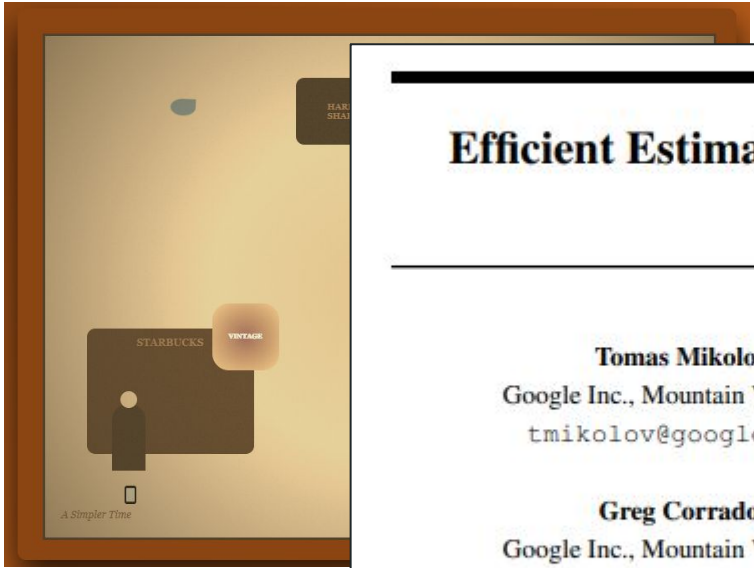
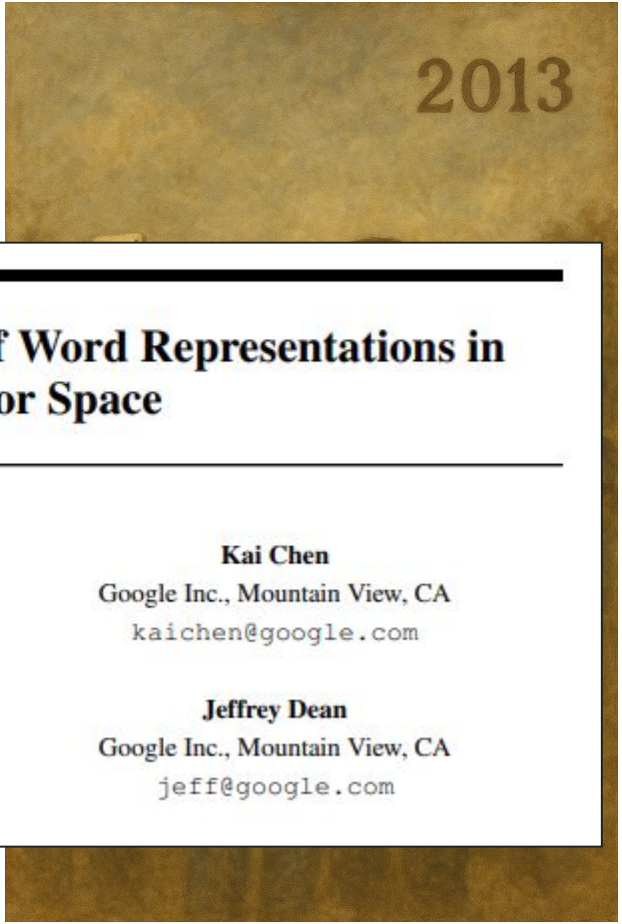
Beat them? Join them? Fix them? Tokenization Research in a Downstream World

Yuval Pinter
Dept. of CS, Ben-Gurion University of the Negev

Tokenization Workshop @ ICML
July 18, 2025

A Tale of NLP Science





Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Everybody knows whom to credit

The second system in each ensemble was a system based on word embeddings (Mikolov et al., 2013).

very large sets of labeled training data. To alleviate this problem, we use pre-trained continuous word embeddings (Mikolov et al., 2013) as input embeddings rather than the one-hot word encodings.

alent questions. The proposed CNN first transforms words into word embeddings (Mikolov et al., 2013), using a large collection of unlabeled data, and then applies a convolutional network to build distributed vector representations for pairs of questions. Finally, it scores the questions using

(Kanerva et al., 2000), and (3) word embeddings from neural language models, such as skip-gram word embeddings (Mikolov et al., 2013).

Given a list of seed words, resources such as WordNet, word embeddings (Mikolov et al., 2013) and paraphrase databases (e.g., PPDB (Ganitkevitch et al., 2013)) can be utilized to find semantically similar words and phrases.

of texts. At present, Neural Network is one of the most used learning techniques for generating word embeddings (Mikolov and Dean, 2013). The essential assumption of this mo-

& Bengio, 2014; Luong et al., 2015), efficient distributed vectorspace word embeddings (Mikolov et al., 2013),

We propose the following compromise: the input and output parameter matrices $\boldsymbol{W}_{in}$ and $\boldsymbol{W}_{out}$ can be thought of as real-valued embeddings, akin to latent factors in matrix factorization models (Koren et al., 2009) or word embeddings (Mikolov et al., 2013). With

But then

GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning
Computer Science Department, Stanford University, Stanford, CA 94305

suggested in Kiros et al. (2015). We use GloVe vectors (Pennington et al., 2014) as pre-trained word embeddings. The MC-QT model outperforms all previous methods, including the variation of Gan et al. (2016) which uses pre-trained word embeddings.

End-to-End Neural Module Networks We use the publicly available implementation.[16] The model parameters used for NLVR2 are the same as those used for the original experiments on VQA. We use GloVe vectors of size 300 to embed words (Pennington et al., 2014). The model

The Word2Vec word embedding made available by Google [1] is trained on Google News dataset. Since our datasets consist of tweets, we use GloVe vectors specifically pre-trained on twitter [2]in this work.

In this thesis, we use GloVe vectors for word embeddings.

Here we take the same CNN architecture as before but we apply Zipf's Law to preprocess the text before we use GloVe vectors for feature extraction. We do this in hope of only extracting

We collected 238,097 historical diplomatic documents that span from 1860 to 1983. We performed stratified sampling to overcome the significant variance in frequency of documents over the year buckets. We also capped the number of documents at 5000 to smooth out the disproportionate representation of certain years. For both models, we use GloVe vectors with 400,000 vocab size of 100 dimensions.

This was easy to do!

- The actual vectors work better
- Out-of-the-box usage is *exactly the same*
  - Just throw away the w2v file and open up the GloVe one!
- No need to know the ins and outs of the algorithm
  - I still teach static embeddings with the word2vec algorithm

```
brillant -0.10595 -0.91585 -0.34122 -0.46488
-0.30766 0.4224 0.54478 0.050233 -0.074818
0.56256 0.46647 0.3628 -0.35072 0.20444 0.090324
-1.2707 -0.48098 0.39637 0.14639 -0.32067 -0.35
-0.49863 -0.17226 -0.17667 0.073446 0.90555
-0.1591 0.20531 0.17151 0.34148 -1.2202 0.0612
0.050296 0.38418 -0.32608 0.55179 0.17156
0.26034 0.022689 -0.017267 0.41452 -0.19447
-0.73922 -0.13019 0.084774 -1.0532 0.39768
0.23137 0.82867 0.32384
rhinovirus 0.42564 -0.010486 -0.10664 -0.31197
0.9388 0.41684 1.4638 -0.33614 0.6207 1.1932
0.069378 0.53001 0.64262 0.678 -0.4701 0.20847
-0.4352 -0.084987 -0.57549 1 -1.0292 -0.091214
1.0561 -0.72 0.89322 0.27471 0.24448 0.25755
0.4183 -0.10341 -1.6764 1.1065 -0.56222 0.042362
-0.62762 -0.23537 0.13483 -0.69969 -0.52485
0.29966 -0.48973 -0.70865 -0.86045 1.0692
0.37511 0.67175 0.099384 0.63725 -0.09825
-0.82316
marciniak -0.84229 0.22514 0.39433 -0.19872
-0.083689 0.24835 -0.12571 0.4825 0.90827
-0.58335 0.30101 -0.11702 0.020311 0.28252
-0.21729 -0.59863 -0.69338 -0.7032 0.64811
-0.55788 -0.63492 0.27522 -0.079907 0.0079145
0.49062 0.39096 0.84874 0.45208 -0.13805
-0.32136 -1.4945 0.15928 0.46679 -0.072639
0.06353 -0.2029 -0.44887 0.79926 -0.13688
-0.30252 0.34524 0.2689 0.8492 -0.69336 0.19409
-0.85535 0.88239 -0.30634 0.33366 0.55691
```

What finally replaced word2vec?

- ELMo, and then BERT
- They introduced **a new paradigm** of contextual representation
- With performance that was impossible to ignore
    - (And indeed, some predecessors were not ignored but are definitely much less well-known)

context2vec:
Learning Generic Context Embedding with Bidirectional LSTM

Oren Melamud
Computer Science Dept.
Bar-Ilan University
melamuo@cs.biu.ac.il

Jacob Goldberger
Faculty of Engineering
Bar-Ilan University
goldbej@eng.biu.ac.il

Ido Dagan
Computer Science Dept.
Bar-Ilan University
dagan@cs.biu.ac.il

Beat them

Creator: Virtisus | Credit: Getty Images

Join them

Fix them

But First, What is it that **They** Want?

- They want to **start with embeddings**
  - “Don’t make me think about text”

- They want A TABLE and SOME CODE that turns TEXT into EMBEDDINGS
  - “Don’t make me change the workflow”
  - **.tokenize()**

- They want to **sample from a softmax or whatever** on the way out
  - “text about think me make n’t Do”

For some reason,
They also want this?

Tokenization :(

Tokenization is at the heart of much weirdness of LLMs. Do not brush it off.

- Why can't LLM spell words? **Tokenization**.
- Why can't LLM do super simple string processing tasks like reversing a string? **Tokenization**.
- Why is LLM worse at non-English languages (e.g. Japanese)? **Tokenization**.
- Why is LLM bad at simple arithmetic? **Tokenization**.
- Why did GPT-2 have more than necessary trouble coding in Python? **Tokenization**.
- Why did my LLM abruptly halt when it sees the string "<|endoftext|>"? **Tokenization**.
- What is this weird warning I get about a "trailing whitespace"? **Tokenization**.
- Why the LLM break if I ask it about "SolidGoldMagikarp"? **Tokenization**.
- Why should I prefer to use YAML over JSON with LLMs? **Tokenization**.
- Why is LLM not actually end-to-end language modeling? **Tokenization**.
- What is the real root of suffering? **Tokenization**.

Good tokenization web app: https://tiktokenizer.vercel.app

Example string:

```
Tokenization is at the heart of much weirdness of LLMs. Do not brush it off.

127 + 677 = 804
1275 + 6773 = 8041

Egg.
```

Tokenization :(

Tokenization is at the heart of much weirdness of LLMs. Do not brush it off.

- Why can't LLM spell words? **Tokenization.**
- Why can't LLM do super simple string processing tasks like reversing a string? **Tokenization.**
- Why is LLM worse at non-English languages (e.g. Japanese)? **Tokenization.**
- Why is LLM bad at simple arithmetic? **Tokenization.**
- Why did GPT-2 have more than necessary trouble coding in Python? **Tokenization.**
- Why did my LLM abruptly halt when it sees the string "<|endoftext|>"? **Tokenization.**
- What is this weird warning I get about a "trailing whitespace"? **Tokenization.**
- Why the LLM break if I ask it about "SolidGoldMagikarp"? **Tokenization.**
- Why should I prefer to use YAML over JSON with LLMs? **Tokenization.**
- Why is LLM not actually end-to-end language modeling? **Tokenization.**
- What is the real root of suffering? **Tokenization.**

ok but who cares
(write a script)

not only (and not unfixable)

pre-tokenization, actually

ok but who cares
(insane edge case)

vanilla BPE, actually

Riddle me this

How many r's are there not not in "strawberry"?

Ah, we're going deeper now. Let's break it down carefully:

Question:

"How many r's are there not not in 'strawberry'?"

That double **"not not"** cancels out — it's equivalent to just saying:

> **"How many r's are there in 'strawberry'?"**

Answer:

In the word **"strawberry"**, the letters are:

s, t, r, a, w, b, e, r, r, y

Count the **r's**:

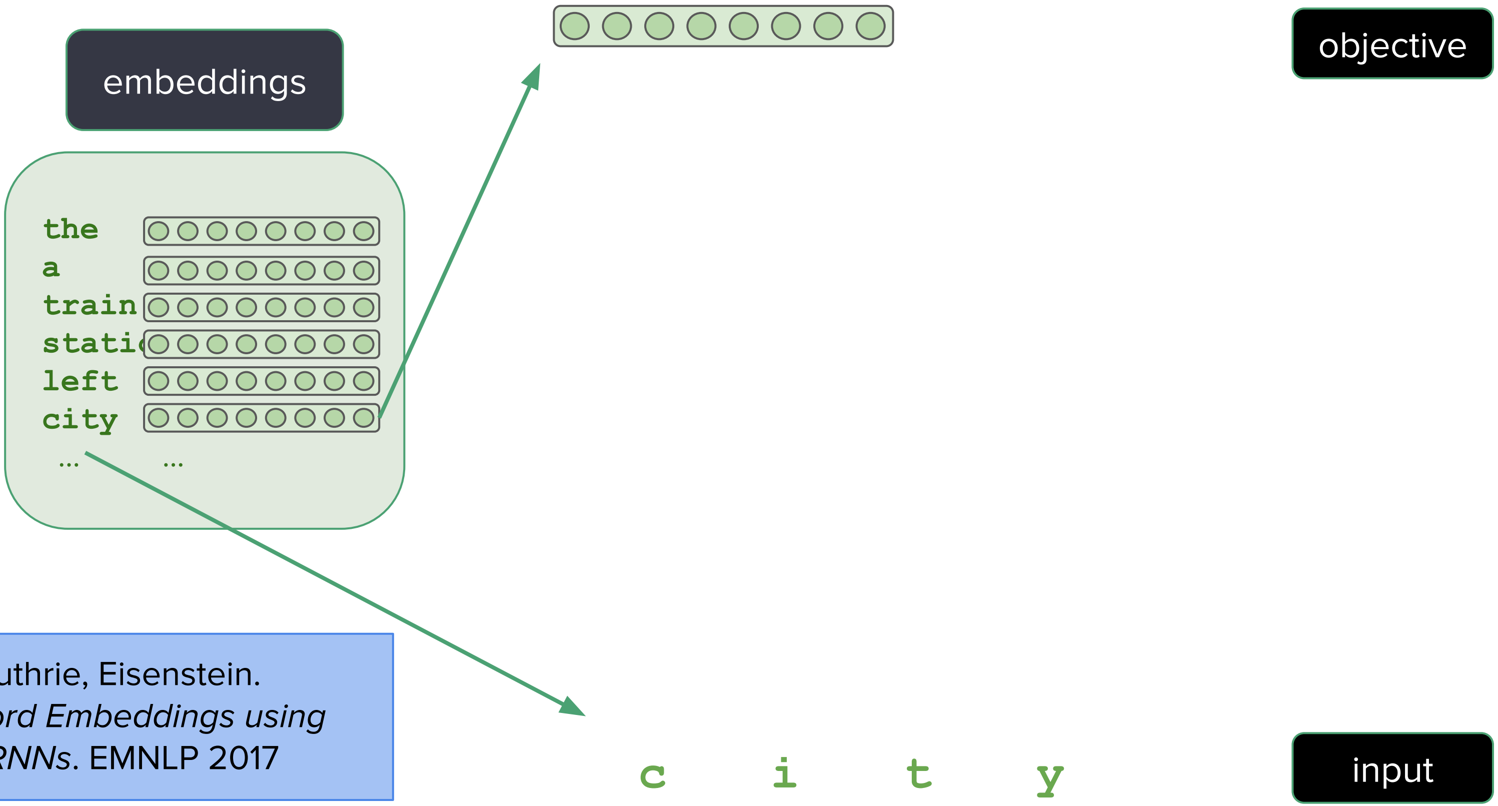
- There are **3 r's**.

