Sorbonne Université

Ecole Doctorale Informatique, Télécommunications et Electronique - ED130

Inria de Paris / Équipe ALMAnaCH

Thèse de doctorat
Discipline : Informatique

Présentée par

Pedro Ortiz Suarez

Dirigée par

Laurent Romary et **Benoît Sagot**

Pour obtenir le grade universitaire de

Docteur de Sorbonne Université

On Language Modeling and its Applications for Contemporary and Historical French

---

Présentée et soutenue publiquement le 30 avril 2022 devant le jury composé de :

Francis Bach	Inria - SIERRA	Examinateur
Alexander Geyken	Berlin Academy	Examinateur
Julia Kreutzer	University of Stuttgart	Examinateur
Barbara Plank	IT University of Copenhagen	Rapporteur
Laurent Romary	Inria - ALMAnaCH	Directeur
Benoît Sagot	Inria - ALMAnaCH	Directeur
Holger Schwenk	Facebook AI Research	Rapporteur

Abstract

Scientific documents often use LaTeX for typesetting. While numerous packages and templates exist, it makes sense to create a new one. Just because.

Contents

1 Introduction

> In which the reasons for doing this Ph.D. are laid bare for the whole world to see and we encounter some answers to questions in which, frankly, only an extremely small number of people were interested in the first place.

This package contains a minimal, modern template for writing your thesis. While originally meant to be used for a Ph. D. thesis, you can equally well use it for your honour thesis, bachelor thesis, and so on—some adjustments may be necessary, though.

1.1 Why?

I was not satisfied with the available templates for LaTeX and wanted to heed the style advice given by people such as Robert Bringhurst or Edward R. Tufte . While there *are* some packages out there that attempt to emulate these styles, I found them to be either too bloated, too playful, or too constraining. This template attempts to produce a beautiful look without having to resort to any sort of hacks. I hope you like it.

1.2 How?

The package tries to be easy to use. If you are satisfied with the default settings, just add

```
\documentclass{mimosis}
```

at the beginning of your document. This is sufficient to use the class. It is possible to build your document using either LaTeX|, XeLaTeX, or LuaLaTeX. I personally prefer one of the latter two because they make it easier to select proper fonts.

Package	Purpose
`amsmath`	Basic mathematical typography
`amsthm`	Basic mathematical environments for proofs etc.
`booktabs`	Typographically light rules for tables
`bookmarks`	Bookmarks in the resulting PDF
`dsfont`	Double-stroke font for mathematical concepts
`graphicx`	Graphics
`hyperref`	Hyperlinks
`multirow`	Permits table content to span multiple rows or columns
`paralist`	Paragraph ('in-line') lists and compact enumerations
`scrlayer-scrpage`	Page headings
`setspace`	Line spacing
`siunitx`	Proper typesetting of units
`subcaption`	Proper sub-captions for figures

Table 1.1: A list of the most relevant packages required (and automatically imported) by this template.

1.3 Features

The template automatically imports numerous convenience packages that aid in your typesetting process. Table 1.1 lists the most important ones. Let's briefly discuss some examples below. Please refer to the source code for more demonstrations.

1.3.1 Typesetting mathematics

This template uses `amsmath` and `amssymb`, which are the de-facto standard for typesetting mathematics. Use numbered equations using the `equation` environment. If you want to show multiple equations and align them, use the `align` environment:

$$V := \{1, 2, \dots\} \tag{1.1}$$
$$E := \{(u, v) \mid \operatorname{dist}(p_u, p_v) \leq \epsilon\} \tag{1.2}$$

Define new mathematical operators using `\DeclareMathOperator`. Some operators are already pre-defined by the template, such as the distance between two objects. Please see the template for some examples. Moreover, this template contains a correct differential operator. Use `\diff` to typeset the differential of integrals:

$$f(u) := \int_{v \in \mathbb{D}} \operatorname{dist}(u, v) \, \mathrm{d}v \tag{1.3}$$

You can see that, as a courtesy towards most mathematicians, this template gives you the possibility to refer to the real numbers $\mathbb{R}$ and the domain $\mathbb{D}$ of some function.

