Building Blocks of Modern LLMs 2: Pretraining Data

11-667

Chenyan Xiong

Carnegie Mellon University

Fall 2023

Pretraining Data: Outline

Preprocessing clean texts

- Simple Tokenization
- Subword Tokenization
- Batching

Scaling up pretraining data using the Web

- Obtaining Web Pages
- Scrape Texts
- Clean Texts

Preprocessing Clean Texts

Many scenarios work on relatively clean texts

- Wikipedia, BookCorpus, and classic NLP applications
- High-resource languages: English, French, etc.

A perfect world situation: Texts are clean and well-formatted

- (Unlikely to achieve in real-world systems)

Preprocessing Clean Texts

Many scenarios work on relatively clean texts

- Wikipedia, BookCorpus, and classic NLP applications
- High-resource languages: English, French, etc.

A perfect world situation: Texts are clean and well-formatted

- (Unlikely to achieve in real-world systems)

Preprocessing is to convert them into batches of training data

The Steelers enjoy a large, widespread fanbase nicknamed Steeler Nation. They currently play their home games at Acrisure Stadium.

Raw Clean Text

Preprocessing Clean Texts

Many scenarios work on relatively clean texts

- Wikipedia, BookCorpus, and classic NLP applications
- High-resource languages: English, French, etc.

A perfect world situation: Texts are clean and well-formatted

- (Unlikely to achieve in real-world systems)

Preprocessing is to convert them into batches of training data

The Steelers enjoy a large, widespread fanbase nicknamed Steeler Nation. They currently play their home games at Acrisure Stadium.

Raw Clean Text

Tokenization →

'▁The', '▁Steel', 'ers', '▁enjoy', '▁a', '▁large', ',', '▁wide', 'spre', 'ad', '▁fan', 'base', '▁nick', 'na', 'med', '▁Steel', 'er', '▁Nation', '.', '▁They', '▁currently', '▁play', '▁their', '▁home', '▁games', '▁at', '▁A', 'cris', 'ure', '▁Stadium', '.'

Tokenized

Preprocessing Clean Texts

Many scenarios work on relatively clean texts

- Wikipedia, BookCorpus, and classic NLP applications
- High-resource languages: English, French, etc.

A perfect world situation: Texts are clean and well-formatted

- (Unlikely to achieve in real-world systems)

Preprocessing is to convert them into batches of training data

The Steelers enjoy a large, widespread fanbase nicknamed Steeler Nation. They currently play their home games at Acrisure Stadium.

Raw Clean Text

→ **Tokenization** →

'▁The', '▁Steel', 'ers', '▁enjoy', '▁a', '▁large', ',', '▁wide', 'spre', 'ad', '▁fan', 'base', '▁nick', 'na', 'med', '▁Steel', 'er', '▁Nation', '.', '▁They', '▁currently', '▁play', '▁their', '▁home', '▁games', '▁at', '▁A', 'cris', 'ure', '▁Stadium', '.'

Tokenized

→ **Batching** →

[580, 109027, 1313, 25224, 9, 21333, 3, 38133, 21328, 711, 1206, 37381, 128910, 75, 4805, 109027, 55, 82580, 4, 0]
[10659, 82423, 11300, 2362, 5367, 27527, 98, 61, 58531, 3407, 88259, 4,0,0,0,0,0,0,0,0]

Tensor

Tokenization

A simple way: Split by spaces + rules to handle punctuations and special cases

“They currently play their home games at Acrisure Stadium.” → “They” “currently” “play” “their” “home” “games” “at” “Acrisure” “Stadium” “.”

Tokenization

A simple way: Split by spaces + rules to handle punctuations and special cases

“They currently play their home games at Acrisure Stadium.” → “They” “currently” “play” “their” “home” “games” “at” “Acrisure” “Stadium” “.”

Challenges:

- Rules differ language to language
  - E.g. in English “don’t”, “cannot”, “pre-training”
  - There are always edge cases
  - Some languages do not use space to separate words

Tokenization

A simple way: Split by spaces + rules to handle punctuations and specific cases

"They currently play their home games at Acrisure Stadium." → "They" "currently" "play" "their" "home" "games" "at" "Acrisure" "Stadium" "."

Challenges:

- Vocabulary explosion
  - Large vocabulary creates instability issue
  - Little signals for rare words in the long tail
    - Many of them are important, such as named entities ("Acrisure")

Tokenization

A simple way: Split by spaces + rules to handle punctuations and specific cases

“They currently play their home games at Acrisure Stadium.” → “They” “currently” “play” “their” “home” “games” “at” “Acrisure” “Stadium” “.”

Challenges:

- Open vocabulary problem
  - Many words may never appear in training data. They become [UNK].
  - More severe in some languages
    - Low resource language
    - Language that have a large vocabulary, e.g., those that concatenate words

Subword Tokenization

Tokenize sequences into sub-words

"They currently play their home games at Acrisure Stadium." → '▁They', '▁currently', '▁play', '▁their', '▁home', '▁games', '▁at', '▁A', 'cris', 'ure', '▁Stadium', '.'

- A dynamic tokenization:
  - Frequent words kept as whole
  - Rare words split into sub-words

[9] Sennrich et al. "Neural Machine Translation of Rare Words with Subword Units." ACL. 2016.

Subword Tokenization: Byte Pair Encoding (BPE)

Byte Pair Encoding: Construct subword vocabulary by learning to merge characters

- Inspiration from compression algorithms

Training Steps:

1. Start from single character vocabulary
   - E.g., in English, alphabets + punctuations
2. Merge the most frequent subword pair
   - Vocabulary size +1
3. Re-tokenize the corpus with the merged subword pair
   - Merge all appearances of that subword pair
4. Repeat step 2-3 till reached target vocabulary size

[9] Sennrich et al. "Neural Machine Translation of Rare Words with Subword Units." ACL. 2016.

Subword Tokenization: Byte Pair Encoding (BPE)

An efficient learning implementation of BPE

1. Start from a pre-tokenized texts, with sequence split to words, e.g., by rules
2. Construct a word dictionary with frequency counts

("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

3. Start from uni-character vocabulary, merge pairs by frequency, till reached target vocabulary size

Example from https://huggingface.co/docs/transformers/tokenizer_summary

Subword Tokenization: Byte Pair Encoding (BPE)

An efficient learning implementation of BPE

1. Start from a pre-tokenized texts, with sequence split to words, e.g., by rules
2. Construct a word dictionary with frequency counts

   ("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

3. Start from uni-character vocabulary, merge pairs by frequency, till reached target vocabulary size

Vocabulary	**Tokenization**
(“h”, “u”, “g”, “p”, “n”, “b”, “s”)	("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)

Subword Tokenization: Byte Pair Encoding (BPE)

An efficient learning implementation of BPE

1. Start from a pre-tokenized texts, with sequence split to words, e.g., by rules
2. Construct a word dictionary with frequency counts

("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

3. Start from uni-character vocabulary, merge pairs by frequency, till reached target vocabulary size

Vocabulary	Tokenization
(“h”, “u”, “g”, “p”, “n”, “b”, “s”)	("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)
(“h”, “u”, “ug”, “p”, “n”, “b”, “s”)	("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "ug" "s", 5)

Subword Tokenization: Byte Pair Encoding (BPE)

An efficient learning implementation of BPE

1. Start from a pre-tokenized texts, with sequence split to words, e.g., by rules
2. Construct a word dictionary with frequency counts