Final Answer:

👉 **3** r's are there not not in *strawberry*.

Act I: Beat Them

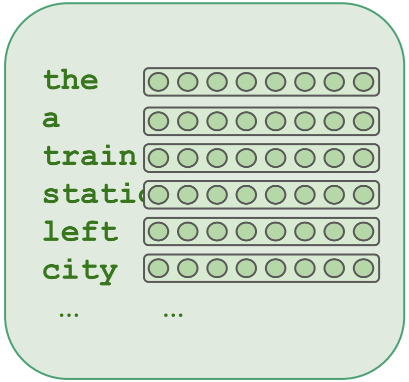
MIMICK, or How I Once Thought We Beat OOVs



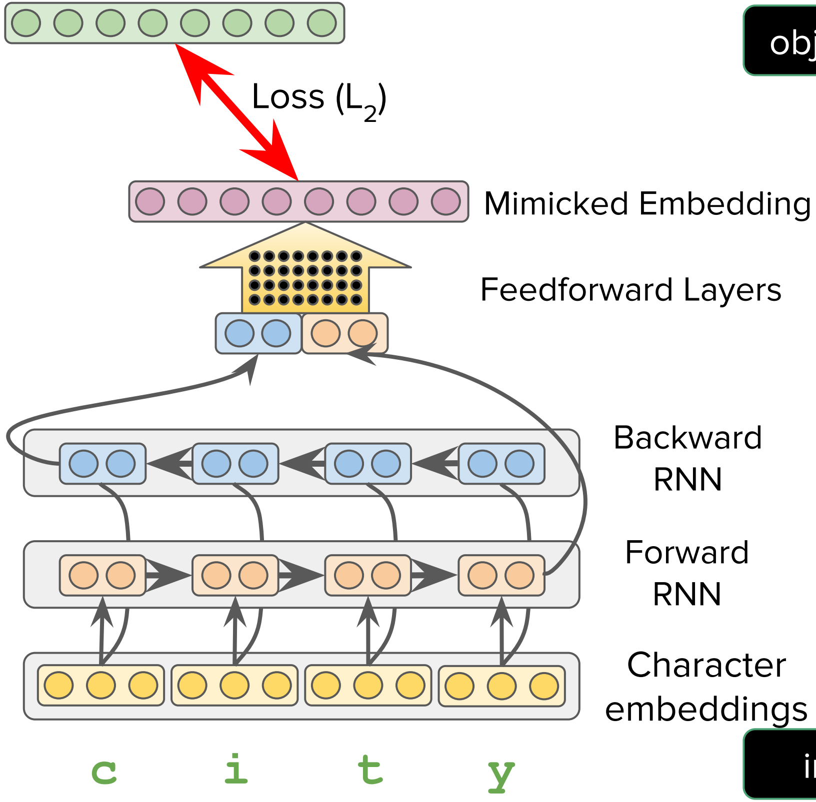
Pinter, Guthrie, Eisenstein.
Mimicking Word Embeddings using Subword RNNs. EMNLP 2017

MIMICK

embeddings



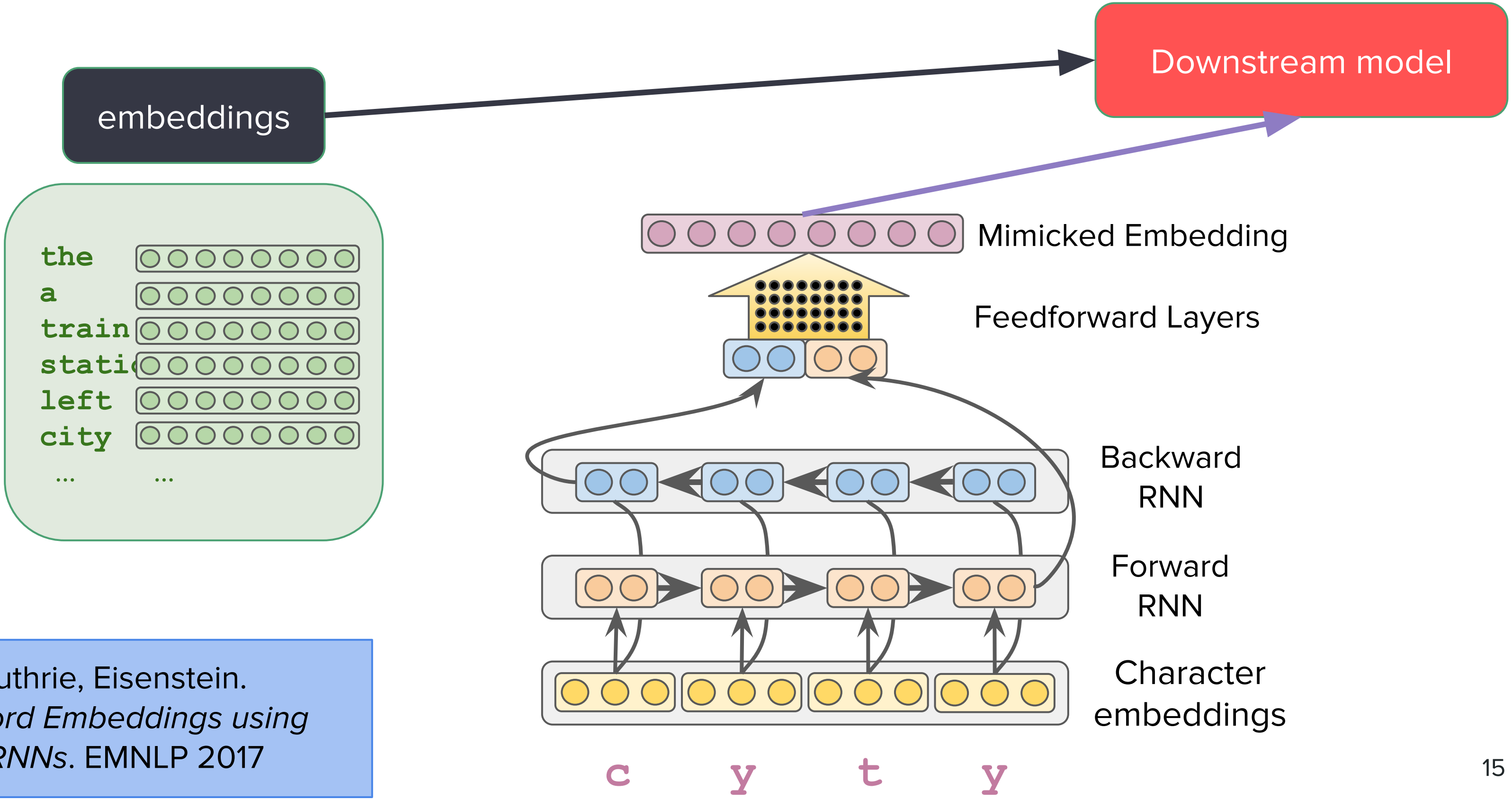
objective



input

Pinter, Guthrie, Eisenstein. *Mimicking Word Embeddings using Subword RNNs*. EMNLP 2017

MIMICK

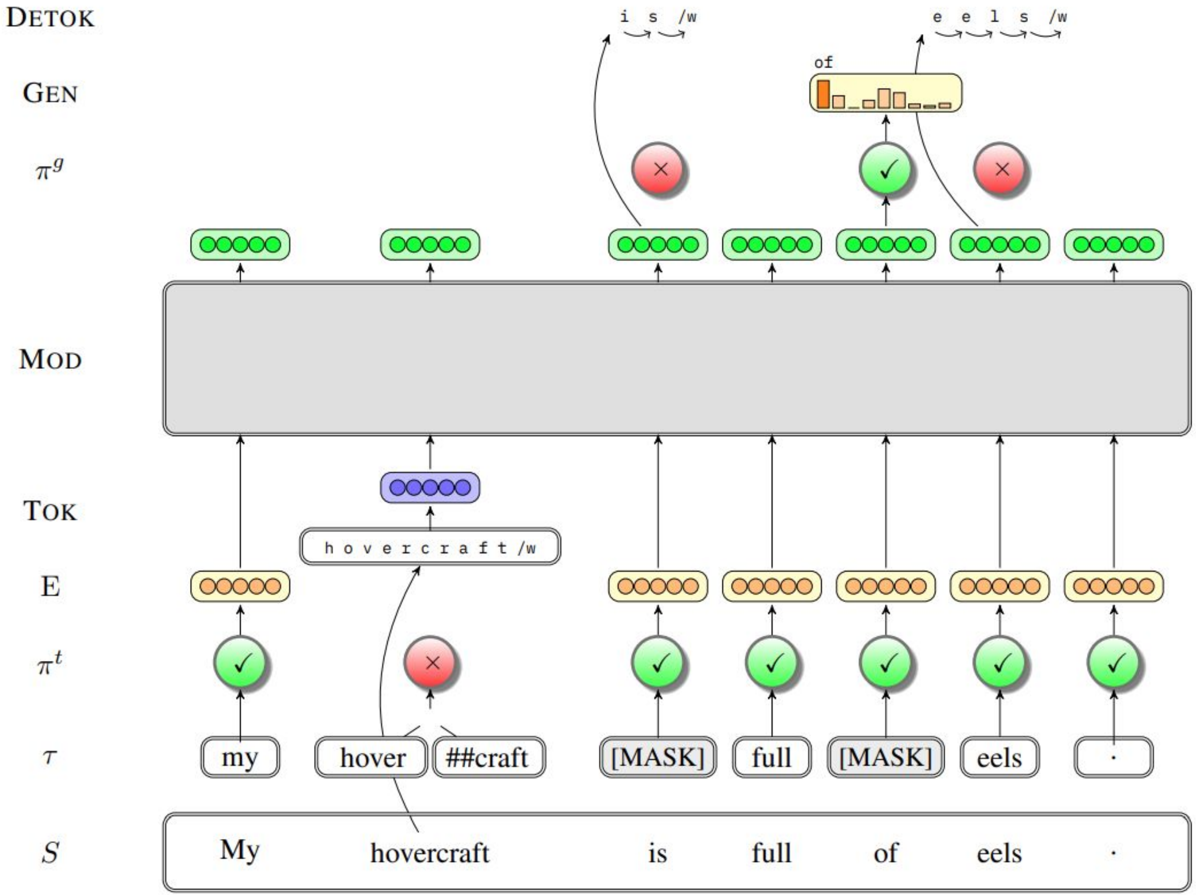


Pinter, Guthrie, Eisenstein. *Mimicking Word Embeddings using Subword RNNs*. EMNLP 2017

Follow-ups

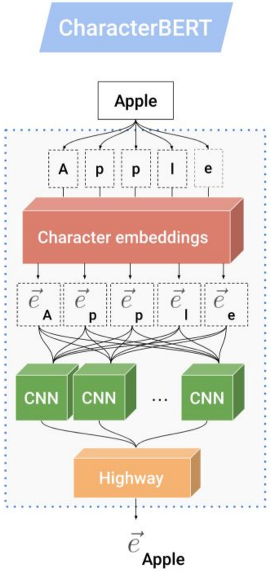
- Different mimicking architectures [Zhao et al. 2018]
- Combining Mimick with large-corpus contexts [Schick & Schütze 2018-19]
- Combining Mimick with downstream contexts [Garneau et al. 2018-19]

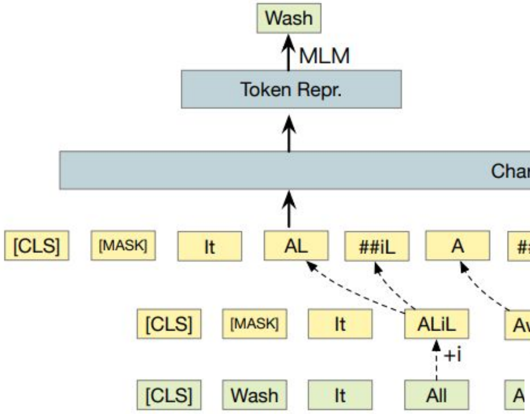
Follow-ups



Pinter, Stent, Dredze, Eisenstein. 2021.
Learning to Look Inside: Augmenting Token-Based Encoders with Character-Level Information. In limbo.

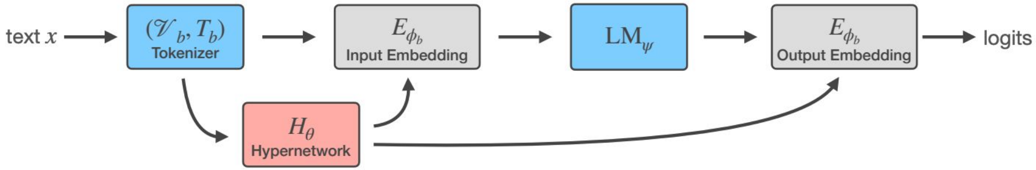
And Similarly





CharBERT: Character-awar

Wentao Ma†, Yiming Cui‡†, Chenglei Si
†State Key Laboratory of Cognitive Intelligence, iFLY
‡Research Center for Social Computing and Informati
Harbin Institute of Technology, Harbin
§iFLYTEK AI Research (Hebei), Langfa
¶University of Maryland, College Park, M



Word-Level
acters

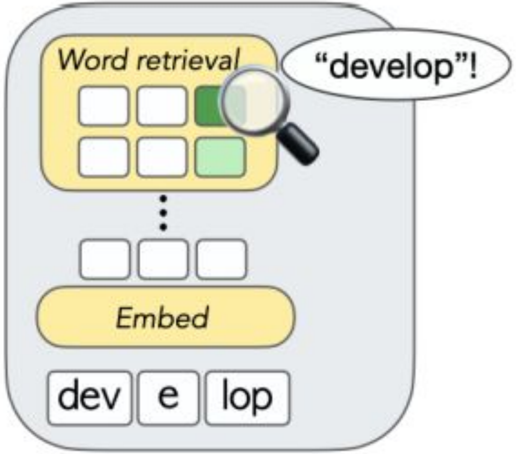
Hiroshi Noji[3],
e,
rance,
apan

Zero-Shot Tokenizer Transfer

Benjamin Minixhofer [SEP] **Edoardo M. Ponti** [CLS] **Ivan Vulić** [SEP]
[SEP] University of Cambridge [CLS] University of Edinburgh

And Also

- CANINE, ByT5, MambaByte
- PIXEL
- Byte Latent Transformers
- Dynamic Hourglass
- H-Net
- “We already beat them”



FROM TOKENS TO WORDS:
ON THE INNER LEXICON OF LLMS

Guy Kaplan, Matanel Oren, Yuval Reif, and Roy Schwartz
The Hebrew University of Jerusalem

Clark et al. 2022
Xue et al. 2022
Wang et al. 2024
Rust et al. 2022
Pagnoni et al. 2024
Nawrot et al. 2024
Hwang et al. 2025
^ like literally last week

Large Language Model

The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation

April 5, 2025 · 12 minute read

Our new Llama 4 models are our first models that use a mixture of experts (MoE) architecture. In MoE models, a single token activates only a fraction of the total parameters. MoE architectures are more compute efficient for training and inference and, given a fixed training FLOPs budget, delivers higher quality compared to a dense model.

vocab 0/0

… will llama 5 finally abolish tokens?

Act II: Join Them

Tokenization is a notorious step of all language modeling pipelines (most commonly the "BPE" algorithm [38], which I'll use interchangeably with "tokenization"), where textual data is

These questions point at the following idea: **is each individual token semantically meaningful?**

The Pipeline

dogmatism → dog mat ism

dogmatism → dogma tism

~~Learn subword vocabulary~~

~~**Statistically-optimized subword segmentation**~~

(e.g., BPE)

Learn contextual embedding model from a large corpus

Fine-tune model parameters on downstream task

Learn **better** subword vocabulary

dogmatism → dogma tism

← How can we get this?

