

Asynchronous Pipeline for Processing Huge Corpora on Medium to Low Resource Infrastructures

Pedro Javier ORTIZ SUÁREZ^{1,2}, Benoît SAGOT¹, Laurent ROMARY¹

July 21, 2019

¹Inria, Paris, France

²Sorbonne Université, Paris, France

Table of contents

1. Introduction
2. Common Crawl
3. fastText Pipeline
4. Asynchronous pipeline
5. Benchmarks
6. OSCAR

Introduction

Transfer Learning in NLP

Transfer Learning models for Natural Language Processing (NLP) have consistently and repeatedly improved the state-of-the-art in a wide variety of NLP tasks. There are two types of Transfer Learning models:

Non contextualised:

- word2vec [9],
- GloVe [11],
- fastText [8].

Contextualised:

- ELMo [12],
- GPT-1 [13],
- GPT-2 [14],
- BERT [4],
- XLNet [15].

Even though these models have clear advantages, their main drawback is the amount of data that is needed to train them in order to obtain a functional and efficient model:

English models:

- **word2vec**: one billion word dataset (news articles) [9].
- **fastText**: Common Crawl [8].
- **ELMo**: 5.5 billion token dataset (crawled news + Wikipedia) [12].
- **BERT**: 3.3 billion word corpus (Wikipedia + BooksCorpus [16]) [4].
- **GPT-2**: 40GB corpus (scraping outbound links from Reddit) [14].
- **XLNet**: BookCorpus, Wikipedia, Giga5 [10], ClueWeb 2012-B [3] and Common Crawl.

In comparison the availability of models in languages other than English has been rather limited.

Multilingual models:

- `word2vec/fastText`: plain text from Wikipedia [1, 2].
- `fastText`: Common Crawl (157 different languages) [5].

Moreover, even some of the bigger English corpora mentioned above are not made available by the authors due to copyright issues or to the infrastructure costs attached to maintaining and distributing such big corpora.

Common Crawl

Common Crawl

Common Crawl is a non-profit foundation which produces and maintains an open repository of web crawled data that is both accessible and analysable.

Common Crawl's complete web archive consists of petabytes of data collected over 8 years of web crawling. The repository comprises:

- **WARC files:** raw web page HTML data,
- **WAT files:** metadata extracts,
- **WET files:** plain text extracts.

The organisation's crawlers respects both `nofollow` and `robots.txt` policies.

Common Crawl Snapshots

To address the problems of getting large corpora for multiple languages we chose to work with one of the monthly snapshots of Common Crawl.

Each snapshot is in itself a **massive multilingual corpus** (about 20TB of plain text), where every single file contains data coming from multiple web pages written in a large variety of languages and covering all possible types of topics.

In order to effectively use this corpus for the mentioned Natural Language Processing and Machine Learning applications, one has first to **extract, filter, clean** and **classify** the data in the snapshot by language.

fastText Pipeline

fastText Pipeline

The `fastText` authors have already proposed a pipeline to filter, clean and classify Common Crawl [5].

Their solution, consisting of a series of BASH scripts:

- Is a **synchronous blocking pipeline**,
- Works well on infrastructures having high I/O speeds,
- But **downscales poorly** to medium-low resource infrastructures using electromechanical mediums.

Even though they open source their pipeline under an MIT-license, they **do not distribute** their cleaned version of Common Crawl

fastText Pipeline

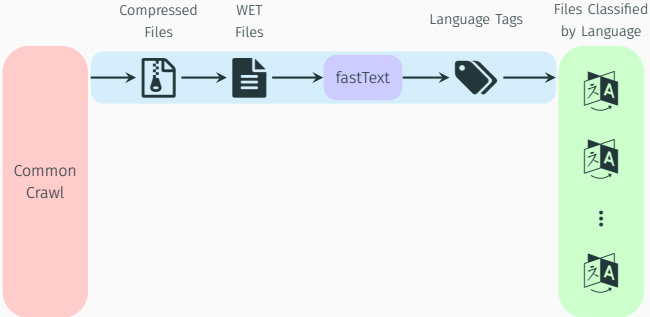
The fasttext pipeline consists of two stages:

1. Their pipeline first launches multiple process, preferably as many as available cores. Each process:
 - 1.1 Downloads one Common Crawl WET file.
 - 1.2 Decompresses The WET.
 - 1.3 Launches an instance of the fastText linear classifier [6, 7].
 - 1.4 The classifier generates a language tag for each line in the WET file.
 - 1.5 The tags are used to append each line (longer than 100 bytes) to its corresponding language file.
2. When all the WET files are classified, again a number of processes are launched, each process:
 - 2.1 Reads a language file and filters for invalid UTF-8 characters.
 - 2.2 Reads the filtered file and performs deduplication.