("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

3. Start from uni-character vocabulary, merge pairs by frequency, till reached target vocabulary size

Vocabulary	Tokenization
(“h”, “u”, “g”, “p”, “n”, “b”, “s”)	("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)
(“h”, “u”, “ug”, “p”, “n”, “b”, “s”)	("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "ug" "s", 5)
(“h”, “u”, “ug”, “p”, “un”, “b”, “s”)	("h" "ug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("h" "ug" "s", 5)

Subword Tokenization: Byte Pair Encoding (BPE)

An efficient learning implementation of BPE

1. Start from a pre-tokenized texts, with sequence split to words, e.g., by rules
2. Construct a word dictionary with frequency counts

("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

3. Start from uni-character vocabulary, merge pairs by frequency, till reached target vocabulary size

Vocabulary	Tokenization
(“h”, “u”, “g”, “p”, “n”, “b”, “s”)	("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)
(“h”, “u”, “ug”, “p”, “n”, “b”, “s”)	("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "ug" "s", 5)
(“h”, “u”, “ug”, “p”, “un”, “b”, “s”)	("h" "ug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("h" "ug" "s", 5)
(“hug”, “p”, “ug”, “un”, “b”, “h”, “s”)	("hug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("h" "ug" "s", 5)

Subword Tokenization: Byte Pair Encoding (BPE)

Byte Pair Encoding: Inference

1. Start from single character tokenization
2. Applied learned merge operations till non merges possible
   1. In the order of their learned sequence

      (“u” + “g”) → (“u” + “n”) → (“h” + “ug”)

   2. A deterministic operation, often performed at sequence/sentence level

[9] Sennrich et al. "Neural Machine Translation of Rare Words with Subword Units." ACL. 2016.

Subword Tokenization: Pros

Controlled vocabulary size

- A pre-defined hyperparameter as a design choice

Learned vocabulary, best of two-worlds

- Frequent words kept whole
- Tail words split to sub-words
    - More observations on sub-words
    - Utilization of morphology information

Subword Tokenization: Pros

Controlled vocabulary size

- A pre-defined hyperparameter as a design choice

Learned vocabulary, best of two-worlds

- Frequent words kept whole
- Tail words split to sub-words
  - More observations on sub-words
  - Utilization of morphology information

Sub/raw units suit well with neural methods

- Trivial for Transformers to figure out common combinations
- Neural representations smooth rare combinations

Subword Tokenization: Further Upgrades

Unknown (sub)tokens still exist from unknow characters

→ Byte-level BPE: Use raw bytes (e.g., Unicode bytes) as the character sets [GPT-2]

[10] Kudo and Richardson. "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing." ACL. 2018.

Subword Tokenization: Further Upgrades

Unknown (sub)tokens still exist from unknow characters

→ Byte-level BPE: Use raw bytes (e.g., Unicode bytes) as the character sets [GPT-2]

Require pre-tokenization to words

→ SentencePiece: Treat space as a special character and learn subword with it at sentence level [10]

[10] Kudo and Richardson. "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing." ACL. 2018.

Subword Tokenization: Further Upgrades

Unknown (sub)tokens still exist from unknow characters

→ Byte-level BPE: Use raw bytes (e.g., Unicode bytes) as the character sets [GPT-2]

Require pre-tokenization to words

→ SentencePiece: Treat space as a special character and learn subword with it at sentence level [10]

Inferencing via merge operation is $\mathrm{O}(n^2)$

- Normally performed at sentence level thus n is sentence length
- Not a bottleneck in LLM pipelines
- Many ways to improve it, algorithm-wise or implementation-wise

[10] Kudo and Richardson. "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing." ACL. 2018.

Questions?

Pretraining Data: Outline

Preprocessing clean texts

- Simple Tokenization
- Subword Tokenization
- **Batching**

Scaling up pretraining data using the Web

- Obtaining Web Pages
- Scrape Texts
- Clean Texts

Preprocessing Clean Texts: Batching

Task: put tokenized text sequences into training batches

'▁The', '▁Steel', 'ers', '▁enjoy', '▁a', '▁large', ',', '▁wide', 'spre', 'ad', '▁fan', 'base', '▁nick', 'na', 'med', '▁Steel', 'er', '▁Nation', '.', '▁They', '▁currently', '▁play', '▁their', '▁home', '▁games', '▁at', '▁A', 'cris', 'ure', '▁Stadium', '.'

Tokenized

→ **Batching**

[580, 109027, 1313, 25224, 9, 21333, 3, 38133, 21328, 711, 1206, 37381, 128910, 75, 4805, 109027, 55, 82580, 4, 0]
[10659, 82423, 11300, 2362, 5367, 27527, 98, 61, 58531, 3407, 88259, 4,0,0,0,0,0,0,0,0]

Tensor Batch

1. Map subword to token ids
2. Group sequences into batches

Preprocessing Clean Texts: Batching

Task: put tokenized text sequences into training batches

'▁The', '▁Steel', 'ers', '▁enjoy', '▁a', '▁large', ',', '▁wide', 'spre', 'ad', '▁fan', 'base', '▁nick', 'na', 'med', '▁Steel', 'er', '▁Nation', '.', '▁They', '▁currently', '▁play', '▁their', '▁home', '▁games', '▁at', '▁A', 'cris', 'ure', '▁Stadium', '.'

Tokenized

Batching →

[580, 109027, 1313, 25224, 9, 21333, 3, 38133, 21328, 711, 1206, 37381, 128910, 75, 4805, 109027, 55, 82580, 4, 0]
[10659, 82423, 11300, 2362, 5367, 27527, 98, 61, 58531, 3407, 88259, 4,0,0,0,0,0,0,0,0]

Tensor Batch

Some notable design choices:

- What is a sequence?
  - A sentence? A paragraph?
  - Breaking at paragraph boundary or document boundary?

Preprocessing Clean Texts: Batching

Task: put tokenized text sequences into training batches

Tokenized		Tensor Batch
'▁The', '▁Steel', 'ers', '▁enjoy', '▁a', '▁large', ',', '▁wide', 'spre', 'ad', '▁fan', 'base', '▁nick', 'na', 'med', '▁Steel', 'er', '▁Nation', '.', '▁They', '▁currently', '▁play', '▁their', '▁home', '▁games', '▁at', '▁A', 'cris', 'ure', '▁Stadium', '.'	**Batching** →	[580, 109027, 1313, 25224, 9, 21333, 3, 38133, 21328, 711, 1206, 37381, 128910, 75, 4805, 109027, 55, 82580, 4, 0] [10659, 82423, 11300, 2362, 5367, 27527, 98, 61, 58531, 3407, 88259, 4,0,0,0,0,0,0,0,0]

Some notable design choices:

- How to group sequence in a batch?
  - Straightforward but slow: static # of rows per batch, one sequence per row, padding to max length
  - Speed up but non-uniform: group sequence by length, pad to max sequence length, till reached target token number
  - More customized grouping in tensors may require customized support from the framework

Questions?

Pretraining Data: Outline

Preprocessing clean texts

- Simple Tokenization
- Subword Tokenization
- Batching

Scaling up pretraining data using the Web

- Obtaining Web Pages
- Scrape Texts
- Clean Texts

Scaling up Pretraining Data

Benefits of pretraining with more data

Table 11: $\mathbf{RoBERTa_{Large}}$ Pretrained with Different Data Setups[10].