The Objective

The Objective

bite

leash

bark

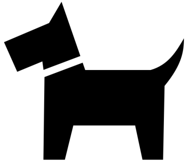
house

food

leash

bark

bone

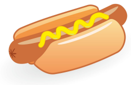


dog

food

mustard

baseball



food

hot

bun

church

faith

pope

##ism

belief

The Objective

bite

leash

bark

house

food

leash

bark

bone

dogm

food

mustard

baseball

food

hot

bun

church

faith

pope

##ism

belief

The Objective

bite

leash

bark

house

food

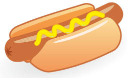
leash

bone

tdog

food

baseball



mustard

food

hot

bun

The idea: prefer subwords that have fewer contexts

bark

pope

##ism

belief

SaGe - a Context-Infused Subword Vocabulary

- SaGe builds on any existing subword vocabulary creation method, adding a **SkipGram*-inspired** objective:

*Mikolov et al. 2013

- For our corpus C and vocabulary V, we learn token embeddings for all t in V, and maximize the **overall log-probability** of contexts given the tokens

$$\mathcal{L}(\mathcal{V},\mathcal{C}) = -\sum_{t\in \mathrm{tok}(\mathcal{C},\mathcal{V})} \sum_{c_j \in W_t} \log\left(\sigma(\boldsymbol{E}_t^{(T)} \cdot \boldsymbol{E}_{c_j}^{(C)})\right)$$

JOIN THEM

- Downstream code (pre-training & fine-tuning) stays **exactly the same**

Yehezkel and **Pinter**. *Incorporating Context into Subword Vocabularies*. EACL 2023.

SaGe - the Algorithm

Initialize V to all possible tokens

While V is too big:

- Tokenize corpus using V
- Compute embedding table
- Compute ablation loss for all tokens
- Throw away least lossy token

Return V

Ablation loss: “if we got rid of this token, how would the skipgram objective be affected?”

$$loss_t \leftarrow \mathcal{L}(\mathcal{V} \setminus \{t\}, \mathcal{C}) - \mathcal{L}(\mathcal{V}, \mathcal{C})$$

https://github.com/MeLeLBGU/SaGe

SaGe - the Algorithm

Initialize V to large BPE-trained vocabulary

While V is too big:

- Tokenize corpus using V
- Compute embedding table
- Compute ablation loss for all tokens
- Throw away least lossy token

Return V

Problem: this takes too much time!

Fix #1: start with an approximately good (but still large) vocabulary

SaGe - the Algorithm

Problem: this takes too much time!

Fix #2: only compute loss for candidate set of “bottom” tokens, refreshed every *m* iterations

Fix #3: remove *k* tokens at once, every iteration

Initialize V to large BPE-trained vocabulary

While V is too big:

- Tokenize corpus using V
- Compute embedding table
  - Every *m* steps: refresh bottom set
- Compute ablation loss for tokens in current set of bottom tokens
- Throw away *k* least lossy tokens

Return V

SaGe - the Algorithm

Problem: this takes too much time!

Fix #4: recompute embedding table only every *l* calculations of bottom set

Initialize V to large BPE-trained vocabulary

While V is too big:

- Tokenize corpus using V
- Every *l x m* steps: compute embedding table
- Every *m* steps: refresh bottom set
- Compute ablation loss for tokens in current set of bottom tokens
- Throw away *k* least lossy tokens

Return V

Did Ambiguous Tokens Go Away?

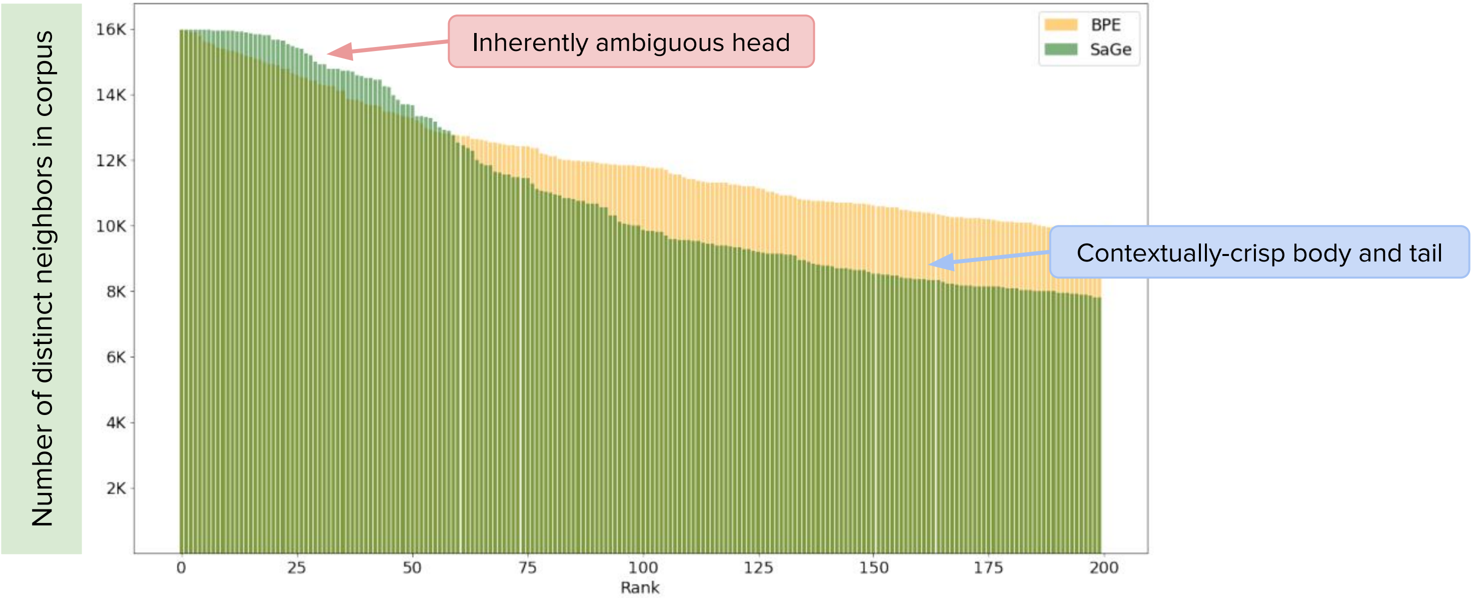
- In English BPE, the token *og* emerges in various contexts

- With SaGe, it is ablated due to its ambiguity, replaced by two possible behaviors:
  - Larger tokens containing it are retained, being more contextually salient; **or**
  - No larger tokens are frequent enough, breaking the token down to characters (inherently ambiguous)

_His _son _Raj ash ri _Sud h ak ar _has _p enn ed **_dial og ues** _and _songs _for _some _films _that _were _dubbed _into _Telugu .
_This _gene _is _a **_pseud og ene** _in _humans _and _most _other _prim ates .
_The **_St o og es** _work _for _Mir acle _Det ective _Agency ,

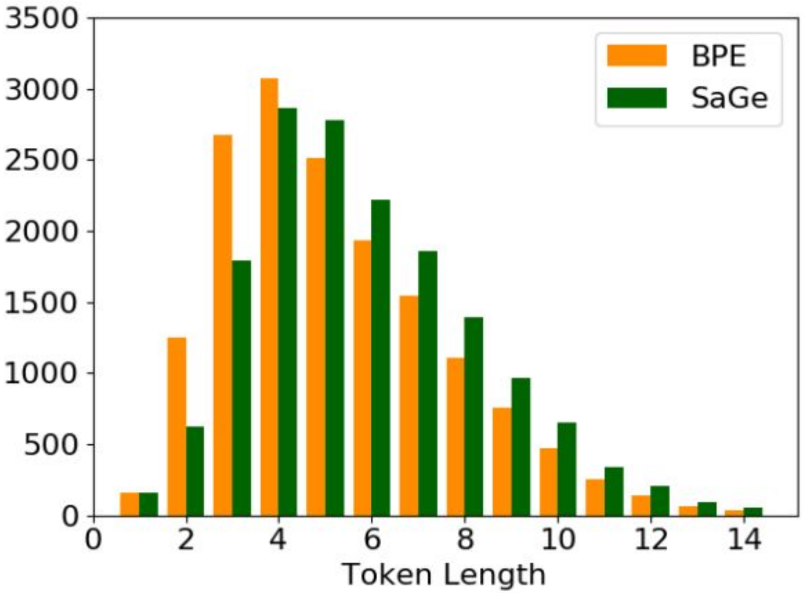
Vocabulary Analysis

- SaGe tokens have fewer contexts
  - (Not conditioned on frequency)



Vocabulary Analysis

- SaGe tokens have fewer contexts
  - (Not conditioned on frequency)
- SaGe produces longer tokens

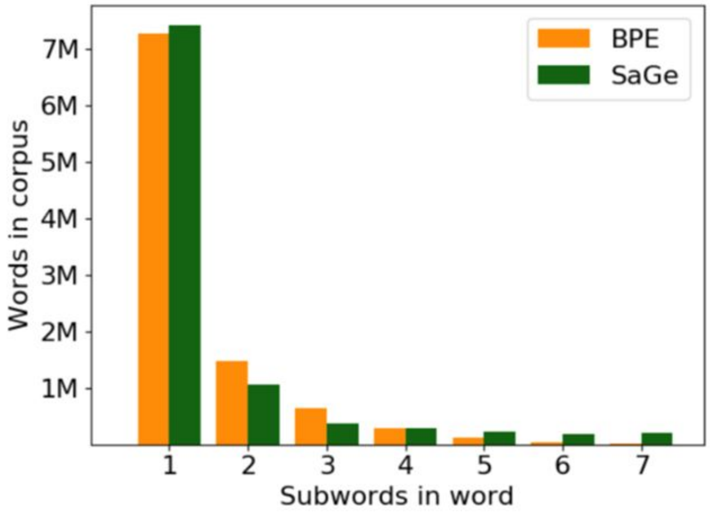


We attribute this to SaGe's ability to remove tokens that are *intermediate* in BPE's vocabulary formation:

BPE t h e → th e → the
SaGe t h e ~~→ th e~~ → the

Vocabulary Analysis

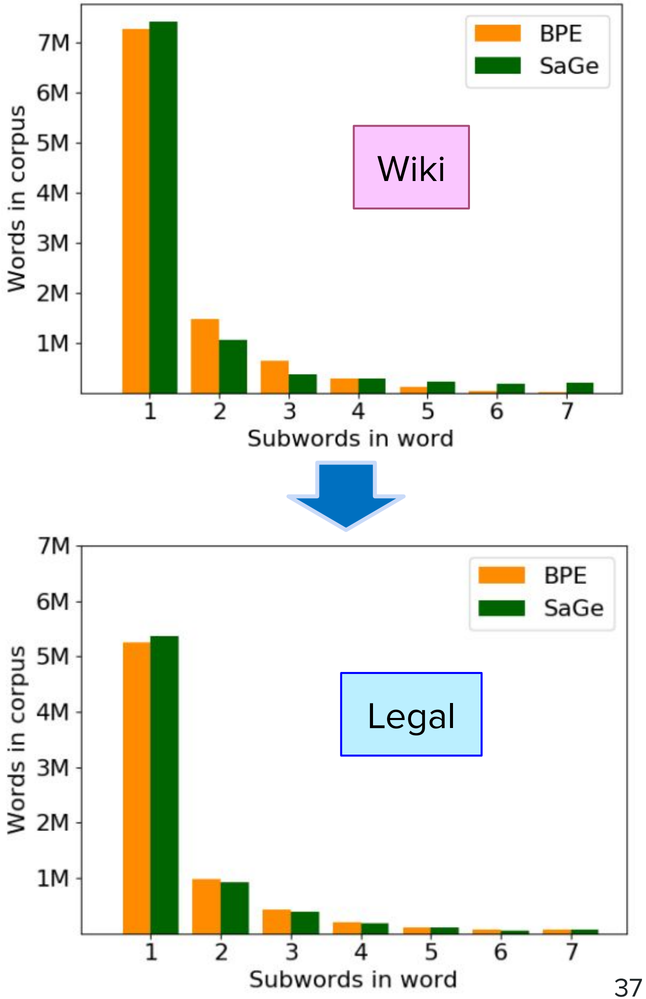
- SaGe tokens have fewer contexts
  - (Not conditioned on frequency)
- SaGe produces longer tokens
- SaGe retains more full words, but breaks down others into many more tokens



Due to the inherently ambiguous single-character tokens

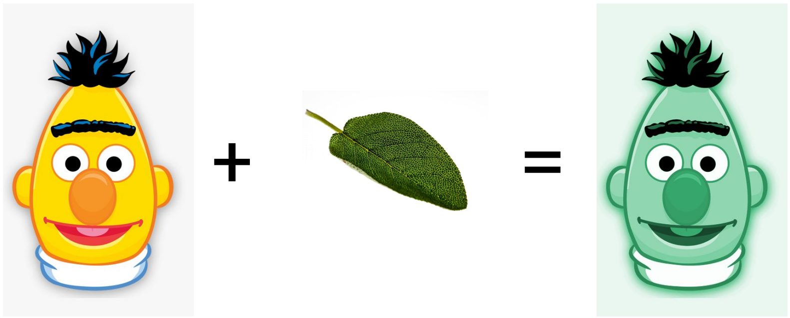
Vocabulary Analysis

- SaGe tokens have fewer contexts
  - (Not conditioned on frequency)
- SaGe produces longer tokens
- SaGe retains more full words, but breaks down others into many more tokens
- Statistics are robust when testing on different domains



Downstream Evaluation

- We pretrained BERT models using BPE and SaGe in English and Turkish
- Implementation: “24-hour academic BERT” (Izsak et al., 2021)
- Data: Wikipedia



Downstream Performance

- Evaluation on English GLUE and NER (CoNLL 2003), Turkish XNLI and NER

GLUE

MRPC (F1)	MNLI (Acc %)	COLA (Matt.)	QNLI (Acc %)	SST2 (Acc %)	STSB (Pear.)	QQP (Acc %)	XNLI_{tur} (Acc %)
.7918	62.57	.0777	66.17	80.54	.3094	82.41	41.20
.8004	64.00	.0985	74.83	79.85	.3387	84.23	46.46

NER

English	Turkish
.7142	.4660
.7502	.5475

Conclusion

- Context is important as far back as pre-pre-training tokenization schemes
- SaGe is a context-aware tokenizer incorporating the SkipGram objective
  - Achieves better results on downstream tasks on 2 typologically-distant languages
  - Both sequence and token levels
- **SaGe is plug-and-play**. No need to change code in LLMs, only the subword vocabulary file
- *We believe further work can improve results even further*
  - *Extend to other languages and tasks*
  - *Optimize the algorithm and remove some of the "fixes"*

Subsequent work

SaGe 2.0 (released):
faster;
no need for bottom set (all vocab ablations computed at once)

Initialize V to large BPE-trained vocabulary

While V is too big:

- Tokenize corpus using V
- Every *l* x *m* steps: compute embedding table
- Every *m* steps: refresh bottom set
- Compute ablation loss for tokens in current set of bottom tokens
- Throw away *k* least lossy tokens

Return V

https://github.com/MeLeLBGU/SaGe

Interlude: Convince Them that It's Better

- Downstream performance
  - Which task? Even this isn't clear, many are just evaluating bit-per-token or perplexity, etc.

- The Correlation Challenge
  - When does higher intrinsic score indicate better downstream performance?

