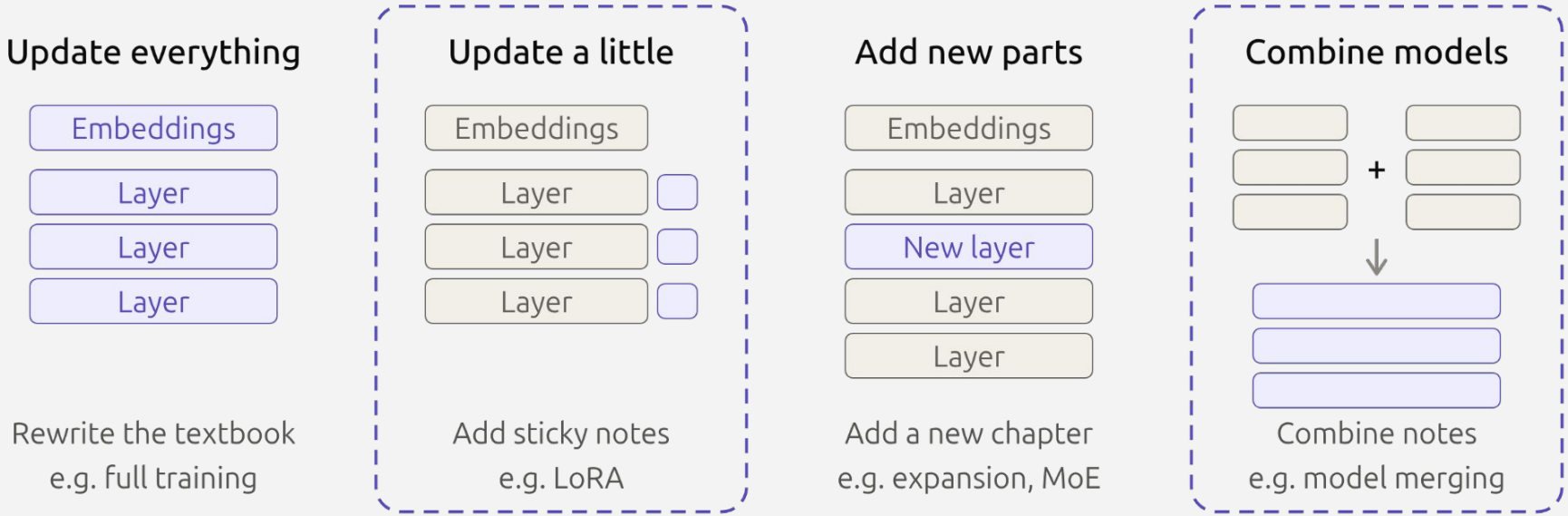
Combine models



Combine notes
e.g. model merging

Which parameters do we update?

Old and new knowledge share the same parameters
→ What if we change fewer of them, or add new ones?



Add new parts:
LLaMA Pro: Progressive LLaMA with Block Expansion (Wu et al., ACL 2024)
MoE-LPR: Multilingual Extension of Large Language Models Through Mixture-of-Experts with Language Priors Routing (Zhou et al., AAAI 2025)

Update a little: LoRA

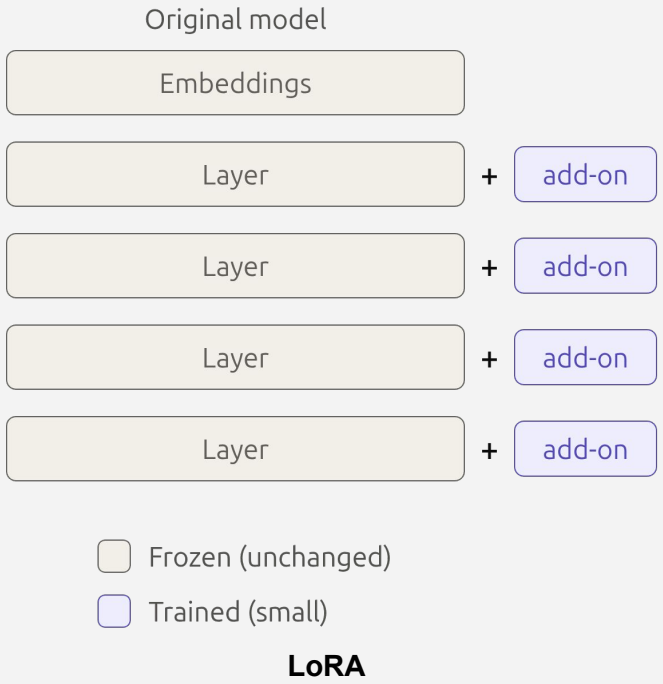
Idea: Freeze the original model and train a small add-on for the new language(s)

LoRA benefits:

- Far fewer parameters to train
  → much cheaper
- The original model stays unchanged
  → less forgetting

Other ways to "update a little":

- Train a random half of the parameters
- Train only the first and last few layers
- Train only the parameters least important to what the model already knows, etc.