Pretrain Setups	Data Size	Batch Size	Steps	MNLI-m (ACC)	SQuAD 1.1 (F1)
Wiki + Books	16GB	8K	100K	89.0	93.6
+Additional Data	160GB	8K	100K	89.3	94.0
+Pretrain More Steps	160GB	8K	300K	90.0	94.4
+Even More Steps	160GB	8K	500K	90.2	94.6

- Same model, same pretraining steps, more data help
- With more data more pretraining steps also help
- Empirical gains more than many “fancier” improvements

[10] Li, et al. "RoBERTa: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692. 2019.

Scaling up Pretraining Data

Where to get more pretraining data?

- High quality clean text corpora, such as Wikipedia, news, scientific papers, eventually run out
  - They also have issues, such as copyrights

Scaling up Pretraining Data

Where to get more pretraining data?

- High quality clean text corpora, such as Wikipedia, news, scientific papers, eventually run out
  - They also have issues, such as copyrights

Web quickly become the only viable option

- The go-to place of general domain human knowledge, in this digital era
- Massive, unlikely grow slower than computing resources
- Publicly available
  - Though with copyright and many other constraints
- Noisy, dirty, and biased

Pretraining Data Curation from the Web

Goal: Obtain high quality, large-scale pretraining data from the public web

Seed URLs $\xrightarrow{\text{Explore}}$ **Target URLs** $\xrightarrow{\text{Crawl}}$ **HTML Contents** $\xrightarrow{\text{Scrape}}$ **Raw Texts** $\xrightarrow{\text{Clean}}$ **Clean Texts**

General Challenges:

- What and which URLs to crawl
- How to extract texts from HTML
- How to select clean texts for pretraining

Pretraining Data Curation from the Web: Obtaining URLs

Seed URLs $\xrightarrow{\text{Explore}}$ **Target URLs** $\xrightarrow{\text{Crawl}}$ **HTML Contents** $\xrightarrow{\text{Scrape}}$ **Raw Texts** $\xrightarrow{\text{Clean}}$ **Clean Texts**

Main questions:

- How to harvest a large number of URLs efficiently
- How to select “high-quality” URLs
  - What makes an URL good?
- How to remove “bad” URLs?
  - Some cases are clear cut: spammy, unsafe, etc.
  - Some are hard to detect or up to debate: certain biases.

WebText Corpus

WebText: The pretraining corpus of GPT-2

- Harvest all outbound links from Reddit
    - Manually mentioned by humans
- Only keep links received >= 3 “Karma”
    - A heuristic to filter good URLs
    - Total 45 million URLs deduped to 8 million web pages

[10] Radford, Alec, et al. "Language models are unsupervised multitask learners.“ OpenAI Blog. 2019

WebText Corpus

WebText: The pretraining corpus of GPT-2

- Harvest all outbound links from Reddit
  - Manually mentioned by humans
- Only keep links received >= 3 “Karma”
  - A heuristic to filter good URLs
  - Total 45 million URLs deduped to 8 million web pages

How to harvest URLs:

- From Reddit mentions

Definition of good URLs:

- Other Reddit users like it

[10] Radford, Alec, et al. "Language models are unsupervised multitask learners.“ OpenAI Blog. 2019

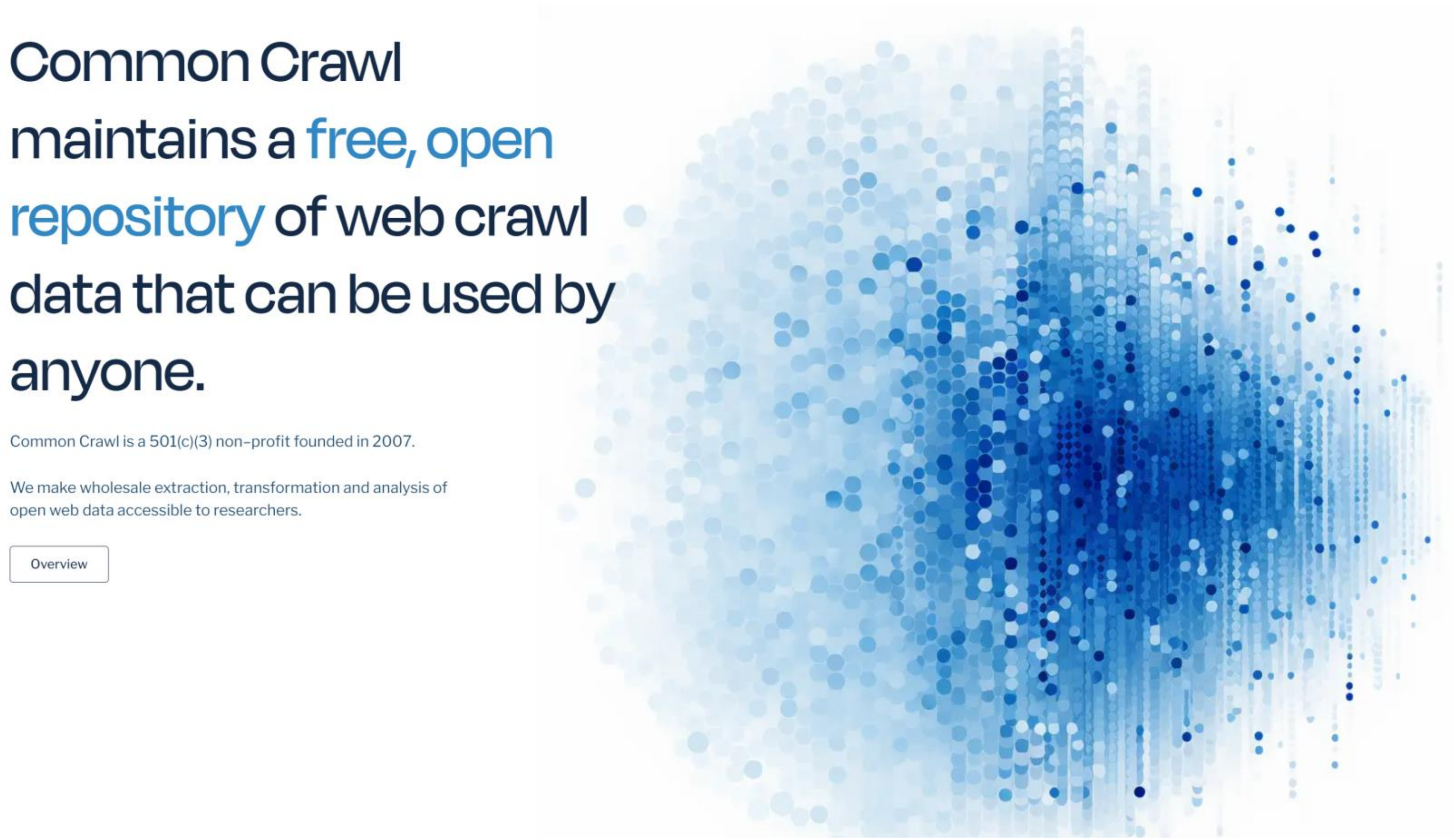
WebText Corpus

WebText:

- Pros
  - An easy to harvest a relatively large set of URLs from a common resource
  - Human votes on the URLs
- Bad
  - Reddit starts to forbid the use of its data for pretraining LLMs
  - Limited Scale
  - Reddit is not the cleanest part of the internet

[10] Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI Blog. 2019

Common Crawl



Common Crawl: commoncrawl.org

- Petabytes of web pages
- Provide open access to large scale web crawls
  - Previously a privilege of search engine companies
- Monthly crawls and dumps
  - Re-crawled web pages and fresh dumps (bi)monthly
  - Recent dumps are ~3 billion pages
  - Date back to past 10 years
    - “220 billion web pages (HTML) captured 2008 – 2021” [10]

[11] Sebastian Nagel. "From Web Graphs to Prioritizing Web Crawls. " OSSYM 2021

Common Crawl: Web Exploration

Common Crawl Web Exploration:

- Start from a set of seed URLs: popular, high-quality, and trustworthy websites
    - Gov, Edu, etc.
    - Top web domains
- Traverse the web to obtain a candidate set of URLs
    - Around 500 billion links discovered per month crawl
    - About 25+ billion unique ones
- Prioritize a subset (~3 billion) of URLs to crawl and include in the dump

[11] Sebastian Nagel. "From Web Graphs to Prioritizing Web Crawls. " OSSYM 2021

Common Crawl: Prioritization

Why?