- Without good intrinsic measures, it's only **beat them** that can make a dent
  - And even that won't guarantee adoption

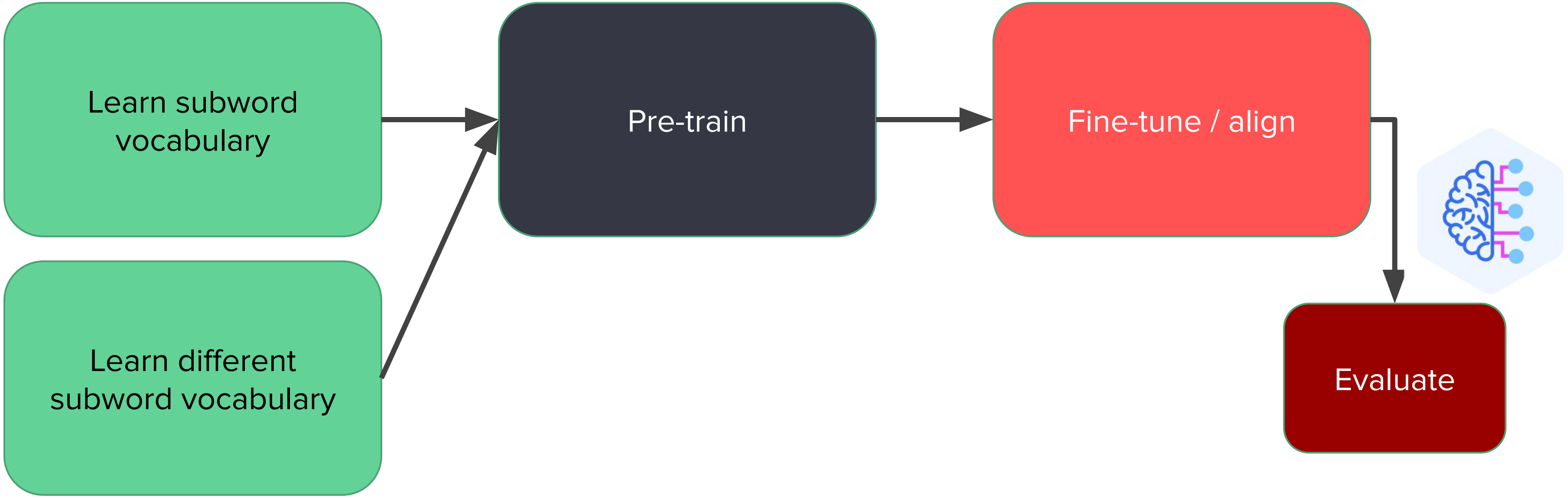
Beyond Text Compression: Evaluating Tokenizers Across Scales

Jonas F. Lotz* University of Copenhagen, Denmark & ROCKWOOL Foundation Research Unit

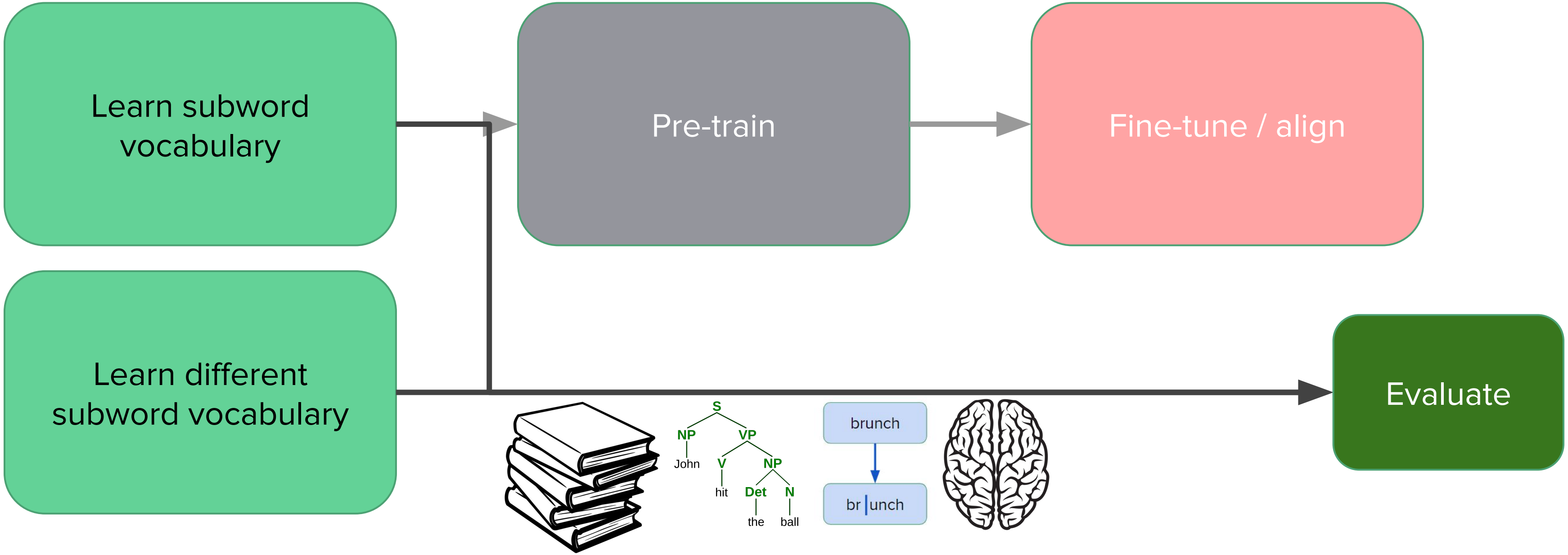
António V. Lopes, **Stephan Peitz**, **Hendra Setiawan**, **Leonardo Emili** Apple

	Multiple-choice	Summarization	Machine translation
COMPRESSION	− **0.59****	−0.09	0.77**
CARDINALITY	0.29*	−0.09	− **0.79****
AUC	0.19	0.14	0.77**
POWER LAW	0.0	0.14	0.78**
SLOPE	0.0	−0.43	−0.44*
Across scales	0.33	−0.07	0.87*

Convince Them that It’s Better



Convince Them that It’s Better



Cognitive Evaluation of Tokenizers

not directly measurable

Easy for humans ⟷ **Easy for tokenizers**

↓ ↓

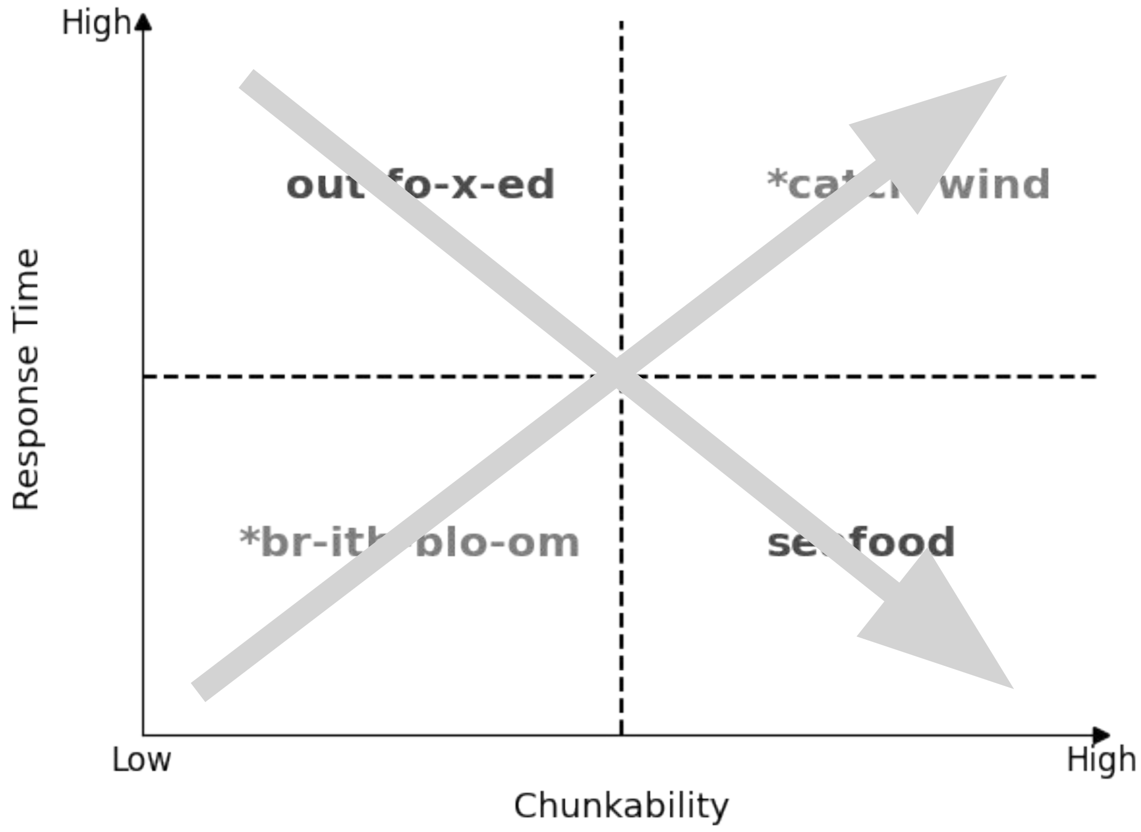
measurable

Fast processing ⟷ **Fewer subwords**

Response time

$$\text{chunkability} = 1 - \frac{\#\text{tokens}}{\#\text{chars}}$$

Lisa Beinborn and **Yuval Pinter**. *Analyzing Cognitive Plausibility of Subword Tokenization*. EMNLP 2023



The British Lexicon Project: Lexical decision data for 28,730 monosyllabic and disyllabic English words

Emmanuel Keuleers, Paula Lacey, Kathleen Rastle & Marc Brysbaert

Practice effects in large-scale visual word recognition studies: a lexical decision study on 14,000 Dutch mono- and disyllabic words and nonwords

Emmanuel Keuleers* Kevin Diependaele Marc Brysbaert

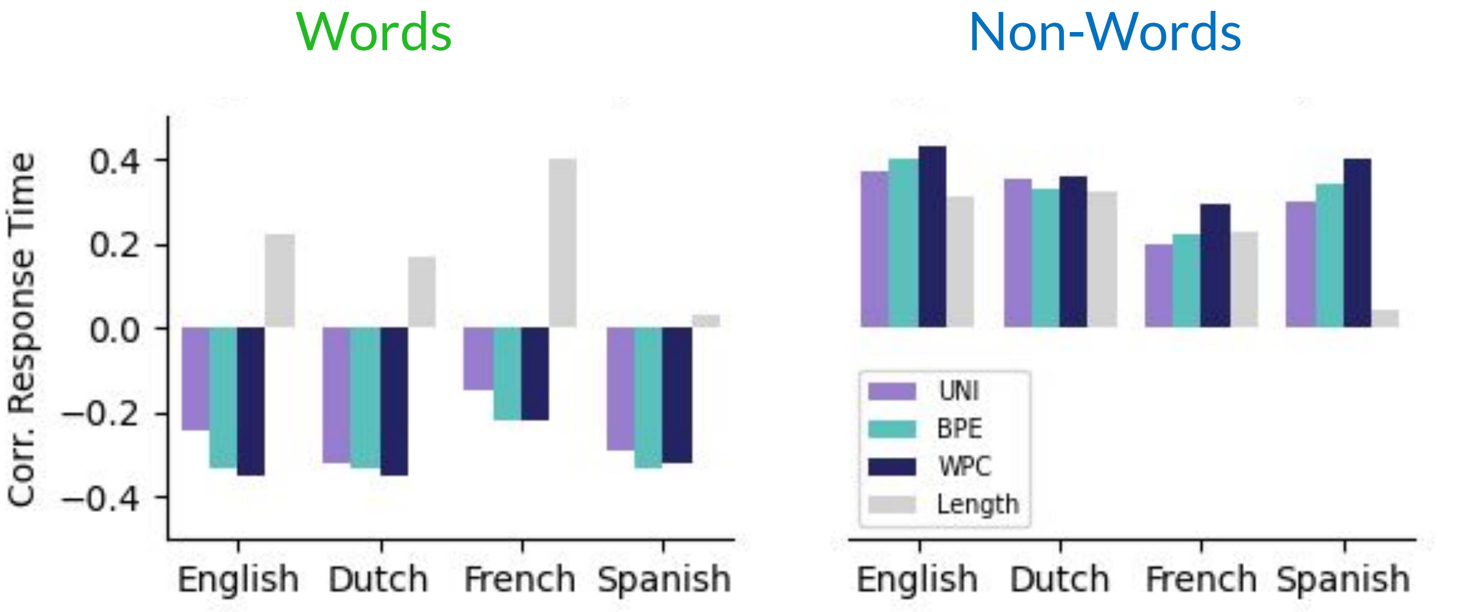
The French Lexicon Project: Lexical decision data for 38,840 French words and 38,840 pseudowords

Ludovic Ferrand, Boris New, Marc Brysbaert, Emmanuel Keuleers, Patrick Bonin, Alain Méot, Maria Augustinova & Christophe Pallier

SPALEX: A Spanish Lexical Decision Database From a Massive Online Data Collection

Jose Armando Aguasvivas[1] Manuel Carreiras[1,2] Marc Brysbaert[3] Paweł Mandera[3] Emmanuel Keuleers[4] Jon Andoni Duñabeitia[1,5*]

Results



WordPiece performs best, UnigramLM disappoints
(contra Bostrom & Durrett 2020, whose eval is statistics + downstream)

Intrinsic Eval Benchmark

- (English only)

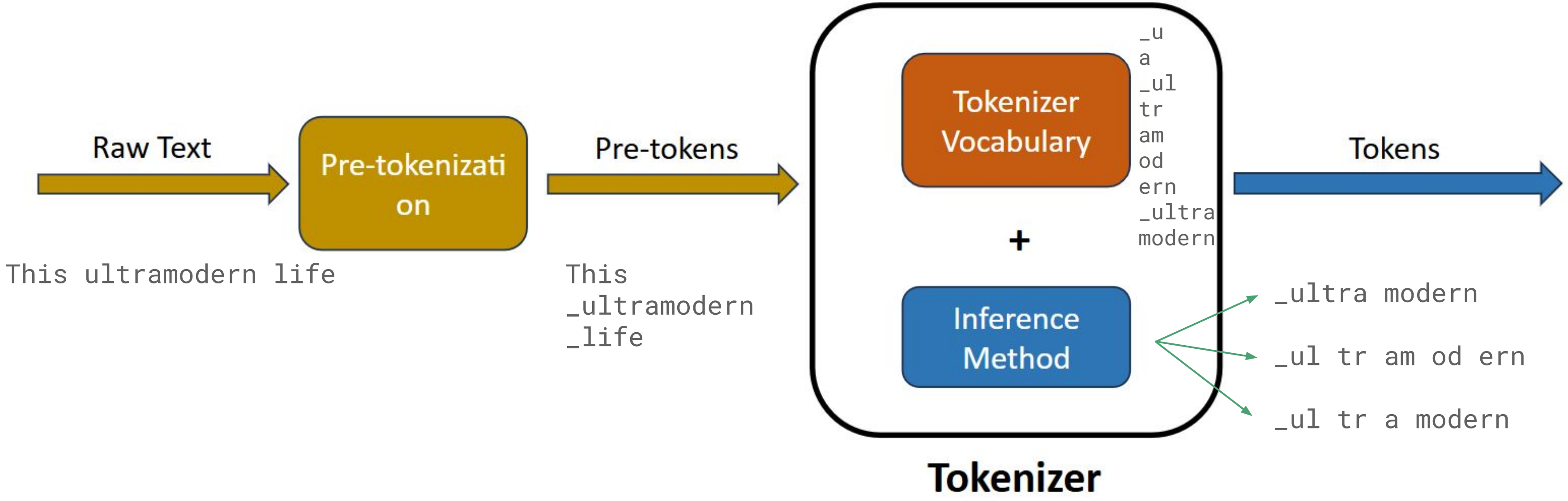
- Combining:
  - Several morphological resources
  - Cognitive data
  - Efficiency metrics

