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

Pedro Javier ORTIZ SUÁREZ¹ ², Benoît SAGOT¹, Laurent ROMARY¹
July 21, 2019

¹Inria, Paris, France
²Sorbonne Université, Paris, France
Introduction
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].
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 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.
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
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.
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 that 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.
fastText Pipeline

- Common Crawl
- Compressed Files
- WET Files
- Language Tags
- Files Classified by Language
- fastText
- Filtered Files
fastText Pipeline

Compressed Files → WET Files → fastText → Language Tags → Files Classified by Language

Common Crawl
fastText Pipeline

- Compressed Files
- WET Files
- Language Tags
- Files Classified by Language

Diagram shows the flow from Compressed Files, through WET Files, using fastText for classification, leading to Files Classified by Language.
fastText Pipeline

Compressed Files → WET Files → fastText → Language Tags → Files Classified by Language → Filtered Files

Common Crawl

...
fastText Pipeline

Common Crawl

Compressed Files → WET Files → fastText → Language Tags → Files Classified by Language → Filtered Files
fastText Pipeline

Compressed Files → WET Files → Language Tags → Files Classified by Language → Filtered Files

Common Crawl

...
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.
goclassy

Compressed Files → WET Files

Common Crawl

Filtered Files

Language Tags

Files Classified by Language
Common Crawl

Compressed Files

WET Files

file

file

file

fastText

fastText

fastText

...
Common Crawl

Compressed Files → WET Files → fastText → Filtered Files Language Tags

...
Common Crawl

Compressed Files → WET Files → fastText → Filtered Files Language Tags → Files Classified by Language

11
Common Crawl

Compressed Files → WET Files → fastText

Filtered Files Language Tags

Files Classified by Language
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.
<table>
<thead>
<tr>
<th>Pipeline</th>
<th>10 files</th>
<th>100 files</th>
<th>200 files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>real</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fastText</td>
<td>3m31s</td>
<td>17m39s</td>
<td>31m4s</td>
</tr>
<tr>
<td>goclassy</td>
<td>1m42s</td>
<td>9m8s</td>
<td>15m16s</td>
</tr>
<tr>
<td><strong>user</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fastText</td>
<td>26m53s</td>
<td>4h23m</td>
<td>8h45m</td>
</tr>
<tr>
<td>goclassy</td>
<td>11m0s</td>
<td>1h49m</td>
<td>3h38m</td>
</tr>
<tr>
<td><strong>sys</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fastText</td>
<td>40.56s</td>
<td>6m15s</td>
<td>12m31s</td>
</tr>
<tr>
<td>goclassy</td>
<td>39.67s</td>
<td>5m16s</td>
<td>10m5s</td>
</tr>
</tbody>
</table>
OSCAR
We publish\footnote{https://team.inria.fr/almanach/oscar/} 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 \textit{Open Super-large Crawled ALMAnaCH}\footnote{https://team.inria.fr/almanach/} corpus.

After processing all the data with goclassy, the size of the whole Common Crawl snapshot is reduced to \textbf{6.3TB}, but in spite of this considerable reduction, OSCAR still dwarfs all previous mentioned corpora having more \textbf{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.
<table>
<thead>
<tr>
<th>Language</th>
<th>Size</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Orig</td>
<td>Dedup</td>
</tr>
<tr>
<td>English</td>
<td>2.3T</td>
<td>1.2T</td>
</tr>
<tr>
<td>Russian</td>
<td>1.2T</td>
<td>568G</td>
</tr>
<tr>
<td>Spanish</td>
<td>278G</td>
<td>149G</td>
</tr>
<tr>
<td>French</td>
<td>282G</td>
<td>138G</td>
</tr>
<tr>
<td>German</td>
<td>308G</td>
<td>145G</td>
</tr>
<tr>
<td>Italian</td>
<td>137G</td>
<td>69G</td>
</tr>
<tr>
<td>Portuguese</td>
<td>124G</td>
<td>64G</td>
</tr>
<tr>
<td>Chinese</td>
<td>508G</td>
<td>249G</td>
</tr>
<tr>
<td>Japanese</td>
<td>216G</td>
<td>106G</td>
</tr>
<tr>
<td>Polish</td>
<td>109G</td>
<td>47G</td>
</tr>
<tr>
<td><strong>Total OSCAR</strong></td>
<td><strong>6.3T</strong></td>
<td><strong>3.2T</strong></td>
</tr>
</tbody>
</table>
Following the example of similar corpora, like ParaCrawl\(^3\) (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/](https://paracrawl.eu/)

\(^4\) [http://creativecommons.org/publicdomain/zero/1.0](http://creativecommons.org/publicdomain/zero/1.0)
Thank you!
Questions?


A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. 
**Bag of tricks for efficient text classification.**

**Advances in pre-training distributed word representations.**
T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. 
Distributed representations of words and phrases and their compositionality.

English gigaword fifth edition, linguistic data consortium.
J. Pennington, R. Socher, and C. Manning. 
**Glove: Global vectors for word representation.**

**Deep contextualized word representations.**
Improving language understanding by generative pre-training. 
*OpenAI Blog*, 2018.

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. 
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.