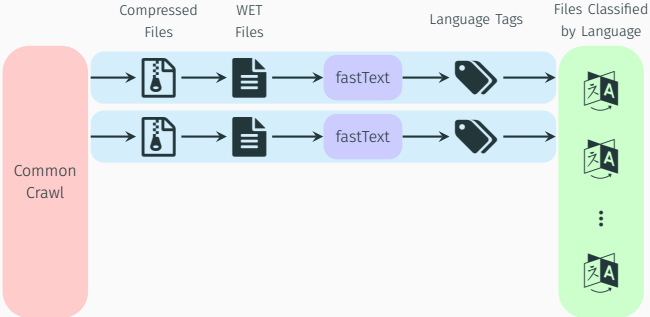


Common
Crawl

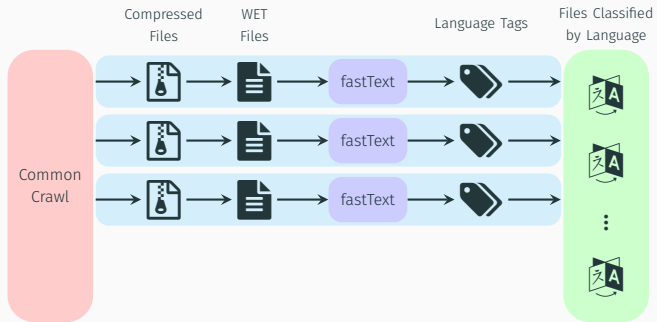
fastText Pipeline



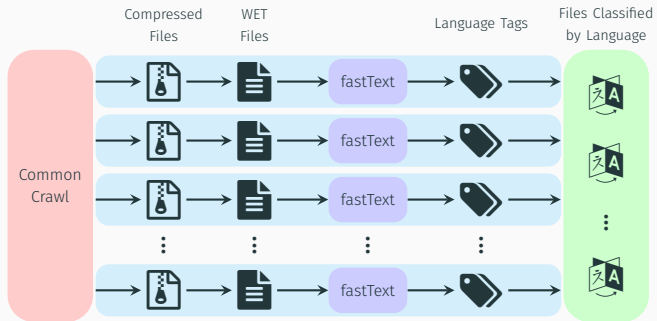
fastText Pipeline



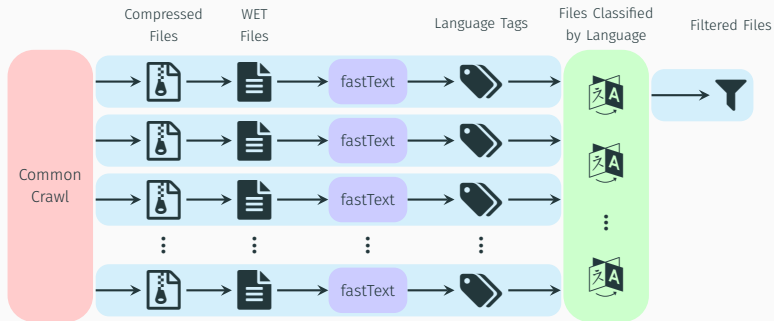
fastText Pipeline



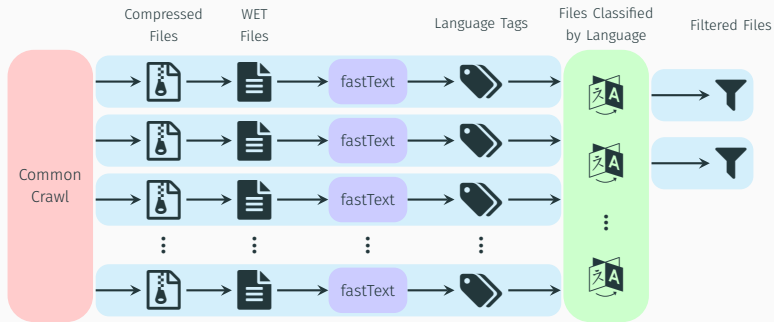
fastText Pipeline



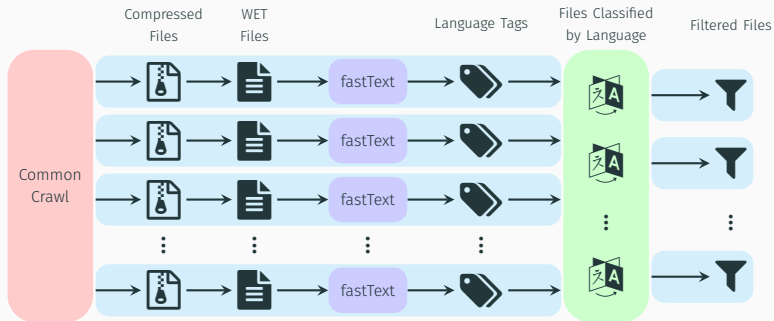
fastText Pipeline



fastText Pipeline



fastText Pipeline



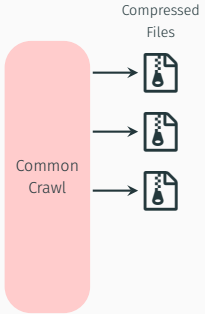
Asynchronous pipeline

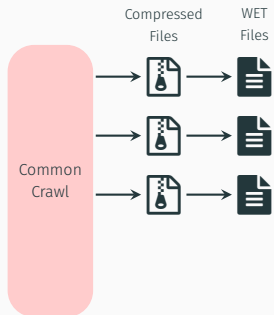
We propose a new pipeline derived from the fastText one which we call **goclassy**, we reuse the fastText linear classifier and the pre-trained fastText model for language recognition[5]. In our pipeline

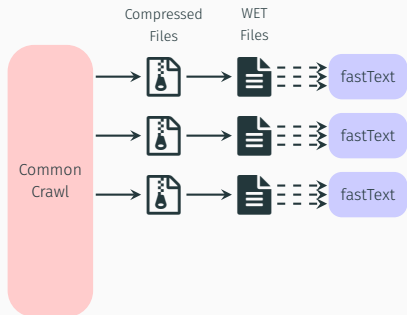
- We launch a worker for each operation instead of clustering multiple operations into a single process.