- [Link]

Resource	Reference
LADEC	paper
MorphoLex	paper
MorphyNet	paper
DagoBert	paper
UniMorph	paper
UnBlend	paper
CompoundPiece	paper
Cognitive data	paper
tokenization-scorer	paper

Uzan, Schmidt, Tanner, **Pinter**. *Greed is All You Need: An Evaluation of Tokenizer Inference Methods*. ACL 2024 (Outstanding Paper)

Bridge: Tokenization is also a Pipeline



Act III: Fix Them

Fix Them

inference

- BPE-Dropout: occasionally forget to merge

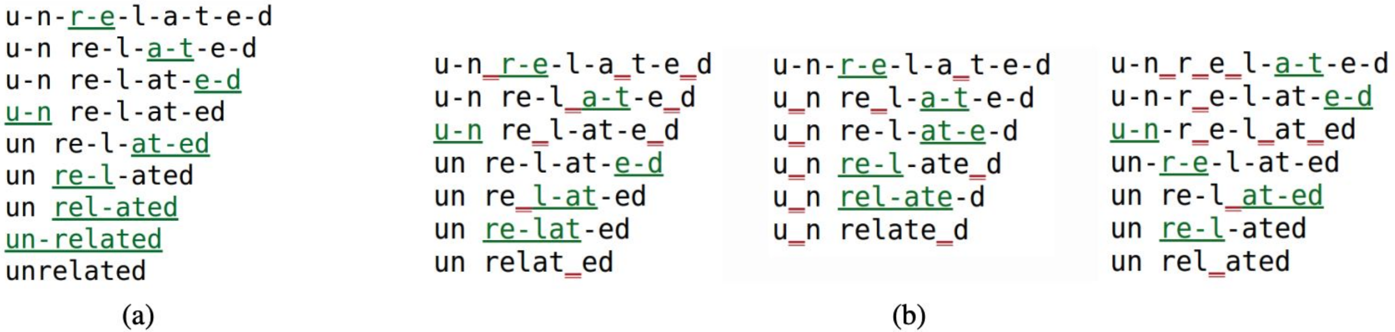


Figure 1: Segmentation process of the word '*unrelated*' using (a) BPE, (b) *BPE-dropout*. Hyphens indicate possible merges (merges which are present in the merge table); merges performed at each iteration are shown in green, dropped – in red.

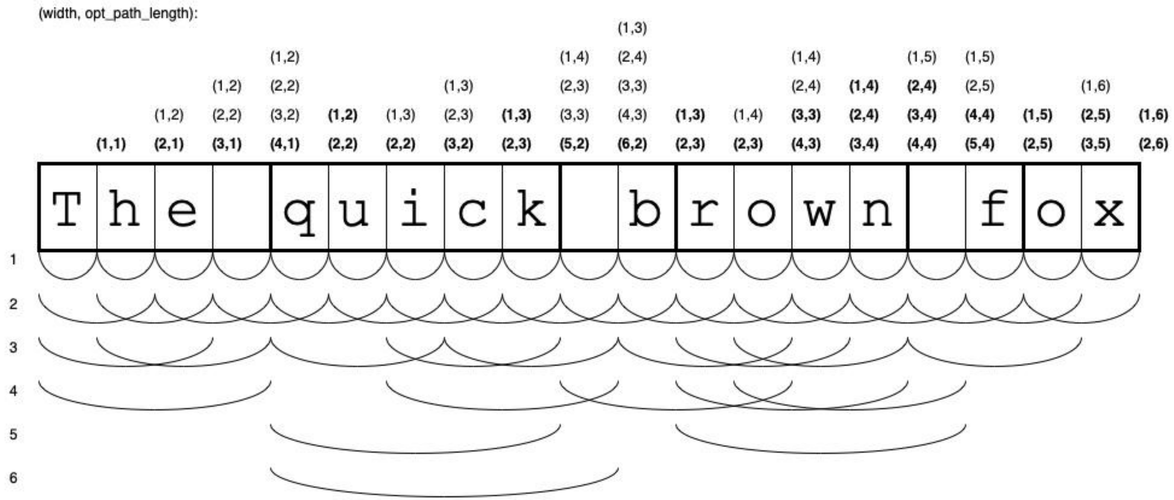
BPE-Dropout: Simple and Effective Subword Regularization

Ivan Provilkov[*1,2] **Dmitrii Emelianenko**[*1,3] **Elena Voita**[4,5]

Fix Them

inference

- PathPiece: keep the vocabulary but make inference maximally compressive



Schmidt, Reddy, Zhang, Alameddine, Uzan, **Pinter**, Tanner. *Tokenization Is More Than Compression.* EMNLP 2024

Fix Them

inference

- FLOTA: keep the vocabulary but find the l o n g e s t token in inference

An Embarrassingly Simple Method to Mitigate und es ira ble Properties of Pretrained Language Model Tokenizers

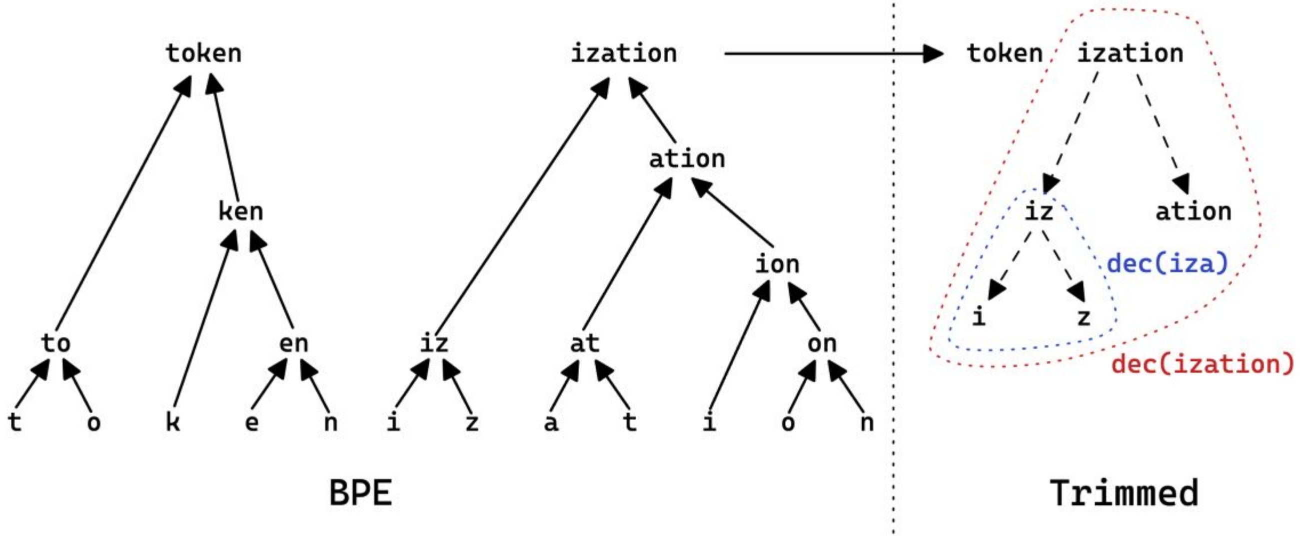
Valentin Hofmann[*‡], Hinrich Schütze[‡], Janet B. Pierrehumbert[†*]

[*]Faculty of Linguistics, University of Oxford
[†]Department of Engineering Science, University of Oxford
[‡]Center for Information and Language Processing, LMU Munich

Fix Them

vocab

- Trimmed-BPE: get rid of rare intermediate tokens post-vocab-building

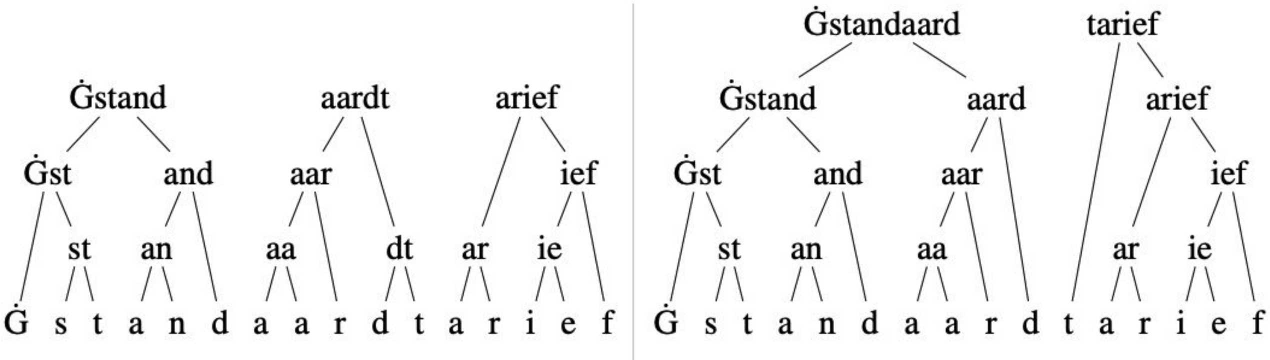


Cognetta, Hiraoka, Okazaki, Sennrich, **Pinter**. *An Analysis of BPE Vocabulary Trimming in Neural Machine Translation*. 2024

Fix Them

vocab

- BPE-knockout: trim post-vocab-building, with the help of morphology



(a) BPE

(b) BPE-knockout (due to knockout of the merge *d+t*)

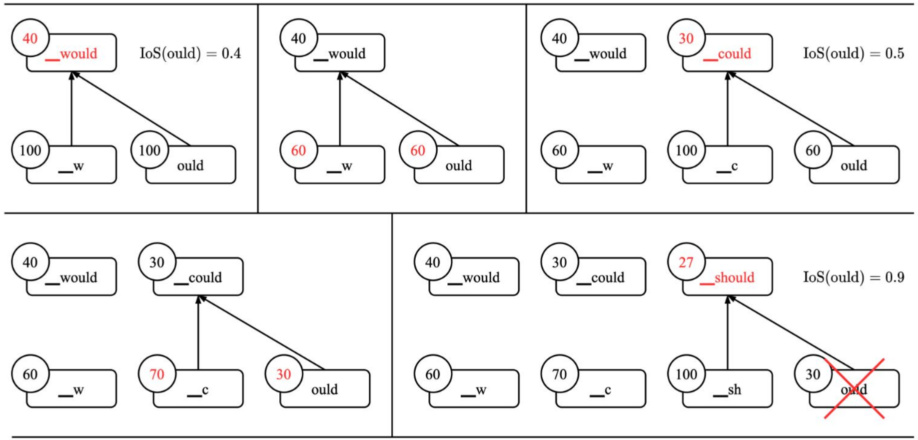
BPE-knockout: Pruning Pre-existing BPE Tokenisers with Backwards-compatible Morphological Semi-supervision

Thomas Bauwens and **Pieter Delobelle**

Fix Them

vocab

- Picky-BPE: get rid of tokens **during vocab-building**, when it “makes sense”



$$\mathrm{IoS}(x_1 \mid x_1, x_2) = \frac{f_p(x_1, x_2)}{f_t(x_1)}$$

Figure 2: Picky BPE tokenization example. Token frequencies are demonstrated in the corresponding circles and are updated on merges. Token “ould” is removed only after merging into three common tokens containing it. The corresponding IoS values are visualized on every merge. Once IoS becomes greater or equal to the threshold $\mathcal{T}$, 0.9 in this example, the token “ould” is removed.

BPE Gets Picky: Efficient Vocabulary Refinement During Tokenizer Training

Pavel Chizhov*[1,2,3] **Catherine Arnett***[2,4] **Elizaveta Korotkova**[3] **Ivan P. Yamshchikov**[1,2]

Fix Them

pretokenization

- Change the regular expression

```
'(?:[sdmt]|ll|ve|re)   # English contractions such as 'm and 've
| ?\p{L}+              # Optional space + one or more letters
| ?\p{N}+              # Optional space + one of more numbers
| ?[^\s\p{L}\p{N}]+    # Optional space + one or more punctuation(-ish)
| \s+(?!\S)            # Whitespace not followed by non-whitespace
| \s+"""
```

```
'(?i:[sdmt]|ll|ve|re)          # English contractions, same as GPT-2
|[^\r\n\p{L}\p{N}]?+\p{L}+     # single space/tab/punctuation + letters
|\p{N}{1,3}                    # 1-3 digits, no leading space
| ?[^\s\p{L}\p{N}]++[\r\n]*    # optional space, punctuation, line breaks
|\s*[\r\n]                     # any whitespace ending in \r or \n
|\s+(?!\S)                     # any whitespace preceding a non-space
|\s+                           # any whitespace
```

Fix Them

pretokenization

- Pretokens don’t control us anymore

Boundless Byte Pair Encoding:
Breaking the Pre-tokenization Barrier

Craig W. Schmidt, Varshini Reddy & Chris Tanner*
Kensho Technologies
Cambridge, MA 02138, USA
{craig.schmidt,varshini.bogolu,chris.tanner}@kensho.com

Yuval Pinter
Department of Computer Science
Ben-Gurion University of the Negev
Beer Sheva, Israel
uvp@cs.bgu.ac.il

```
[['T', 'i', 'p'], [' ', 'o', 'f'], [' ', 't', 'h', 'e'], [' ', 'h', 'a', 't']]
[['T', 'i', 'p'], [' ', 'o', 'f'], [' t', 'h', 'e'], [' ', 'h', 'a', 't']]
[['T', 'i', 'p'], [' ', 'o', 'f'], [' t', 'he'], [' ', 'h', 'a', 't']]
[['T', 'i', 'p'], [' ', 'o', 'f'], [' the'], [' ', 'h', 'a', 't']]
[['T', 'i', 'p'], [' ', 'o', 'f'], [' the'], [' ', 'h', 'at']]
[['T', 'i', 'p'], [' o', 'f'], [' the'], [' ', 'h', 'at']]
[['T', 'i', 'p'], [' of'], [' the'], [' ', 'h', 'at']]
[['T', 'i', 'p'], [' of'], [' the'], [' h', 'at']]
[['T', 'i', 'p'], [' of the'], [' h', 'at']]
[['T', 'ip'], [' of the'], [' h', 'at']]
[['T', 'ip'], [' of the'], [' hat']]
[['Tip'], [' of the'], [' hat']]
```