LoRA: Low-Rank Adaptation of Large Language Models (Hu et al., ICLR 2022)
HFT: Half Fine-Tuning for Large Language Models (Hui et al., ACL 2025)
Trans-Tokenization and Cross-lingual Vocabulary Transfers: Language Adaptation of LLMs for Low-Resource NLP (Remy et al., COLM 2024)
Mitigating Catastrophic Forgetting in Target Language Adaptation of LLMs via Source-Shielded Updates (Yamaguchi et al., ACL 2026)

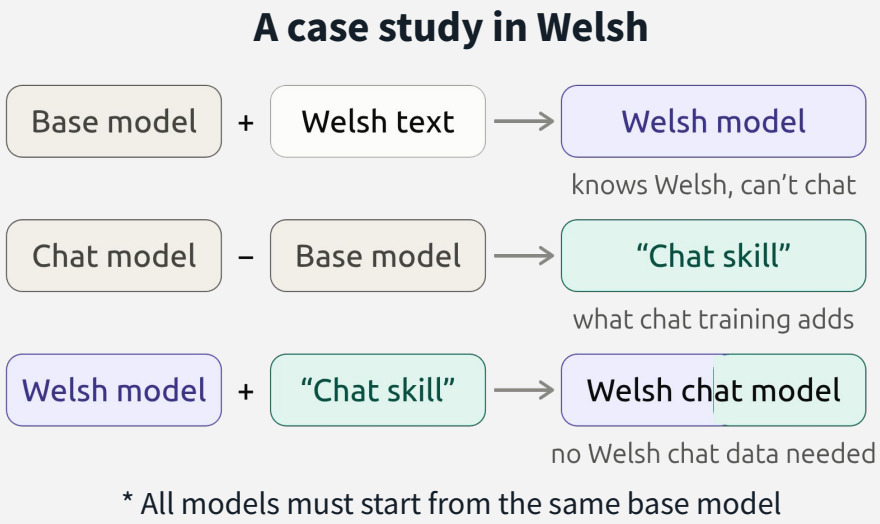
Combine models: model merging

Problem: Continual training on target-language text gives a model that knows a target language, but it can't chat

Idea: Take the "chat skill" from an existing chat model and add it
→ Chat skill = chat model − base model

Benefits:

- No Welsh chat data needed
- No extra training. Just arithmetic on the parameters



Editing models with task arithmetic (Ilharco et al., ICLR 2023)
Chat Vector: A Simple Approach to Equip LLMs with Instruction Following and Model Alignment in New Languages (Huang et al., ACL 2024)
Adapting Chat Language Models Using Only Target Unlabeled Language Data (Yamaguchi et al., TMLR 2025)

University of Sheffield

5. Discussion & Future Directions

Data scarcity & evaluation

What does "low-resource" even mean? Data, speakers, digital support, etc.

Where does data come from?

Problem: little text exists, and much of it is noisy or machine-translated

Workarounds:

- Translate or generate data with LLMs
  → may be unnatural or wrong
- Guide an LLM with dictionaries and grammar books
  → promising, still debated
- Borrow from a related language (e.g. Hindi for Bhojpuri)

Open question: how to adapt with very little natural text?

How do we know language adaptation worked?

Problem: few test sets exist in the target language

- Many benchmarks miss the language
- Translated tests are Western-centric

Open question: how to test what speakers actually need?

The Zeno's Paradox of 'Low-Resource' Languages (Nigatu et al., EMNLP 2024)
Quality at a Glance: An Audit of Web-Crawled Multilingual Datasets (Kreutzer et al., TACL 2022)
A Shocking Amount of the Web is Machine Translated: Insights from Multi-Way Parallelism (Thompson et al., ACL Findings 2024)
Text Generation in Low-Resource Languages Using Human-Curated Vocabularies (Sinanta et al., arXiv 2026)

Safety & cultural adaptation

Safety

Problem: safety training (e.g. refusing harmful requests) is done mostly in English

- Translating a harmful request into a low-resource language can bypass safeguards
- Adaptation can make it worse: training on new data can overwrite safety too

Culture

Problem: generating a fluent text in a language ≠ understanding its culture

- Models default to the most common (often US) norms (e.g. Mother's Day in May, not the UK's March)
- Everyday knowledge of underrepresented cultures is weak

Safety & cultural adaptation

English, no country

What day of the year is usually dedicated to fireworks?

Assumes US

"…the most widely recognized day associated with fireworks is July 4th in the United States."

→ UK: 5 November

English, in the UK

In the UK, what day of the year is usually dedicated to fireworks?

Wrong main answer

"…the day most famously dedicated to fireworks is New Year's Eve, December 31st."

→ Bonfire Night: 5 Nov

English, in Ethiopia

In Ethiopia, what day of the year is usually dedicated to fireworks?

Wrong date

"…the day most famously associated with fireworks is January 1st, New Year's Day (known locally as Enkutatash)."

→ Actually 11 September

Grey: question · Coloured: Qwen3.5-9B answer

Takeaways

1 Imbalanced web data → LLMs inherit a bias towards English
→ Overfragmentation + lower performance for other languages

2 Tokeniser adaptation reduces overfragmentation
→ Fewer tokens: faster, cheaper, more room in the context window

3 Continual training improves performance
→ Update a little, add new parts, or combine models to limit cost and forgetting

Still open: data, evaluation, safety, culture



University of Sheffield

Questions & inquiries:

📧 ayamaguchi1@sheffield.ac.uk

Code (slides 9 & 33) is available at

https://github.com/gucci-j/intro-lang-adaptation