- No one can crawl the entire web
- Discovered URL >>> Crawling budget
- Steer crawler to the better part of the web

[11] Sebastian Nagel. "From Web Graphs to Prioritizing Web Crawls. " OSSYM 2021

Common Crawl: Prioritization

Which URLs to crawl? [11]

- 2008-2012: CC's in-house Page Rank
- 2012-2015: Added ranking and metadata of 22 billion pages donated from web search engine blekko
- 2016-2018: Occasional seed URL donations, ~400 million URLs
- 2016: Alexa and Common Search Rankings
- 2017-Now: CC's in-house web graph based rankings (page rank and centrality)
  - Importance score calculated based on past three-month dumps
  - Steer the crawler for next three month
  - Capped # of URLs per domain

[11] Sebastian Nagel. "From Web Graphs to Prioritizing Web Crawls. " OSSYM 2021

Common Crawl

Top Domains in Common Crawl:

- July 2022 Dump
- Successful Crawls
- Ordered by # of web pages

Favors:

- Sites with more sub-domains
- .edu and .gov
- EU domains

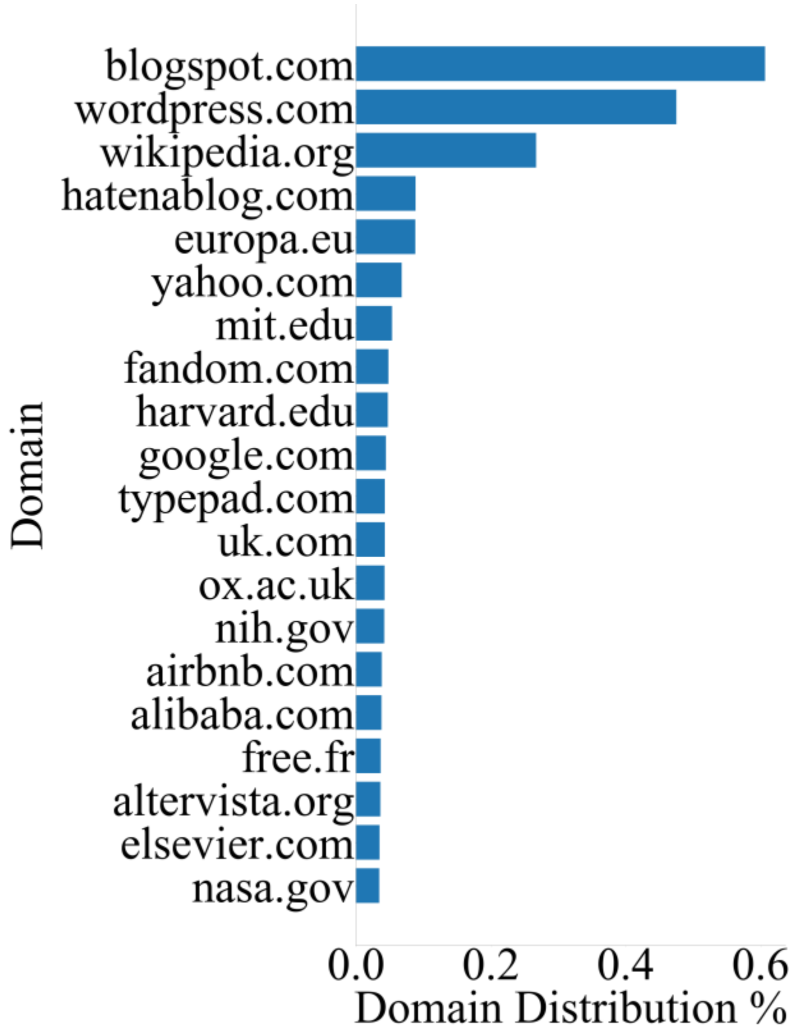


Figure 6: Top 20 domains in Common Crawl July 2022.

Common Crawl: Pros

Likely the one and the only public web crawl at this scale

- Still far away from commercial search engines, but the closest we have

10 years of crawl dumps enabled various data subsamples

- News, medical, etc.

Accumulation of low resource languages

- One can combine low resources languages from years of dumps to pair with English texts from one month

Common Crawl: Cons

Each crawl is ~3billion, still limited coverage of the massive web

- Crawl every month restart from the seed URLs

URL distributions skewed by

- Crawling prioritization
- Per domain URL cap
- Physical location of the crawling machines (and people)

Important Concerns:

- Public web != copyright free web
- Web without filters: noisy, biased, and dirty

ClueWeb22

ClueWeb web corpus series:

- The web corpus constructed and maintained by CMU & Co. for research usage
- ClueWeb09 and ClueWeb12: Each has ~1 billion web pages crawled at CMU in 2009 and 2012
    - Was nearly half of CMU's network traffic at certain time points

[12] Overwijk et al. "ClueWeb22: 10 billion web documents with rich information." SIGIR. 2022.

ClueWeb22

ClueWeb22 is a collaboration between Microsoft and CMU:

- 10 billion web pages
- Sampled from the super head, head, and torso of Bing's search index

Table 12: Statistics of ClueWeb22 categories.

Category	**#Pages**	**#Tokens**	**Sampling Distribution**
ClueWeb22-B	200M	696B	From Most Popular Web Pages ("Super Head")
ClueWeb22-A	2B	6.1T	From Pages also Frequently Visited by Users ("Head")
ClueWeb22-L	10B	16.7T	Mixed Head-Tail Pages ("Head and Tail")

ClueWeb22: Web URL Prioritization

Sampled using predicted web URL importance from Bing's search index

- Prediction Target: the likelihood of a user click on the URL, regardless of which query
- Model Features:
  - web graph connectivity, URL domain, document content, and page structure, etc.
  - Typical information effective in web search

ClueWeb22: Web URL Prioritization

Sampled using predicted web URL importance from Bing's search index

- Prediction Target: the likelihood of a user click on the URL, regardless of which query
- Model Features:
  - web graph connectivity, URL domain, document content, and page structure, etc.
  - Typical information effective in web search
- An effective separation of different types of web pages

Table 13: Example of Wiki Pages in ClueWeb22 categories.

ClueWeb22-B	**Page Type**
https://en.wikipedia.org/wiki/Super_Bowl_XXXVIII	Entity
https://en.wikipedia.org/wiki/Chevrolet_Corvette_(C8)	Entity
https://en.wikipedia.org/wiki/Discourse	Entity
ClueWeb22-A	**Page Type**
https://en.wikipedia.org/wiki/1897_in_film	List
https://en.wikipedia.org/wiki/2019_FFA_Cup_Final	Entity
https://en.wikipedia.org/wiki/Category:Public_administration_schools_in_the_United_States	Category
ClueWeb22-L	**Page Type**
https://en.wikipedia.org/wiki/Template%3ANeighbourhoods_in_Kolkata	Edit Template
https://en.wikipedia.org/wiki/Category:Sports_leagues_established_in_1990	Category
https://en.wikipedia.org/wiki/Talk:The_Shoes_of_the_Fisherman	Wiki Project

[12] Overwijk et al. "ClueWeb22: 10 billion web documents with rich information." SIGIR. 2022.