Fix Them

pretokenization

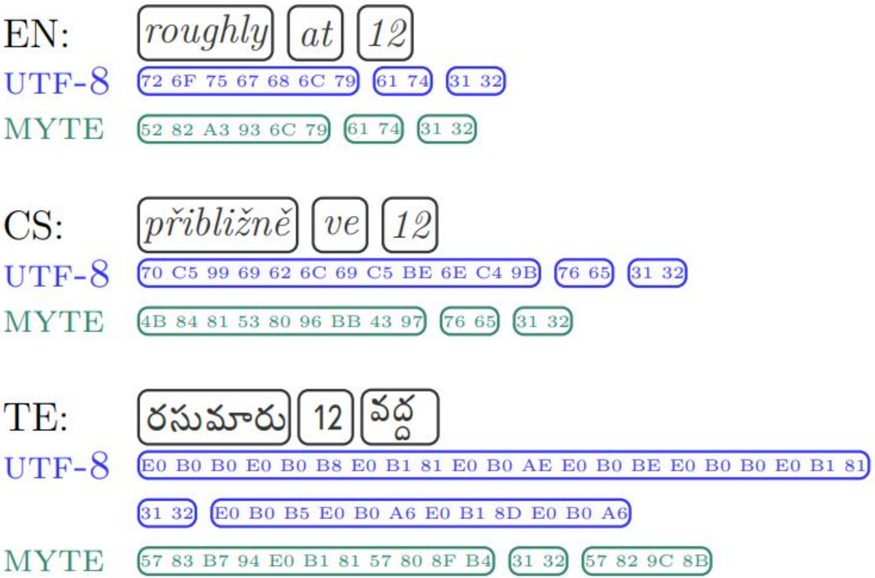
- Pretokens don’t control us anymore

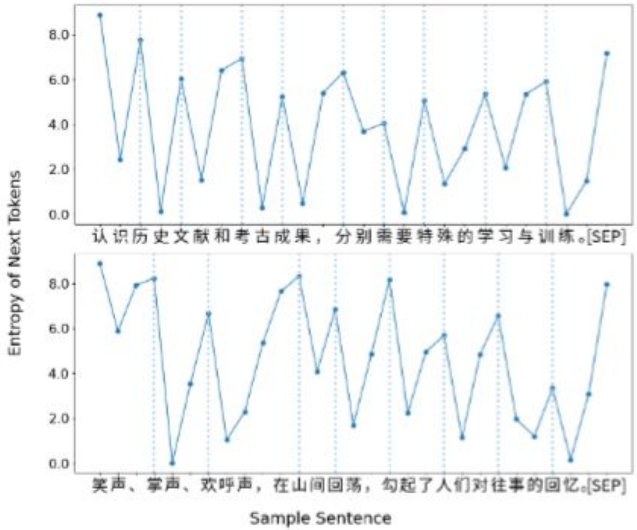
SuperBPE: Space Travel for Language Models

*Alisa Liu[♡♠] *Jonathan Hayase[♡]
Valentin Hofmann[◊♡] Sewoong Oh[♡] Noah A. Smith[♡◊] Yejin Choi[♠]
[♡]University of Washington [♠]NVIDIA [◊]Allen Institute for AI

Fix Them

pre-pretokenization





MYTE: Morphology-Driven Byte Encoding for Better and Fairer Multilingual Language Modeling

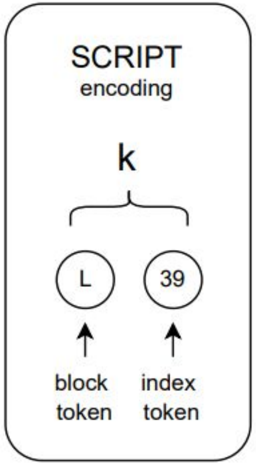
Tomasz Limisiewicz[1†*] Terra Blevins[2] Hila Gonen[2]
Orevaoghene Ahia[2] Luke Zettlemoyer[2]
[1]Faculty of Mathematics and Physics, Charles University in Prague
[2]Paul G. Allen School of Computer Science and Engineering, University of Washington

Entropy-Driven Pre-Tokenization for Byte-Pair Encoding

Yifan Hu[*1] Frank Liang[*1] Dachuan Zhao[*1] Jonathan Geuter[1 2] Varshini Reddy[3] Craig W. Schmidt[3]
Chris Tanner[3]

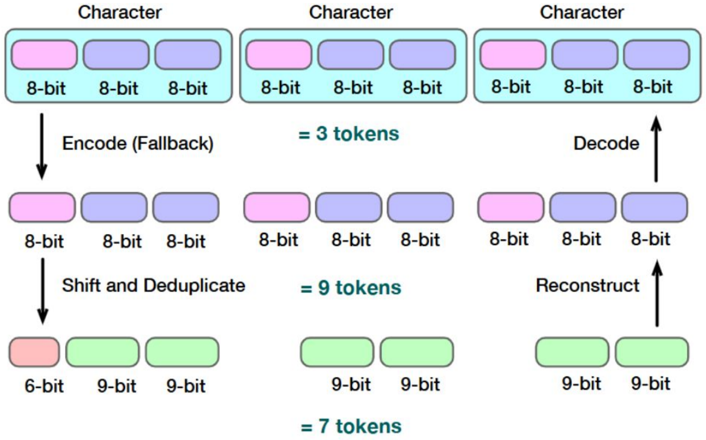
Fix Them

pre-pretokenization



BPE Stays on SCRIPT: Structured Encoding for Robust Multilingual Pretokenization

Sander Land [1] Catherine Arnett [2]



Bit-level BPE: Below the byte boundary

Sangwhan Moon
Google LLC
sangwhan@iki.fi

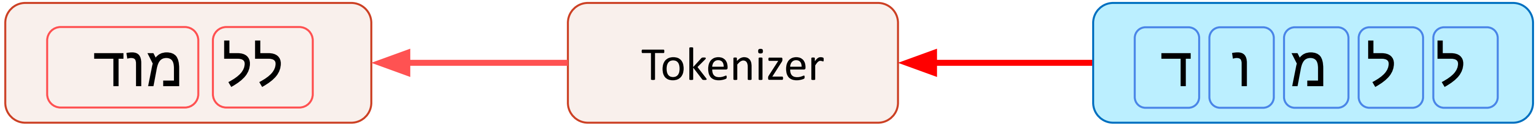
Tatsuya Hiraoka
MBZUAI
tatsuya.hiraoka@mbzuai.ac.ae

Naoaki Okazaki
Institute of Science Tokyo
okazaki@c.titech.ac.jp

Fix Them: Semitic Drift

pre-pretokenization

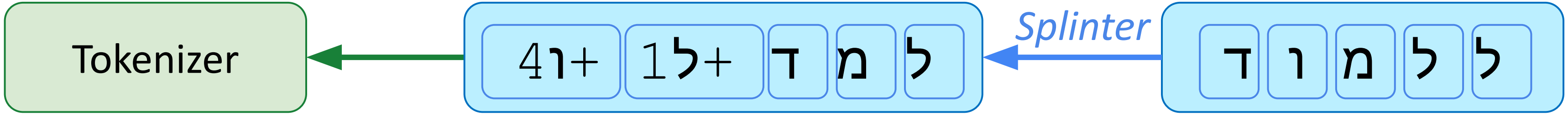
- Existing segmentation methods assume **concatenative** text
- Hebrew, Arabic, Malay, Georgian (and others) don't follow this rule



Fix Them: Semitic Drift

pre-pretokenization

- Existing segmentation methods assume **concatenative** text
- Hebrew, Arabic, Malay, Georgian (and others) don’t follow this rule
- We propose *Splinter*, a learned pre-processing step that re-linearizes the text into concatenative morphemes

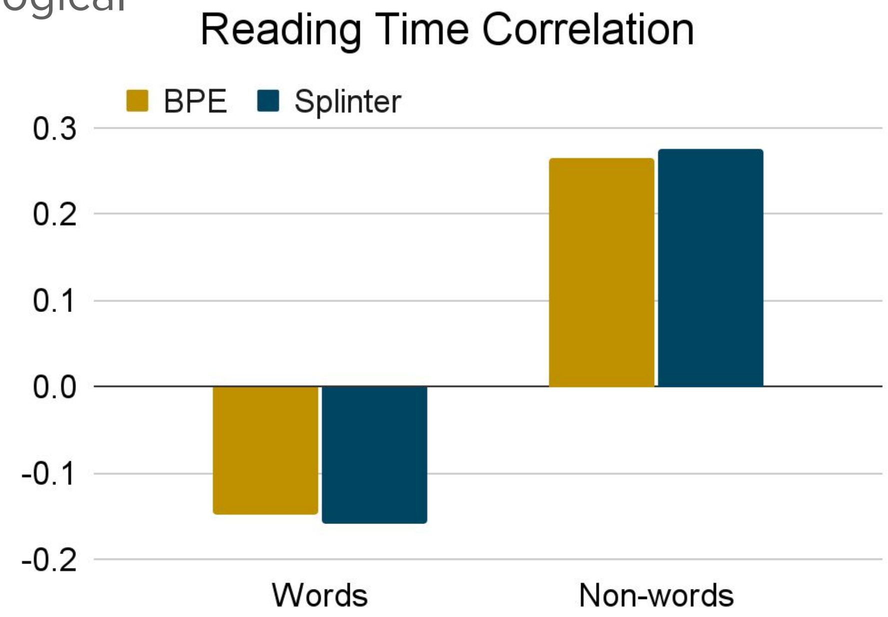


Bar Gazit, S Shmidman, A Shmidman, **Pinter**. *Splintering Nonconcatenative Languages for Better Tokenization.* Findings of ACL 2025.

Splinter - Evaluation

- We can’t even evaluate with morphological data (which assumes concatenative morphology)

- And so:

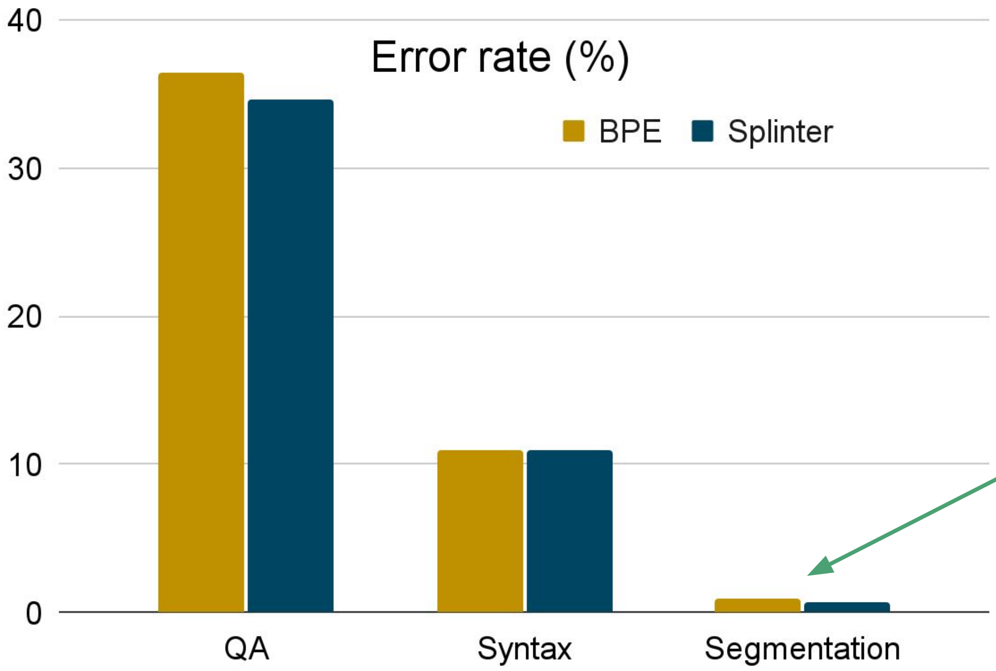


HeLP: The Hebrew Lexicon project

Roni Stein, Ram Frost & Noam Siegelman

Splinter - Evaluation

- But we can evaluate downstream:



(This is a 20% difference)

Where Do We Go From Here

- I think we can put our eggs in more than one basket

- I think “model conservatism” has its advantages ← **we need more “join them”**

- I think we need to look **everywhere** for more evaluation opportunities

- I think we can talk about this!
    - TokShop
    - *Token ##ization* Discord server
    - Applied Sciences Special Issue on Atomic Respresentations

Merge with us

The Token ##ization Discord

I Can Talk About Tokenization Forever

- Toolkits
- Is tokenization still “underexplored”?
- Dealing with auxiliary problems builds a healthy ecosystem that fosters long-lived lines of research
  - Formal properties of tokenizers
  - Scaling laws and training limits
- And more!
- Catch me anytime today or snipe me at the 15:30 panel

Thanks

Machines Learning Language MELEL

Funding:




[Shaked asked for no photo]








Thank you!

yuvalpinter.com

uvp@cs.bgu.ac.il

Machines Learning Language MELEL

- beat them: pixels, bytes, canines
    - (the bpe strawman)
- fix them (i can fix him meme): dropout, pathpiece, picky, superwords
    - fixing way down in the plumbing: myte, script, splinter & hu-etal-entropy (if you care about a language show it)
- join them: sage, bytespan, splinter (?), zett (?), anything plug-and-play that can lock into existing code pipelines
- loose threads of lines of work (encoding, pretok, vocab, inf, eval, multiling, downstream effects [real tasks and charbench stuff]) not all coming together
    - the diacritics thing wrt pretok - kyle and moi
    - ppl are still not doing vocab algos at all - are sage and bytespan really the only novel (unsupervised) ones since unigramlm?
- bpe's ever-present hidden tokens problem (knockout, trimmed, picky, etc.) and the various ramifications such as the min token counts rising (see chart from craig in screenshot)
- bytes, encoding, characters, a rabbit hole beyond us
- my maxim of "these things have to make sense for language -> make sense for (L)LMs -> make sense for contextual understanding"
- zoom-out science vs. zoom-in
- are we still "underexplored"?
- arguments we need to push back on
    - "everything gets fixed in the transformer layers"
        - Roy S's paper (full-word representations exist in the transformer layers)
- the evaluation crisis?
    - the bottom-up (bpe) vs. top-down (unigram) debate
    - quality vs. efficiency? BPC vs/ fertility? what about morph, cog?
    - (bytespan found a coverage "bug" in our morpho benchmark impl, p.6)
- the strawberry fixation? (this isn't it + charbench) the magikarp thing?
- what pretokenization gives us
    - and what superwords don't solve in a theoretical sense, turning the elephant in the room to a capybara in the room
        - can be framed as "no-pretok has two main disadvantages: inefficiency and empirical suckiness; superwords avoid the latter without totally relaxing the former"
- the spooky magic of (tokenization) inference
- (latter two-related) maybe a map of all the NLP pipeline with where tokenization happens and has effects? like a pre-embedding continent, then the bias stuff affecting downstream, multilingual rivers, etc. <- this might work (also?) better for the SI paper.

…t10n?

The Hot Takes for Today

- “How many r’s are in *strawberry*” is not interesting. Use a script.

Leftovers (?)

- Is tokenization still "underexplored"?
  - Does everything really "sort itself out" in the transformer layers?
    - Yes and no: Schwartz, empirical evidence, but also off-domain, multilingual, etc.

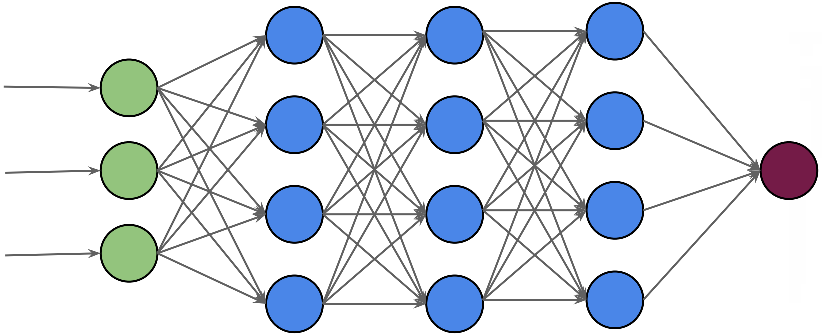
- What do we do about everybody still using vanilla BPE?
  - (Even if they rebrand it, like "TikToken" or "Neo tokenizer" or whatever)

- What are the biggest elephants in our room?
  - Evaluation
  - Sweeping pretokenization under the rug
  - Multilingual and crosslingual
    - Lead in to SPLINTER and Hu-etal-entropy ("if you care about a language, show it")

- The fun of defining auxiliary questions [formal properties, scaling laws] ← leading towards a "mini-NLP" ecosystem that fosters long-lived lines of research

Language Models are Inherently Mismatched

`The train left the station`



Symbolic

Discrete

Real-valued

Smooth

Vocabulary vs. Inference

	Greedy	Merges	Likelihood
Byte Pair Encoding (BPE)	Compatible	Default	
UnigramLM	Compatible		Default
WordPiece	Default	Compatible	
SaGe	Default		Compatible

Train Vocab using One, Infer with Another?