Take a look at the source for more examples. By the way, the template comes with spacing fixes for the automated placement of brackets.

1.3.2 Typesetting text

Along with the standard environments, this template offers `paralist` for lists within paragraphs. Here's a quick example: The American constitution speaks, among others, of (i) life (ii) liberty (iii) the pursuit of happiness. These should be added in equal measure to your own conduct. To typeset units correctly, use the `siunitx` package. For example, you might want to restrict your daily intake of liberty to 750 mg.

Likewise, as a small pet peeve of mine, I offer specific operators for *ordinals*. Use `\th` to typeset things like July 4$^{\text{th}}$ correctly. Or, if you are referring to the 2$^{\text{nd}}$ edition of a book, please use `\nd`. Likewise, if you came in 3$^{\text{rd}}$ in a marathon, use `\rd`. This is my 1$^{\text{st}}$ rule.

1.4 Changing things

Since this class heavily relies on the `scrbook` class, you can use *their* styling commands in order to change the look of things. For example, if you want to change the text in sections to **bold** you can just use

```
\setkomafont{sectioning}{\normalfont\bfseries}
```

at the end of the document preamble—you don't have to modify the class file for this. Please consult the source code for more information.

Part I

OSCAR

2 Goclassy

2.1 Introduction

In recent years neural methods for Natural Language Processing (NLP) have consistently and repeatedly improved the state-of-the-art in a wide variety of NLP tasks such as parsing, PoS-tagging, named entity recognition, machine translation, text classification and reading comprehension among others. Probably the main contributing factor in this steady improvement for NLP models is the raise in usage of *transfer learning* techniques in the field. These methods normally consist of taking a pre-trained model and reusing it, with little to no retraining, to solve a different task from the original one it was intended to solve; in other words, one *transfers* the *knowledge* from one task to another.

Most of the transfer learning done in NLP nowadays is done in an unsupervised manner, that is, it normally consist of a *language model* that is fed unannotated plain text in a particular language; so that it *extracts* or *learns* the basic *features* and patterns of the given language, the model is subsequently used on top of an specialised architecture designed to tackle a particular NLP task. Probably the best known example of this type of model are *word embeddings* which consist of real-valued vector representations that are trained for each word on a given corpus. Some notorious examples of word embeddings are word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and fastText (Mikolov et al., 2018). All these models are *context-free*, meaning that a given word has one single vector representation that is independent of context, thus for a polysemous word like Washington, one would have one single representation that is reused for the city, the state and the US president.

In order to overcome the problem of polysemy, *contextual* models have recently appeared. Most notably ELMo (Peters et al., 2018) which produces deep contextualised word representations out of the internal states of a deep bidirectional language model in order to model word use and how the usage varies across linguistic contexts. ELMo still needs to be used alongside a specialised architecture for each given downstream task, but newer architectures that can be fine-tuned have also appear. For these, the model is first fed unannotated data, and is then fine-tuned with annotated data to a particular downstream task without relying on any other architecture. The most remarkable examples of this type of model are GPT-1, GPT-2 (Radford et al., 2018, 2019), BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019); the latter being the current state-of-the-art for multiple downstream tasks. All of these models are different arrangements of the Transformer architecture (Vaswani et al., 2017) trained with different datasets, except for XLNet which is an instance of the Transformer-XL (Dai et al., 2019).