ClueWeb22: Web URL Distribution

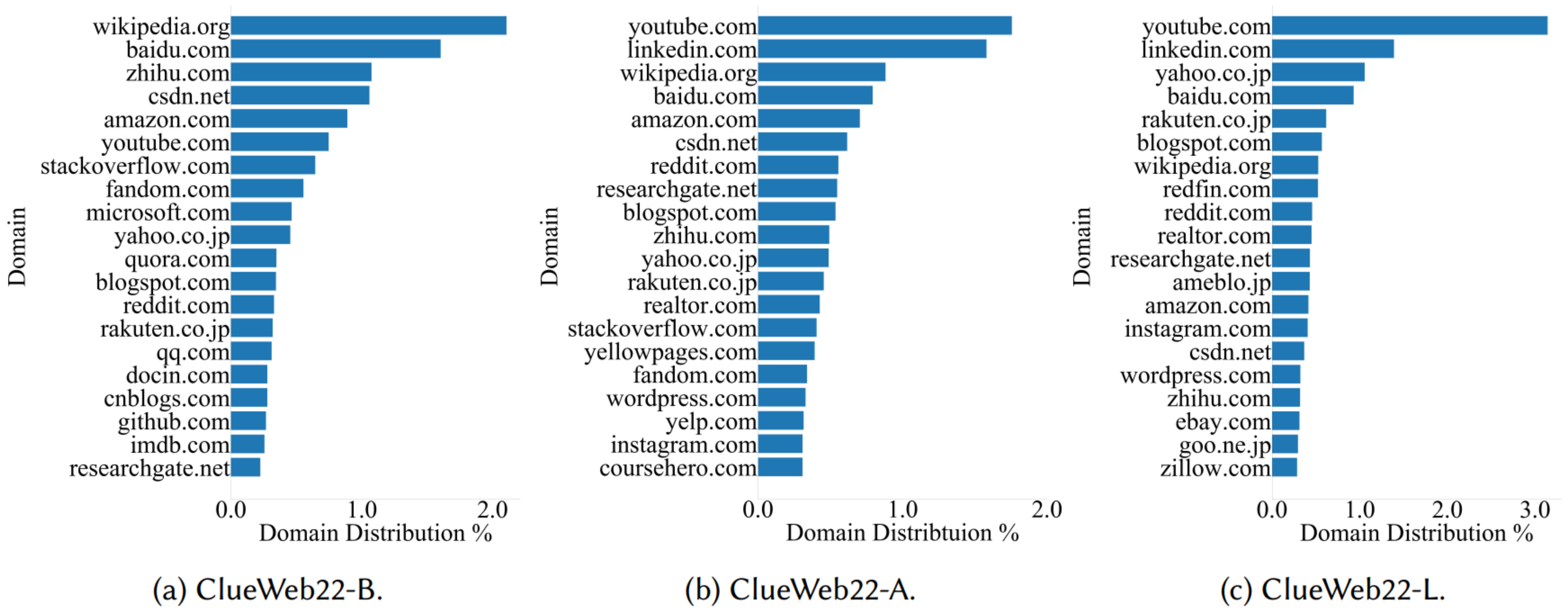


Figure 7: Top 20 domains in ClueWeb22 Categories

Aligned with web search users' preference

- With a twist from the specific search engine's perspective

ClueWeb22: Pros

Web pages sampled from a commercial search engine index

- Better discovery of the internet

Relatively clean URLs

- Went through commercial search filters (adult and spam filter)
- Sampled based on predicted web search traffic

Relatively large scale

- 10 billion web pages

Rich information beside plain HTML

ClueWeb22: Cons

Likely a one time effort

- Hard to bet on the openness of big corporate
- Getting old every day

Sampled from one search engine’s point of view

- Biased towards the search engine
- Unclear web graph connectivity

Research only license for research organizations

- A limitation for some users

Web Corpus: Summary

Table 14: Summary of Web Corpus Sources.

	WebText	Common Crawl	ClueWeb22
URL Sources	Reddit Outlinks	CC's web crawl	Bing crawl
URL Selection	Reddit User Vote	Web graph based importance	Search user click prediction
Scale	8 Million Pages	3 Billion (bi)month for 10 years	10 Billion web pages
Distribution	N.A.	AWS Download	CMU Shipped Disks
License	N.A.	"At your own risk"	End-user license per organization
Copyright Control	N.A.	"At your own risk"	Research only + Honor content deletion request

The construction of web corpus introduce various priors to the end pretraining data

Scale and quality is often a trade-off

Copyright w.r.t. language model pretraining is an active ethical and legal topic

Pretraining Data: Outline

Preprocessing clean texts

- Simple Tokenization
- Subword Tokenization
- Batching

Scaling up pretraining data using the Web

- Obtaining Web Pages
- **Scrape Texts**
- Clean Texts

Pretraining Data Curation from the Web: Scrape Texts

Seed URLs $\xrightarrow{\text{Explore}}$ Target URLs $\xrightarrow{\text{Crawl}}$ HTML Contents $\xrightarrow{\text{Scrape}}$ Raw Texts $\xrightarrow{\text{Clean}}$ Clean Texts

From

```html
<header class="post-header"> <h1 class="post-title"> Chenyan Xiong
</h1> <p class="desc">Associate Professor, Language Technologies
Institute, Carnegie Mellon University.</p> </header> <article> <div
class="profile float-left"> <figure> <picture> <source
class="responsive-img-srcset" media="(max-width: 480px)"
srcset="/~cx/assets/img/prof_pic-480.webp"></source> <source
class="responsive-img-srcset" media="(max-width: 800px)"
srcset="/~cx/assets/img/prof_pic-800.webp"></source> <source
class="responsive-img-srcset" media="(max-width: 1400px)"
srcset="/~cx/assets/img/prof_pic-1400.webp"></source> <img
src="/~cx/assets/img/prof_pic.jpg?ba2865f2a3edfc035418bf286dfb5f61"
class="img-fluid z-depth-1 rounded" width="auto" height="auto"
alt="prof_pic.jpg" onerror="this.onerror=null; $('.responsive-img-
srcset').remove();"> </picture> </figure> <div class="address">
<p>6409 GHC,</p> <p>5000 Forbes Avenue</p> <p>Pittsburgh, PA 15213</p>
</div> </div> <div class="clearfix"> <p>I am an Associate Professor at
Language Technologies Institute (LTI), Carnegie Mellon University
(CMU). My research area is the intersection of information retrieval,
natural language processing, and deep learning. Previously I worked at
Microsoft Research till 2023 Fall, after completing my Ph.D. at LTI,
CMU, in 2018. Before coming to the US, I completed my undergraduate
study at Wuhan University, China, in 2009 and got a master's degree at
the Institute of Software, Chinese Academy of Science, in 2012.</p>
<p>My work mainly appears in IR, NLP, and Machine Learning
conferences. I also host workshops, present tutorials, and serve in
the organization community of venues in these fields. For the most
updated publication list, please refer to my <a
href="https://scholar.google.com/citations?
user=E9BaEBYAAAAJ&amp;hl=en" rel="external nofollow noopener"
target="_blank">Google Scholar</a>.</p> </div> <div class="social">
<div class="contact-icons"> <a
```