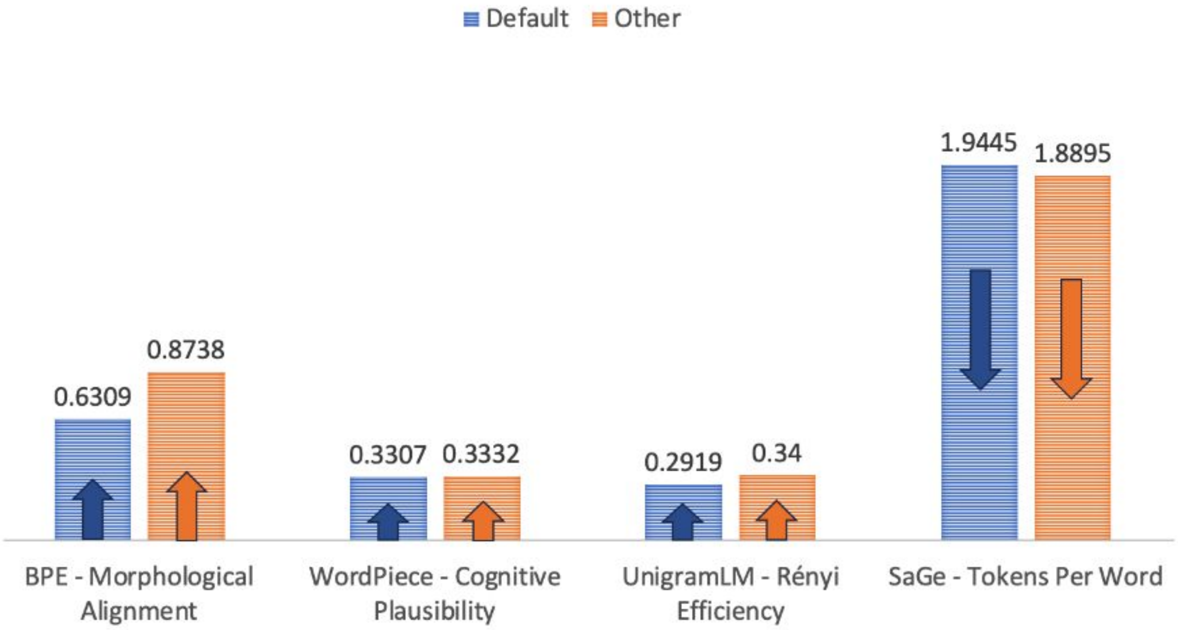
Vocab	Inference method
BPE	longest prefix
	longest suffix
	longest token
	least tokens
	det. merges
	dropout merge
WordPiece	*longest prefix*
	longest suffix
	longest token
	least tokens
UnigramLM	longest prefix
	longest suffix
	longest token
	least tokens
	likelihood
SaGe	*longest prefix*
	longest suffix
	longest token
	least tokens
	likelihood†

Resource	Reference
LADEC	paper
MorphoLex	paper
MorphyNet	paper
DagoBert	paper
UniMorph	paper
UnBlend	paper
CompoundPiece	paper
Cognitive data	paper
tokenization-scorer	paper

Uzan, Schmidt, Tanner, **Pinter**. *Greed is All You Need: An Evaluation of Tokenizer Inference Methods.* ACL 2024 (Outstanding Paper)

Train Vocab using One, Infer with Another?

- The default inference method is **constantly outperformed** on some measure

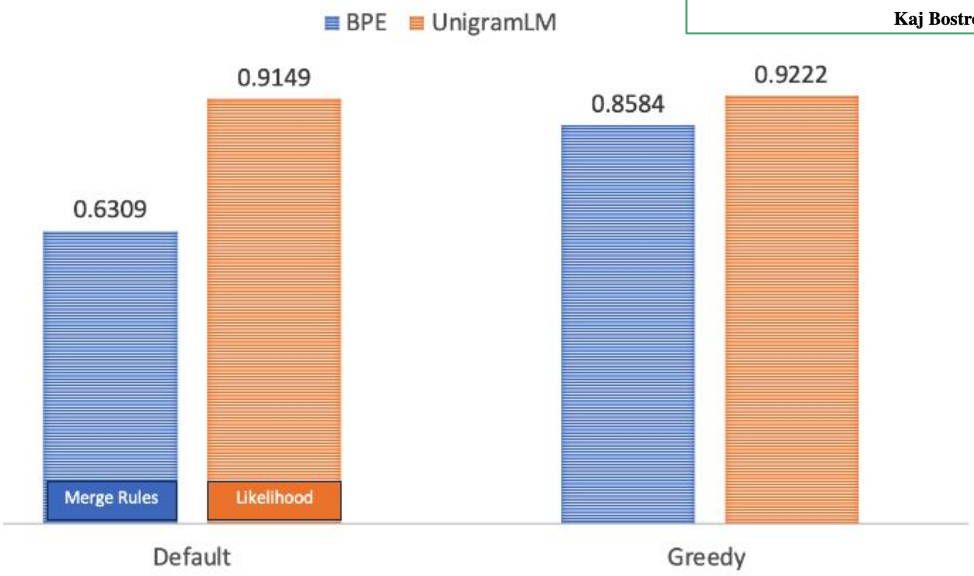


Train Vocab using One, Infer with Another?

- The **morphological gap** between Unigram and BPE can be attributed mainly to the inference method

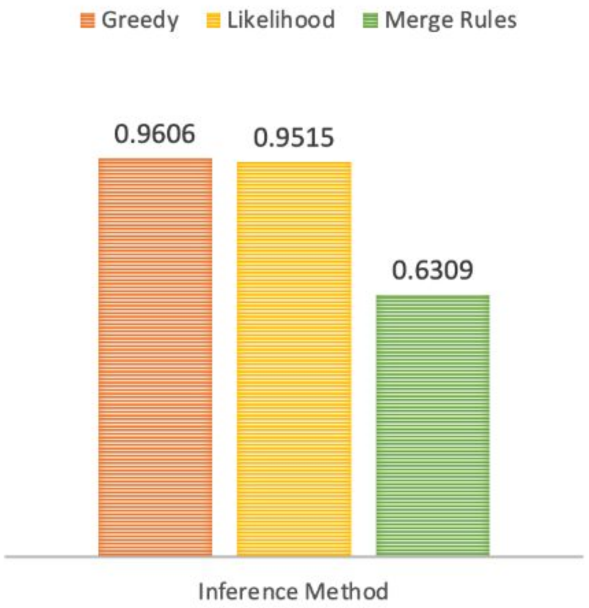
Byte Pair Encoding is Suboptimal for Language Model Pretraining

Kaj Bostrom and Greg Durrett



Train Vocab using One, Infer with Another?

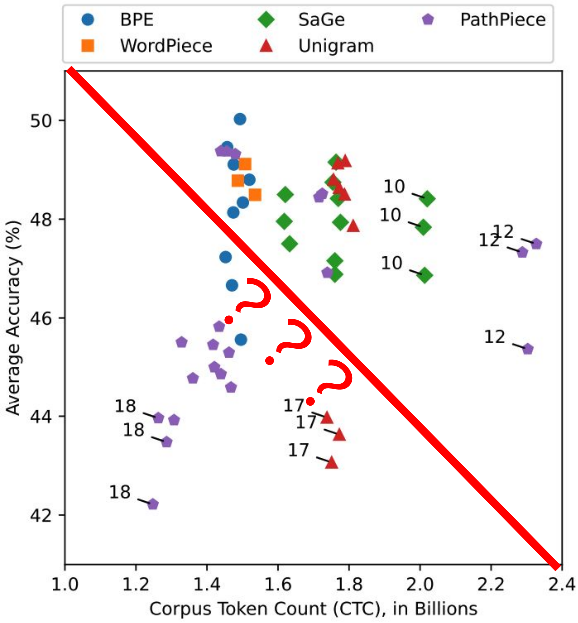
- **Greedy methods** are most aligned to morphology => generally a good choice



Downstream Eval

- **PathPiece**: an algorithm minimizing the total number of tokens in a corpus (**CTC**)
- Evaluation: on `lm-evaluation-harness`
  - 350M params
  - Vocabs: 32k, 40k, 49k

Rank	Vocab Constr	Init Voc	Pre-tok	Segment
1 9 15 16	PathPieceL	BPE Unigram n-gram n-gram	FirstSpace FirstSpace FirstSpDigit FirstSpace	PathPieceL
2 7 17	Unigram		FirstSpace	Likelihood Greedy PathPieceL
3 4 13	BPE		FirstSpace	Merge Greedy PathPieceL
5	WordPiece		FirstSpace	Greedy
6 8 10 11	SaGe	BPE n-gram Unigram n-gram	FirstSpace FirstSpace FirstSpace FirstSpDigit	Greedy
12 14 18	PathPieceR	n-gram	SpaceDigit FirstSpDigit None	PathPieceR
	Random			



Schmidt, Reddy, Zhang, Alameddine, Uzan, **Pinter**, Tanner. *Tokenization Is More Than Compression*. EMNLP 2024

More SaGe!

SaGe 3.0 (work in progress): use Unigram likelihoods in loss; **support likelihood decoding**

Initialize V to large **Unigram**-trained vocabulary

While V is too big:

- Tokenize corpus using V **with likelihood**
  - Every *l* steps: compute embedding table
- Compute **joint ablation loss** for tokens in current set of bottom tokens
- Throw away *k* least lossy tokens

Return V

Talk Overview

- Contextual Models for Tokenizers
- Evaluating Tokenizers Intrinsically
- Decoupling Vocabulary from Inference
- **More Fun with Inference**
- Hebrew Tokenization

Toolkits

- [Huggingface Tokenizers]
  - The most popular
  - Absolutely horrendous
- [SentencePiece]
  - Fast, CLI-based, BPE & Unigram
  - Very hard to extend or debug
- [Tiktoken]
  - For OpenAI's models
  - Only Vanilla BPE (?!?)

Here's a non-exhaustive list of reasons:

1. The HuggingFace `tokenizers` library has horrifically un(der)documented Python interfaces. Some classes even accept arguments that aren't in their signature.
2. The `tokenizers` library is implemented in Rust and hence there is no possibility of inspecting implementations in any Python IDE. Have fun using your black box.
3. The `tokenizers` interface does not allow separating preprocessing from the actual tokenisation algorithm.
   - The `PreTrainedTokenizerBase` class, from which the "slow" (Pythonic) `PreTrainedTokenizer` and "fast" (Rustic) `PreTrainedTokenizerFast` classes both inherit, only declares an end-to-end `.tokenize()` method (equivalent to TkTkT's `.prepareAndTokenise()`). The interface for these subclasses is different enough that both lack features of the other:
     - Whereas `PreTrainedTokenizer` does declare a `._tokenize()` (equivalent to TkTkT's `.tokenise()`), I challenge you to find the equivalent for `PreTrainedTokenizerFast`. Best you'll find is `.backend_tokenizer.model.tokenize()`, which outputs unusable objects of class `tokenizers.Token`.
     - Whereas `PreTrainedTokenizerFast` has fields `.backend_tokenizer.pre_tokenizer` and `.backend_tokenizer.normalizer` (untyped of course, so you can't get autocompletion on their methods unless you manually assign them to a variable and annotate it yourself), `PreTrainedTokenizer` has [no access to a pretokeniser]. Preprocessing has to be defined inside `._tokenize()`, which means you're doing two steps of preprocessing (one inside `.tokenize()` and one inside `._tokenize()`) making this `._tokenize()` no longer equivalent to TkTkT's `.tokenise()`.
     - For `PreTrainedTokenizerFast`, the `.backend_tokenizer.pre tokenizer` and `.backend_tokenizer.normalizer` fields can bo
       TkTkT, meaning you always have to check if the
       can't check if they exist with a simple `if t.bac`
       somehow that's always `False`.
   - Also, the `PreTrainedTokenizerBase` interface is no
     increasing amount of `raise NotImplementedError`
     methods need to be implemented and there's no en
     implemented.
4. The `tokenizers.pre_tokenizers` submodule has technical debt that can't be patched. Some examples:
   - The mapping from Unicode codepoints to UTF-8 bytes, as first used in GPT-2, is only implemented in the `ByteLevel` pretokeniser. Yet, it is concerned with more than this, since it splits on spaces and punctuation (optionally prefixed by a space) before applying the mapping. This is wrong for at least three reasons:
     - Users of the byte mapping don't necessary want the string to be split;
     - It synonymises prefixed spaces (converted to `Ġ`) with start-of-word boundaries whilst actually all words (even those directly preceded by punctuation) should be marked with such a boundary;
     - It assumes that such boundaries should always be at the start of a word.
   - The GPT-2 convention of having a word boundary at the *start* of (almost) all words is hardcoded throughout `transformers` and `tokenizers` (with options that commonly look like `add_prefix_space`) even though the original BPE paper used word boundaries at the *end* of words (</w>). Only supporting the start-of-word convention is bad because this deteriorates downstream performance for e.g. Germanic languages, where a compound has its head at the end and hence it should be allowed to tokenise the head with the exact same tokens as it would be if it was isolated.
   - There is literally a normaliser class called `Precompiled` which is just one big object stored in base64 in the tokeniser config JSON. No access to it in Python, no interface, no description of what it does. A black box. Probably a holdover from adapting the `sentencepiece` package to HuggingFace, yet TkTkT doesn't do it that way.
5. Did you know that their RoBERTa BPE implementation [removes the highest-priority merge from the tokeniser] unless the merge file is preceded by a `#version` tag? This doesn't conform to [the BPE standard], and almost cost me a paper.
6. In the little documentation that does exist (e.g. for WordPiece and KudoPiece), there are so many theoretical inaccuracies that we shouldn't even have confidence in anything that isn't a BPE tokeniser implemented by them. Their [explanation for KudoPiece], an algorithm which itself was already poorly explained originally, is mathematically absurd.
7. They offer very few core models (basically only BPE and KudoPiece, which [`sentencepiece`] already offers and keeps much more updated) whilst there exist many more in the literature, and the likelihood that someone who knows the literature comes along to implement all of them in C++ is rather low.

Please Try

- TkTkT ("Tokenizers toolkit")
- Supports all the tokenizers and methods I described today
- Separates pre-processing, vocab building, and inference

TkTkT

A collection of Pythonic subword tokenisers and text preprocessing tools, with full backwards- *and* forwards-compatibility with HuggingFace `tokenizers`!