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. For the first English version of word2vec, Mikolov et al. (2013) used a one billion word dataset consisting of various news articles. Later Al-Rfou' et al. (2013) and then Bojanowski et al. (2017) used the plain text from Wikipedia to train

distributions of word2vec and fastText respectively, for languages other than English. Now, the problem of obtaining large quantities of data aggravates even more for contextual models, as they normally need multiple instances of a given word in order to capture all its different uses and in order to avoid overfitting due to the large quantity of hyperparameters that these models have. Peters et al. (2018) for example use a 5.5 billion token[1] dataset comprised of crawled news articles plus the English Wikipedia in order to train ELMo, Devlin et al. (2019) use a 3.3 billion word[2] corpus made by merging the English Wikipedia with the BooksCorpus (Zhu et al., 2015), and Radford et al. (2019) use a 40GB English corpus created by scraping outbound links from Reddit.[3]

While Wikipedia is freely available, and multiple pipelines exist[4,5] to extract plain text from it, some of the bigger corpora mentioned above are not made available by the authors either due to copyright issues or probably because of the infrastructure needed to serve and distribute such big corpora. Moreover the vast majority of both these models and the corpora they are trained with are in English, meaning that the availability of high quality NLP for other languages, specially for low-resource languages, is rather limited.

To address this problem, we choose Common Crawl[6], which is a 20TB mutilingual free to use corpus composed of crawled websites from the internet, and we propose a highly parallel multithreaded asynchronous pipeline that applies well-known concurrency patterns, to clean and classify by language the whole Common Crawl corpus to a point where it is usable for Machine Learning and in particular for neural NLP applications. We optimise the pipeline so that the process can be completed in a sensible amount of time even in infrastructures where Input/Output (I/O) speeds become the main bottleneck.

Knowing that even running our pipeline will not always be feasible, we also commit to publishing our own version of a classified by language, filtered and ready to use Common Crawl corpus upon publication of this article. We will set up an easy to use interface so that people can download a manageable amount of data on a desired target language.

2.2 Related Work

Common Crawl has already been successfully used to train language models, even multilingual ones. The most notable example in probably fastText which was first trained for English using Common Crawl (Mikolov et al., 2018) and then for other 157

[1] Punctuation marks are counted as tokens.
[2] Space sparated tokens.
[3] https://www.reddit.com/
[4] https://github.com/attardi/wikiextractor
[5] https://github.com/hghodrati/wikifil
[6] http://commoncrawl.org/

different languages (Grave et al., 2018). In fact Grave et al. (2018) proposed a pipeline to filter, clean and classify Common Crawl, which we shall call the "fastText preprocessing pipeline." They used the fastText linear classifier (Joulin et al., 2016; Joulin et al., 2017) to classify each line of Common Crawl by language, and downloaded the initial corpus and schedule the I/O using some simple Bash scripts. Their solution, however, proved to be a synchronous blocking pipeline that works well on infrastructures having the necessary hardware to assure high I/O speeds even when storing tens of terabytes of data at a time. But that downscales poorly to medium-low resource infrastructures that rely on more traditional cost-effective electromechanical mediums in order to store this amount of data.

Concerning contextual models, Baevski et al. (2019) trained a BERT-like bi-directional Transformer for English using Common Crawl. They followed the "fastText preprocessing pipeline" but they removed all copies of Wikipedia inside Common Crawl. They also trained their model using News Crawl (Bojar et al., 2018) and using Wikipedia + BooksCorpus, they compared three models and showed that Common Crawl gives the best performance out of the three corpora.

The XLNet model was trained for English by joining the BookCorpus, English Wikipedia, Giga5 (Parker et al., 2011), ClueWeb 2012-B (Callan et al., 2009) and Common Crawl. Particularly for Common Crawl, Yang et al. (2019) say they use "heuristics to aggressively filter out short or low-quality articles" from Common Crawl, however they don't give any detail about these "heuristics" nor about the pipeline they use to classify and extract the English part of Common Crawl.

It is important to note that none of these projects distributed their classified, filtered and cleaned versions of Common Crawl, making it difficult in general to faithfully reproduce their results.

2.3 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.[7] Common Crawl's complete web archive consists of petabytes of data collected over 8 years of web crawling. The repository contains raw web page HTML data (WARC files), metdata extracts (WAT files) and plain text extracts (WET files). The organisation's crawlers has always respected `nofollow`[8] and `robots.txt`[9] policies.