To

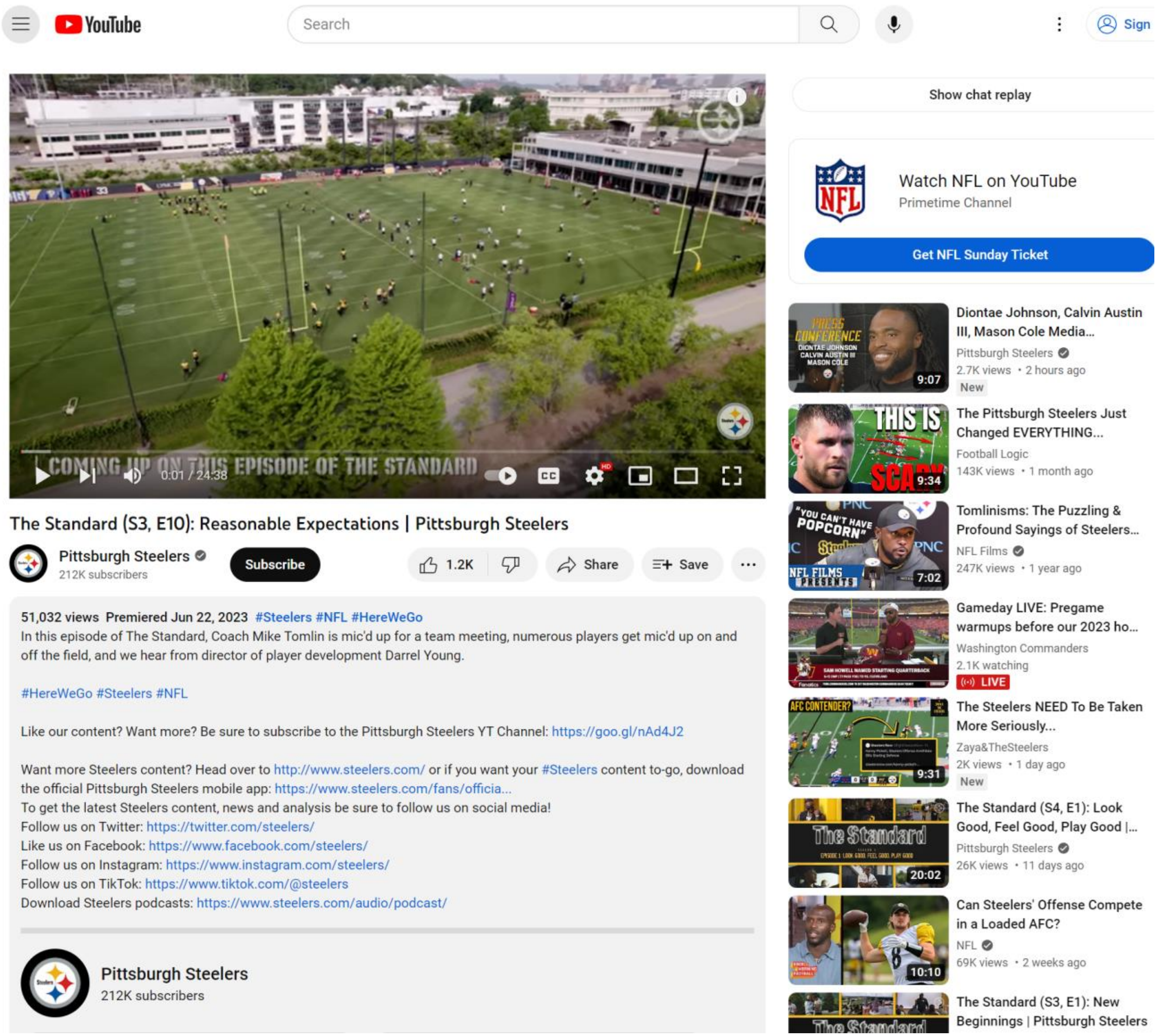
Chenyan Xiong.
I am an Associate Professor at Language Technologies Institute (LTI), Carnegie Mellon University (CMU). My research area is the intersection of information retrieval, natural language processing, and deep learning. Previously I worked at Microsoft Research till 2023 Fall, after completing my Ph.D. at LTI, CMU, in 2018. Before coming to the US, I completed my undergraduate study at Wuhan University, China, in 2009 and got a master's degree at the Institute of Software, Chinese Academy of Science, in 2012.

Scrape Texts from Web Pages: Common Solution

HTML Parsing is a commodity tool for many web related applications

- Many open-source toolkits are available
  - Boilerpipe, selectolax
- Many legacy tools still being used:
  - Common Crawl's official text extraction (WET files) https://htmlparser.sourceforge.net/ has not updated for 10 years
  - Recent one used to produce refined web: Trafilatura
- Often use a series of rules
  - Identify content nodes and extract the text pieces
  - Remove non-useful HTML codes
  - Glue fragmented text pieces to paragraphs