- We implement goclassy using the Go programming language and let the Go runtime handle the scheduling.
- We introduced buffers in all our I/O operations to reduce I/O blocking.
- We do the filtering (of lines shorter than 100 UTF-8 characters) and cleaning processes at line level before feeding each line to the classifier.
- After all WET files are processed, we then use **runiq** for deduplication and **pigz** for compression.

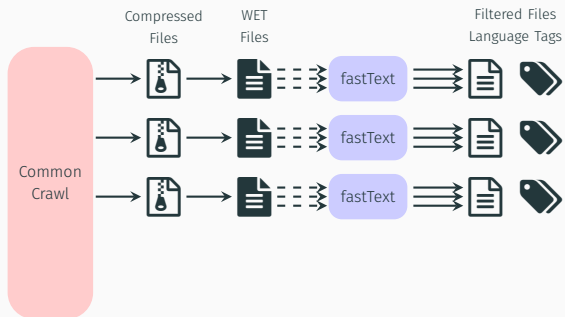


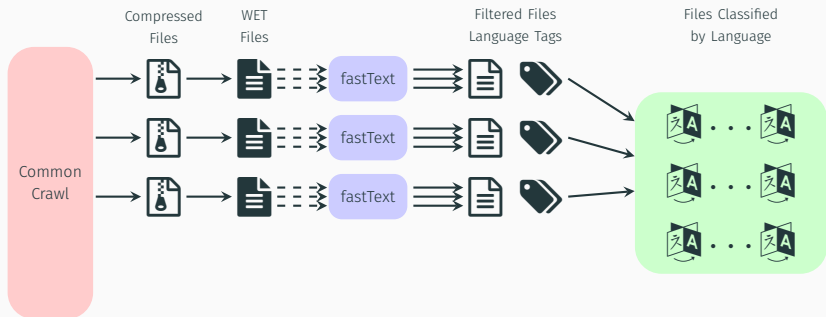
Common
Crawl

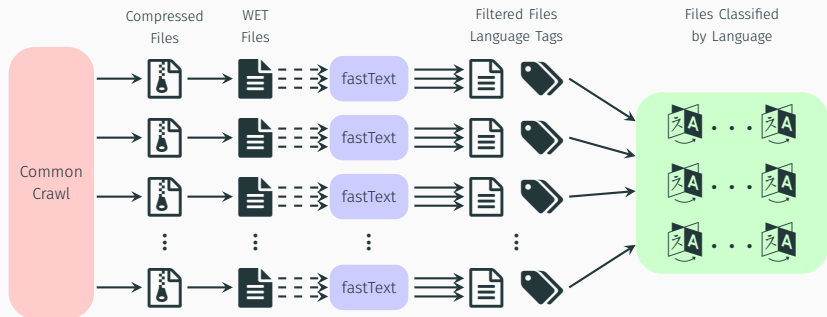












Benchmarks

Benchmarks

We test both pipelines against one another in an infrastructure using traditional hard drives that are connected to the main processing machine via an Ethernet interface. The machine we use has

- an Intel® Xeon® Processor E5-2650 2.00 GHz,
- 20M Cache,
- 203.1 GiB of RAM.