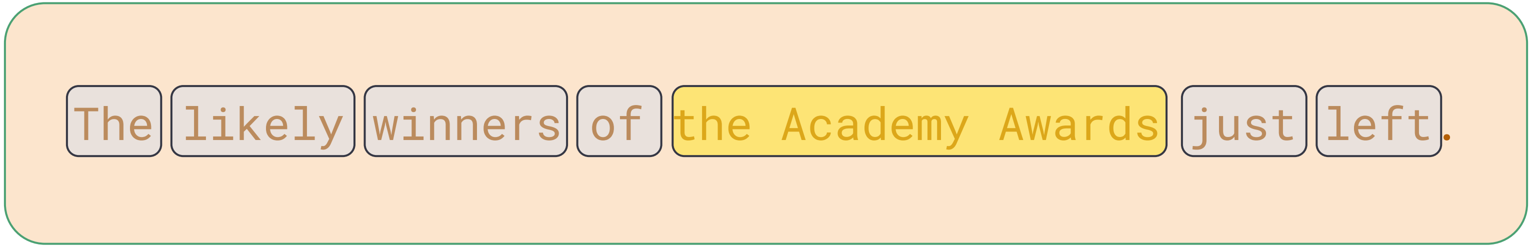

LIFE IN 2013

- *2024: Craig W. Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, Chris Tanner.* ***Tokenization Is More Than Compression.*** *EMNLP. Preprint.*
- *2024: Marco Cognetta, Tatsuya Hiraoka, Naoaki Okazaki, Rico Sennrich, Yuval Pinter.* ***An Analysis of BPE Vocabulary Trimming in Neural Machine Translation.*** *Insights on Negative Results in NLP. Abstract. Preprint.*
- *2024: Omri Uzan, Craig W. Schmidt, Chris Tanner, Yuval Pinter.* ***Greed is All You Need: An Evaluation of Tokenizer Inference Methods.*** 🏆 *Outstanding paper at ACL. PDF. Intrinsic tokenizer benchmark.*
- *2024: Khuyagbaatar Batsuren et al..* ***Evaluating Subword Tokenization: Alien Subword Composition and OOV Generalization Challenge.*** *Preprint.*
- *2024: Anaelia Ovalle et al..* ***Tokenization Matters: Navigating Data-Scarce Tokenization for Gender Inclusive Language Technologies.*** *Findings of NAACL. PDF.*
- *2023: Lisa Beinborn and Yuval Pinter.* ***Analyzing Cognitive Plausibility of Subword Tokenization.*** *EMNLP. PDF. Code.*
- *2023: Shaked Yehezkel and Yuval Pinter.* ***Incorporating Context into Subword Vocabularies.*** *EACL. PDF. Code. Video.*
- *2022: Cassandra L. Jacobs and Yuval Pinter.* ***Lost in Space Marking.*** *Preprint.*
- *2021: Yuval Pinter.* ***Integrating Approaches to Word Representation***. *Preprint. This is an edited version of my dissertation introduction.*
- *2021: Yuval Pinter, Amanda Stent, Mark Dredze, Jacob Eisenstein.* ***Learning to Look Inside: Augmenting Token-Based Encoders with Character-Level Information***. *Preprint.*
- *2020: Yuval Pinter, Cassandra L. Jacobs, Max Bittker.* ***NYTWIT: A Dataset of Novel Words in the New York Times***. *COLING. PDF. Data.*
- *2020: Yuval Pinter, Cassandra L. Jacobs, Jacob Eisenstein.* ***Will it Unblend?***. *Findings of EMNLP. PDF. Video (lay audience). Handout (linguist audience). Also presented at SCiL 2021.*
- *2019: Nicolas Garneau, Jean-Samuel Leboeuf, Yuval Pinter, Luc Lamontagne.* ***Attending Form and Context to Generate Specialized Out-of-Vocabulary Words Representations***. *Preprint.*
- *2019: Yuval Pinter, Marc Marone, Jacob Eisenstein.* ***Character Eyes: Seeing Language through Character-Level Taggers***. *Blackbox NLP Workshop. PDF. Slides. Code. In June 2019 I gave a talk about this project at CUNY, as well as a (different) talk in December 2019 - February 2020 at Amazon Research, at the Tel Aviv University Machine Learning Seminar, and at AISC (video). Slides from the academic venues available upon request.*
- *2017: Yuval Pinter, Robert Guthrie, Jacob Eisenstein.* ***Mimicking Word Embeddings using Subword RNNs***. *Proceedings of EMNLP. PDF. Blog post. Talk. Slides. Code.*

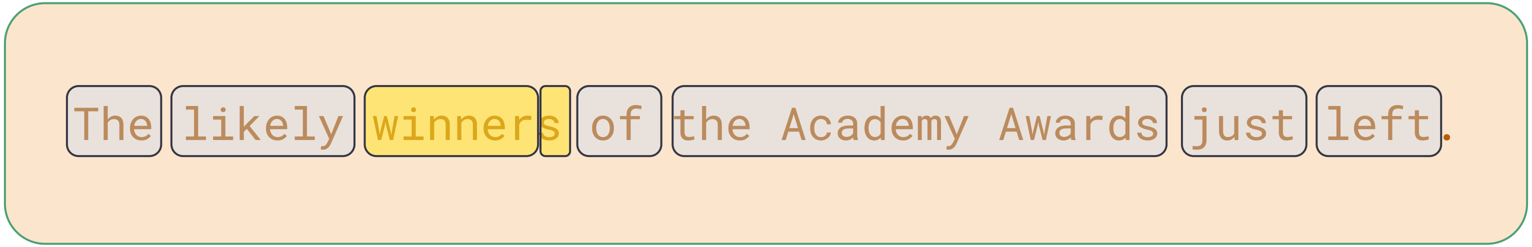
Talk Overview

- Contextual Models for Tokenizers
  - V SaGe
- Evaluating Tokenizers Intrinsically
  - V (Beinborn)
  - V (Greed p1)
  - (Huygaa?)
- Decoupling Vocabulary from Inference
  - V (Greed p2)
  - V (Pathpiece)
  - V (Cognetta) [and list the other ones]
- More Fun with Inference

Representing Language

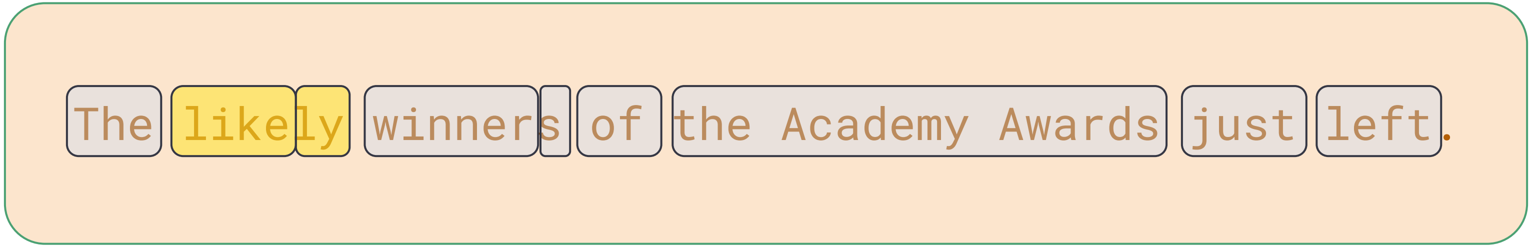


Representing Language



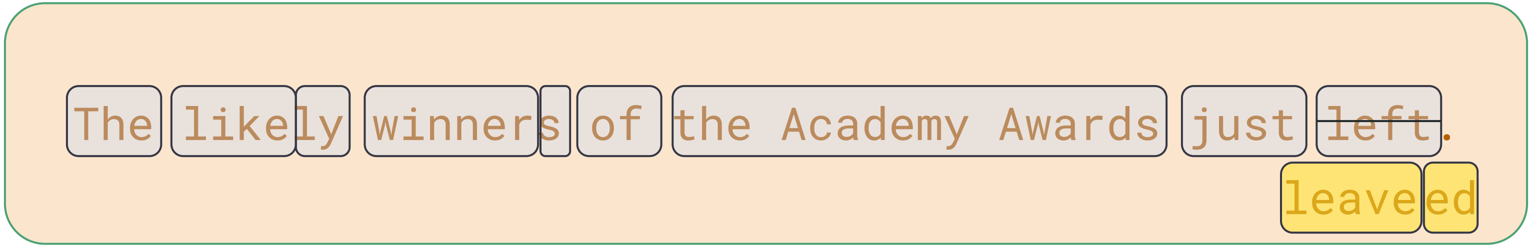
Subword modeling

Representing Language



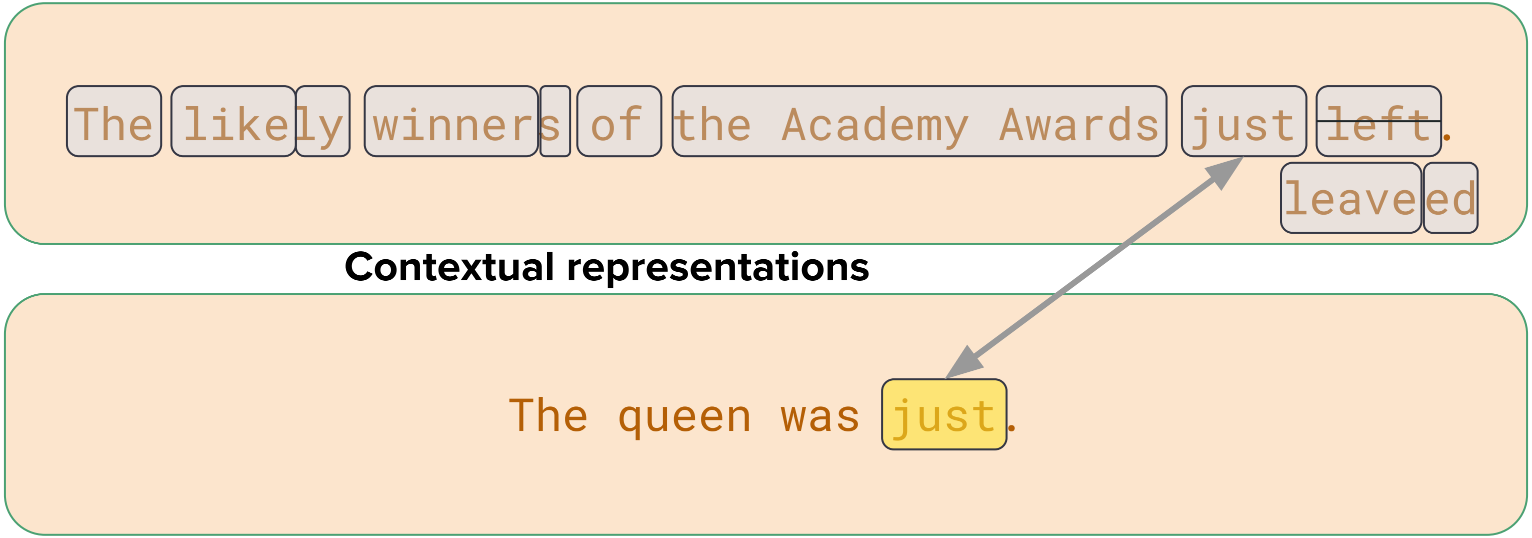
Subword modeling

Representing Language

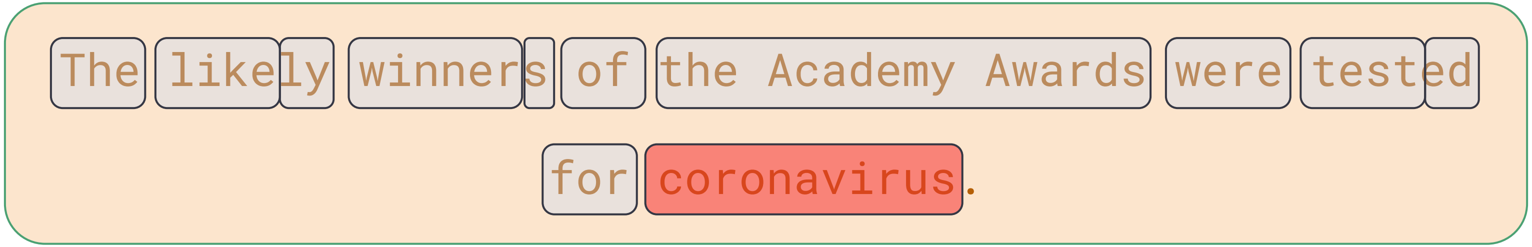


Subword modeling

Representing Language



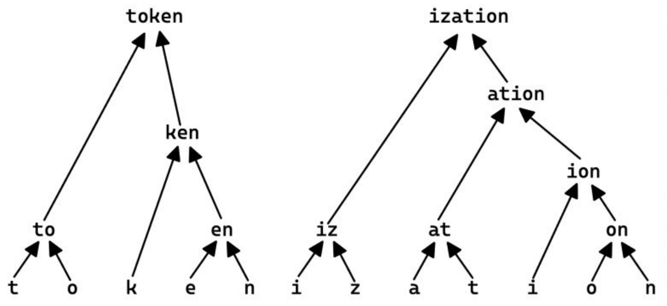
Representing Language



Out-of-vocabulary terms (OOV)

But This is Mostly about **BPE** (Byte-Pair Encoding)

- BPE is **one** algorithm for tokenization, it's bottom up: merge the most frequent token-pair into a new token, iteratively



- **WordPiece** is mostly the same

- **UnigramLM** is top-down, retaining tokens that have a high "likelihood" across the training corpus
  - When a model is claimed to use "**sentencepiece** tokenization", it usually means this

- This 2020 paper argues UnigramLM is better than BPE:

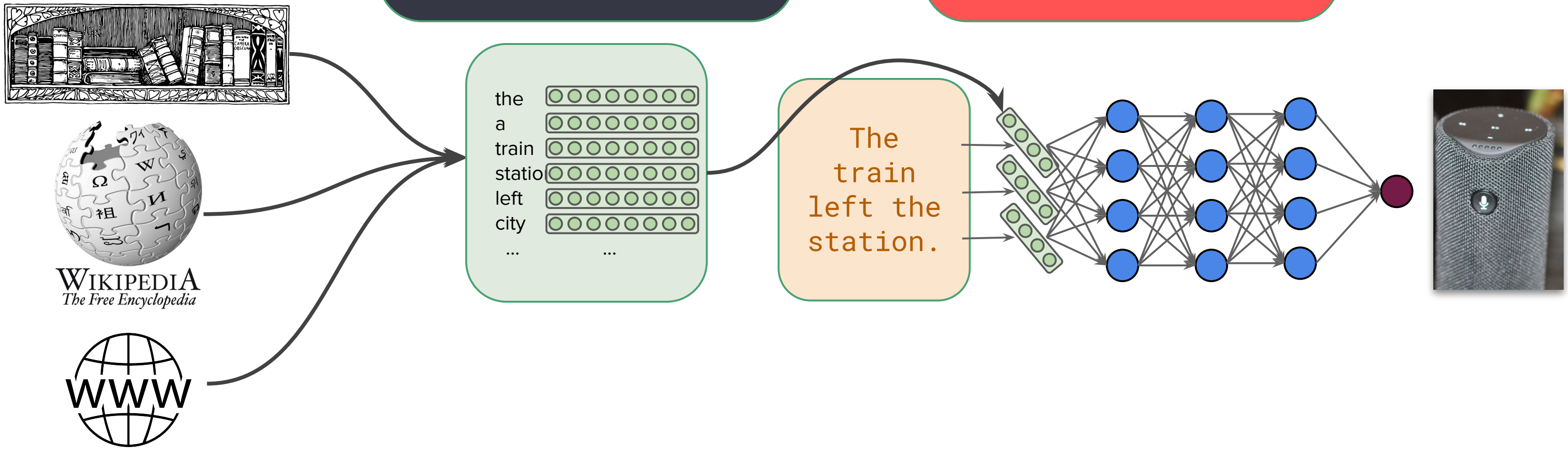
Byte Pair Encoding is Suboptimal for Language Model Pretraining

Kaj Bostrom and **Greg Durrett**

The Static Embeddings Pipeline

Pre-train a model over a large corpus (“unlabeled”)

Use model artifacts in **downstream task** (“labeled”)

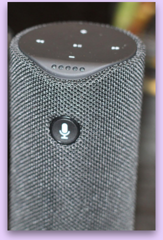


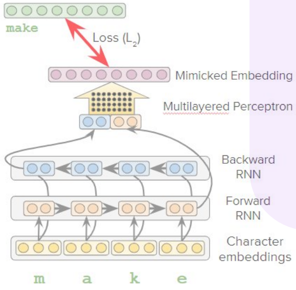
Integrating the Character Level into Subwords

Learn subword vocabulary

Learn contextual embedding model from a large corpus

Fine-tune model parameters on downstream task





Add character-level sequence model

More learning, with both tokenization strategies

Talk Overview

- Contextual Models for Tokenizers
- Evaluating Tokenizers Intrinsically
- Decoupling Vocabulary from Inference
- More Fun with Inference
- **Hebrew Tokenization**

Everything's Worse in ~*~Hebrew~*~