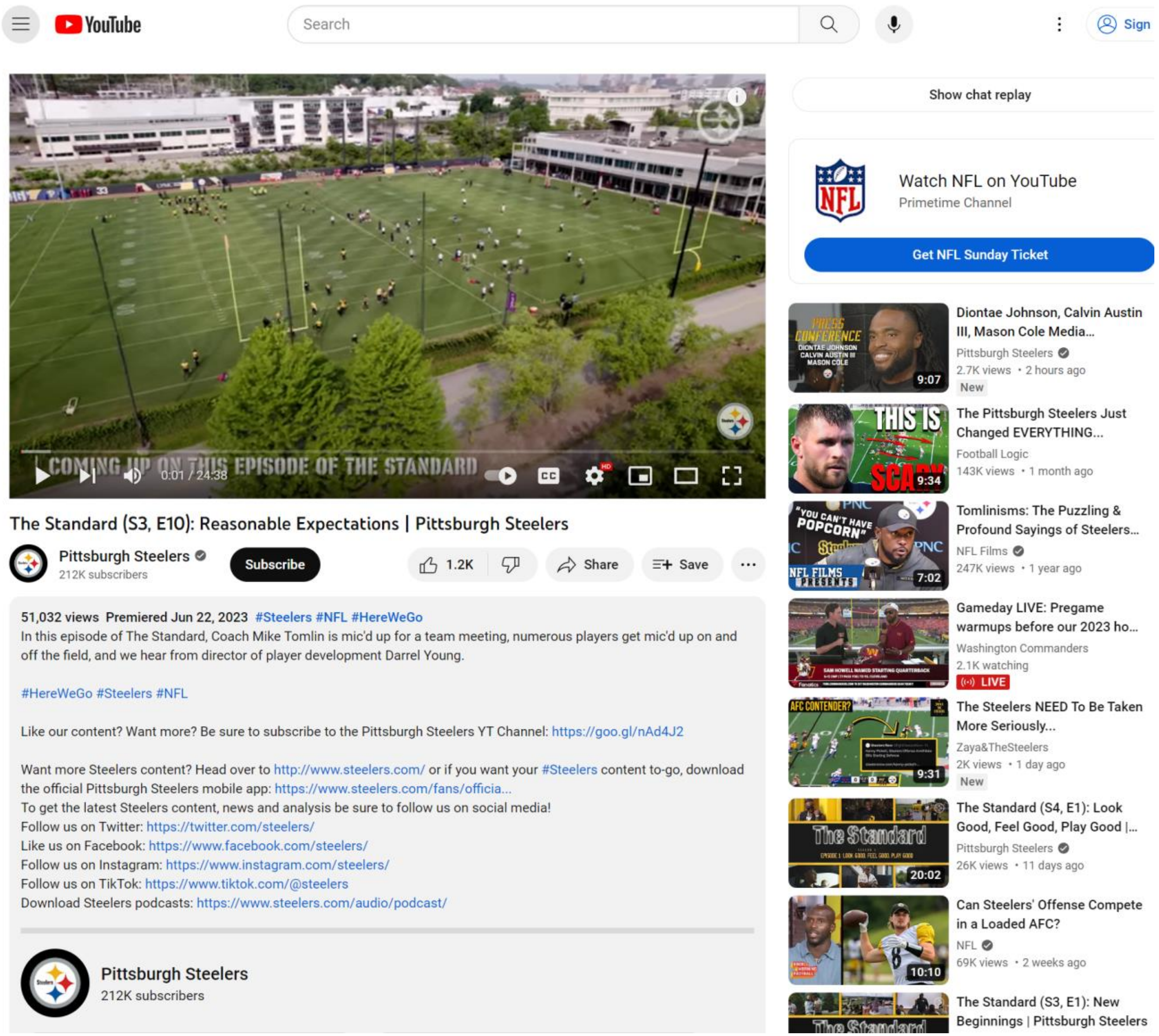
Scrape Texts from Web Pages: Challenges



Web is much more dynamic than static HTMLs

- CSS, Java Scripts, etc.
- Each HTML involves 20+ secondary URLs

Scrape Texts from Web Pages: Challenges



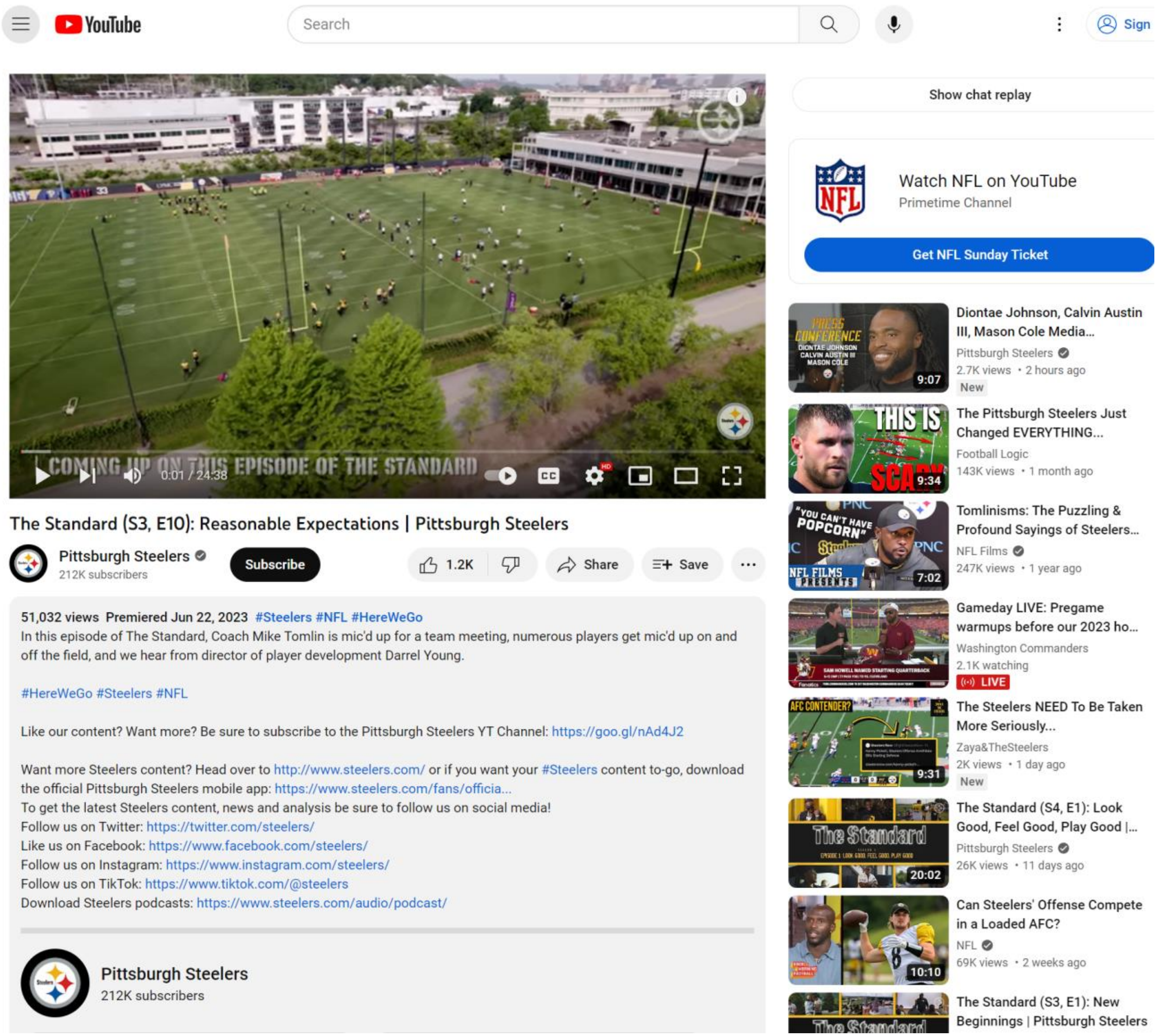
Web is much more dynamic than static HTMLs

- CSS, Java Scripts, etc.
- Each HTML involves 20+ secondary URLs

Content versus Non-contents

- Ads, recommendation, navigation, etc.
- Multi-media
- Spams

Scrape Texts from Web Pages: Challenges



Web is much more dynamic than static HTMLs

- CSS, Java Scripts, etc.
- Each HTML involves 20+ secondary URLs

Content versus Non-contents

- Ads, recommendation, navigation, etc.
- Multi-media
- Spams

Presenting location versus coding location

- Various ways to present a web page
- Only way to know for sure is to render it

Scrape Texts from Web Pages: Commercial Solution

Content extraction is a heavily invested area in web related companies, especially search engine companies

- Initial stage of the system pipeline
- Low quality extractions hard to recover
- Very engineer and resource heavy
- A strategic advantage of some proprietary LLMs

Scrape Texts from Web Pages: Commercial Solution

Content extraction is a heavily invested area in web related companies, especially search engine companies

- Initial stage of the system pipeline
- Low quality extractions hard to recover
- Very engineer and resource heavy
- A strategic advantage of some proprietary LLMs

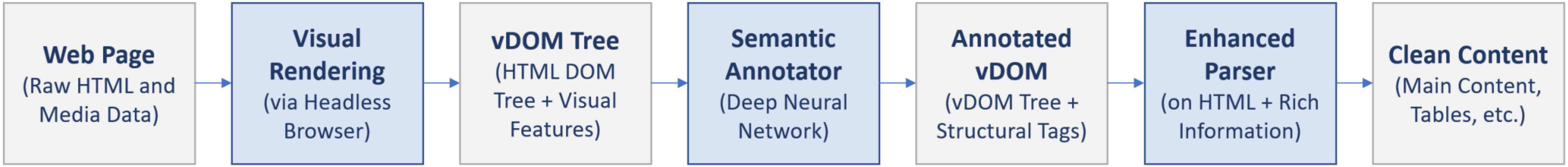


Figure 8: A Content Extraction Pipeline from Bing Used for ClueWeb22

Pretraining Data: Outline

Preprocessing clean texts

- Simple Tokenization
- Subword Tokenization
- Batching

Scaling up pretraining data using the Web

- Obtaining Web Pages
- Scrape Texts
- **Clean Texts**

Pretraining Data Curation from the Web: Clean Texts

Remove noisy, spammy, and fragmented texts

- Non-content texts unlikely to help pretraining

Select higher quality texts from a massive candidate pool

- Given limited pretraining compute budget, pretrain on better texts

Avoid toxic and biased contents

- NSFW contents
- Texts with strong biases

Text Selection for Pretraining

Seed URLs $\xrightarrow{\text{Explore}}$ **Target URLs** $\xrightarrow{\text{Crawl}}$ **HTML Contents** $\xrightarrow{\text{Scrape}}$ **Raw Texts** $\xrightarrow{\text{Clean}}$ **Clean Texts**