We do not include downloading, decompression or deduplication in our benchmarks. We use the **time** UNIX tool for our benchmark.

Benchmarks

Pipeline	10 files	100 files	200 files
<i>real</i>			
fastText	3m31s	17m39s	31m4s
goclassy	1m42s	9m8s	15m16s
<i>user</i>			
fastText	26m53s	4h23m	8h45m
goclassy	11m0s	1h49m	3h38m
<i>sys</i>			
fastText	40.56s	6m15s	12m31s
goclassy	39.67s	5m16s	10m5s

OSCAR

We publish¹ a pre-processed version of the November 2018 snapshot of Common Crawl which is comprised of usable data in 166 different languages, we our corpus under the name **OSCAR** which is short for


Open Super-large Crawled ALMANACH² coRpus.

After processing all the data with goclassy, the size of the whole Common Crawl snapshot is reduced to **6.3TB**, but in spite of this considerable reduction, OSCAR still dwarfs all previous mentioned corpora having more **800 billion “words”** or spaced separated tokens and noting that this in fact is an understatement of how big OSCAR is, as some of the largest languages within OSCAR **do not use spaces**.

¹<https://team.inria.fr/almanach/oscar/>

²<https://team.inria.fr/almanach/>

Language	Size		Words	
	Orig	Dedup	Orig	Dedup
English	2.3T	1.2T	418,187,793,408	215,841,256,971
Russian	1.2T	568G	92,522,407,837	46,692,691,520
Spanish	278G	149G	47,545,122,279	25,928,290,729
French	282G	138G	46,896,036,417	23,206,776,649
German	308G	145G	44,878,908,446	21,529,164,172
Italian	137G	69G	22,248,707,341	11,250,012,896
Portuguese	124G	64G	20,641,903,898	10,751,156,918
Chinese	508G	249G	14,986,424,850	6,350,215,113
Japanese	216G	106G	4,962,979,182	1,123,067,063
Polish	109G	47G	15,277,255,137	6,708,709,674
Total OSCAR	6.3T	3.2T	844,315,434,723	425,651,344,234

Following the example of similar corpora, like ParaCrawl³ (Broader Web-Scale Provision of Parallel Corpora for European Languages). We license the packaging of OSCAR under the **Creative Commons CC0 license ("no rights reserved")**⁴ .

We also note that **We do not own any of the text from which these data has been extracted.**

Finally, should someone consider that our data contains material that is owned by them and should therefore not be reproduced in OSCAR, we put in place a form in our site allowing that person to point out the content so that we can delete from OSCAR

³<https://paracrawl.eu/>

⁴<http://creativecommons.org/publicdomain/zero/1.0>

Thank you!

Questions?



R. Al-Rfou, B. Perozzi, and S. Skiena.

Polyglot: Distributed word representations for multilingual NLP.

In Proceedings of the Seventeenth Conference on Computational Natural Language Learning, pages 183–192, Sofia, Bulgaria, Aug. 2013. Association for Computational Linguistics.



P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov.

Enriching word vectors with subword information.

Transactions of the Association for Computational Linguistics, 5:135–146, 2017.



J. Callan, M. Hoy, C. Yoo, and L. Zhao.

Clueweb09 data set, 2009.



J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova.
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

arXiv e-prints, page arXiv:1810.04805, Oct 2018.



E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov.
Learning word vectors for 157 languages.

In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan, May 2018. European Language Resource Association.



A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov.

Fasttext.zip: Compressing text classification models.

CoRR, abs/1612.03651, 2016.



A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov.

Bag of tricks for efficient text classification.

In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain, Apr. 2017. Association for Computational Linguistics.



T. Mikolov, E. Grave, P. Bojanowski, C. Puhersch, and A. Joulin.

Advances in pre-training distributed word representations.

In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018.*, 2018.



T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean.
Distributed representations of words and phrases and their compositionality.

In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.



R. Parker, D. Graff, J. Kong, K. Chen, and K. Maeda.
English gigaword fifth edition, linguistic data consortium.

Technical report, Technical Report. Linguistic Data Consortium, 2011.



J. Pennington, R. Socher, and C. Manning.

Glove: Global vectors for word representation.




In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.



M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer.

Deep contextualized word representations.

In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

-  A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever.
Improving language understanding by generative pre-training.
OpenAI Blog, 2018.
-  A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever.
Language models are unsupervised multitask learners.
OpenAI Blog, 1:8, 2019.
-  Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le.
XLnet: Generalized autoregressive pretraining for language understanding.
CoRR, abs/1906.08237, 2019.



Y. Zhu, R. Kiros, R. S. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler.

Aligning books and movies: Towards story-like visual explanations by watching movies and reading books.

In 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015, pages 19–27. IEEE Computer Society, 2015.