Three common approaches to select clean texts

- Rule-based filtering
- Proximity to high-quality content
- Toxic Classifier

Text Selection for Pretraining: Rule-Based Filtering

Rule-based filtering in Colossal Clean Crawled Corpus (C4, T5 pretraining data)

1. Start from Common Crawl's official extracted texts from HTML
2. Only keep text lines ended with a terminal punctuation mark
3. Discard pages with fewer than 5 sentences
4. Only keep lines with at least 3 words
5. Remove any line with the word "Javascript"
6. Remove any page
    1. with any words in a toxic word dictionary
    2. with the phrase "lorem ipsum"
    3. With "{"
7. De-dup at three-sentence span level

Text Selection for Pretraining: C4 Filtering

Experimental Results on T5 base

Table 15: T5 base Pretrained with different datasets [7].

Data Set	GLUE AVG	SQuAD
C4 Filtered	83.3	80.9
C4 Unfiltered	81.5	78.8

[7] Colin, et al. "Exploring the limits of transfer learning with a unified text-to-text transformer.“ JMLR. 2020.

Text Selection for Pretraining: C4 Filtering

Experimental Results on T5 base

Table 15: T5 base Pretrained with different datasets [7].

Data Set	GLUE AVG	SQuAD
C4 Filtered	83.3	80.9
C4 Unfiltered	81.5	78.8

- A list of simple heuristic rules, bigger gains than twisting different Mask-LM variations

Table 9: Pretraining Tasks Results with T5 base [7].

Denoising Task	GLUE AVG	SQuAD
Auto-Regressive LM	80.7	78.0
De-shuffling	73.2	67.6
Mask-LM, Reconstruct All	83.0	80.7
Replace Corrupted Spans	83.3	80.9
Drop Corrupted Tokens	84.4	80.5

[7] Colin, et al. "Exploring the limits of transfer learning with a unified text-to-text transformer.“ JMLR. 2020.

Text Selection for Pretraining: Proximity

Select text sequences “similar” to “good” content

- Good content: often Wikipedia dumps
- Similarity definition:
  - Perplexity on a language model pretrained on good content
  - Classification model trained using good content as positive, rest as negative

Text Selection for Pretraining: Proximity

Select text sequences “similar” to “good” content

- Good content: often Wikipedia dumps
- Similarity definition:
  - Perplexity on a language model pretrained on good content
  - Classification model trained using good content as positive, rest as negative

Performance:

- Believed to be effective, especially on noisy content
  - At least filter out non-language sequences
- Not necessarily helpful, e.g., on filtered, less-noisy content

Text Selection for Pretraining: Proximity

Select text sequences "similar" to "good" content

- Good content: often Wikipedia dumps
- Similarity definition:
  - Perplexity on a language model pretrained on good content
  - Classification model trained using good content as positive, rest as negative

Performance:

- Believed to be effective, especially on noisy content
  - At least filter out non-language sequences
- Not necessarily helpful, e.g., on filtered, less-noisy content

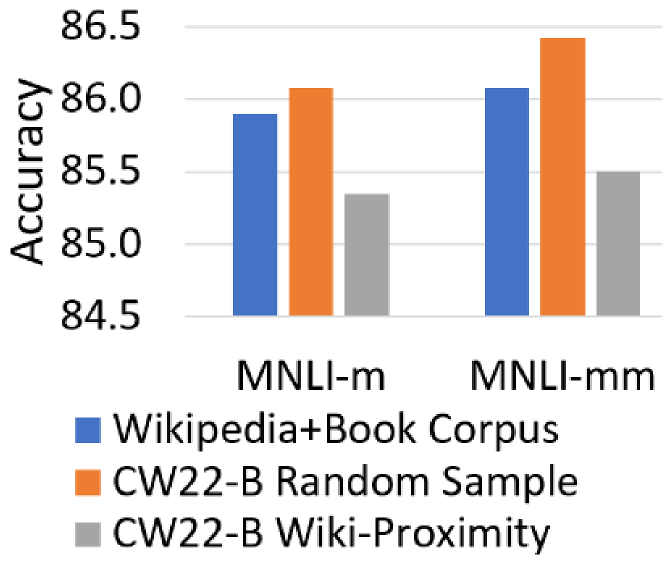


Figure 9: RoBERTa base pretrained on different datasets

Text Selection for Pretraining: Toxic Filter

Remove toxic contents

- By word-based filter (toxic words)
- By toxic classifier
  - Trained using an existing set of toxic texts as filtering target
- Necessary when dealing with open web data

Text Selection for Pretraining: Toxic Filter

Remove toxic contents

- By word-based filter (toxic words)
- By toxic classifier
  - Trained using an existing set of toxic texts as filtering target
- Necessary when dealing with open web data

A challenging problem:

- Variant ways of “being toxic”
- Some toxicity is explicit, but some, e.g., biases are implicit
- Definition of toxic varies, often a question for our society

Pretraining Data: Final Remarks

A crucial component of the pretraining ecosystem

- Engineering resource, technical, and investment heavy area
- Huge impacts on LLM capabilities

Large gaps between proprietary LLMs and open LLMs

- Companies consider data a strategic advantage and become more and more secretive
- An important but less clean problem, often avoided by some academics

Many concerns beyond technology

- Toxic, biases, and privacy
- Copyrights

Quiz: What the pros and cons of pretraining on web data compared with only on Wikipedia?

References: Pretraining Data

- [BPE] Sennrich, Rico, Barry Haddow, and Alexandra Birch. "Neural machine translation of rare words with subword units." arXiv preprint arXiv:1508.07909 (2015).
- [SentencePiece] Kudo, Taku, and John Richardson. "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing." arXiv preprint arXiv:1808.06226 (2018).
- [BERT] Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In Proceedings of NAACL-HLT, pp. 4171-4186. 2019.
- [GPT-2] Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI blog 1.8 (2019): 9.
- [RoBERTa] Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).
- [XL-NET] Yang, Zhilin, et al. "XLNet: Generalized autoregressive pretraining for language understanding." Advances in neural information processing systems 32 (2019).
- [T5] Raffel, Colin, et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." The Journal of Machine Learning Research 21.1 (2020): 5485-5551.
- [The PILE] Gao, Leo, et al. "The pile: An 800gb dataset of diverse text for language modeling." arXiv preprint arXiv:2101.00027 (2020).
- [GaLM] Du, Nan, et al. "Glam: Efficient scaling of language models with mixture-of-experts." International Conference on Machine Learning. PMLR, 2022.
- [Don’t Stop] Gururangan, Suchin, et al. "Don't stop pretraining: Adapt language models to domains and tasks." arXiv preprint arXiv:2004.10964 (2020).
- [Med] Gu, Yu, et al. "Domain-specific language model pretraining for biomedical natural language processing." ACM Transactions on Computing for Healthcare (HEALTH) 3.1 (2021): 1-23.