Each monthly Common Crawl snapshot is in itself a massive multilingual corpus, 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. Thus, in order to effectively use this corpus for the previously mentioned Natural Language Processing

[7] http://commoncrawl.org/about/
[8] http://microformats.org/wiki/rel-nofollow
[9] https://www.robotstxt.org/

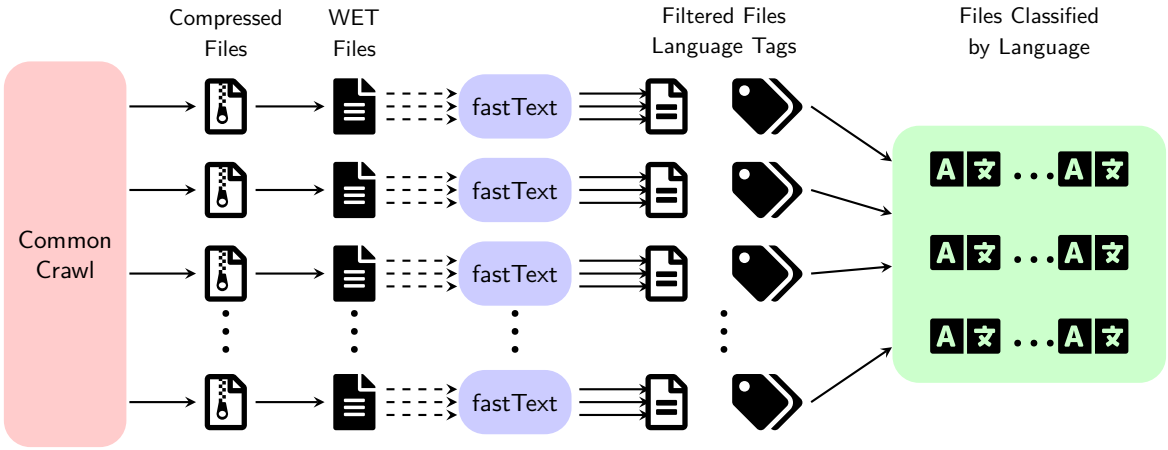


Figure 2.1: A scheme of the *goclassy* pipeline. The red square represents the Compressed WET files stored on Amazon Web Services. The icons represent the gzip files stored locally, the represent one of the 50K WET files. The represents the filtered file and the represents a file of language tags, one tag per line in . The represents one of the 166 classified files. Each arrow represents an asynchronous non blocking worker and dotted arrows represent a line filtering process.

and Machine Learning applications, one has first to extract, filter, clean and classify the data in the snapshot by language.

For our purposes we use the WET files which contain the extracted plain texts from the websites mostly converted to UTF-8, as well as headers containing the metatada of each crawled document. Each WET file comes compressed in gzip format[10] and is stored on Amazon Web Services. We use the November 2018 snapshot which surpasses 20TB of uncompressed data and contains more than 50 thousand plain text files where each file consists of the plain text from multiple websites along its metadata header. From now on, when we mention the "Common Crawl" corpus, we refer to this particular November 2018 snapshot.

2.4 fastText's Pipeline

In order to download, extract, filter, clean and classify Common Crawl we base ourselves on the "fastText pre-processing pipeline" used by Grave et al. (2018). Their pipeline first launches multiple process, preferably as many as available cores. Each of these processes first downloads one Common Crawl WET file which then proceeds to decompress after the download is over. After decompressing, an instance of the fastText linear classifier (Joulin et al., 2016; Joulin et al., 2017) is launched, the classifier processes each WET file line by line, generating a language tag for each

[10] https://www.gnu.org/software/gzip/

line. The tags are then stored in a tag file which holds a one-to-one correspondence between lines of the WET file and its corresponding language tag. The WET file and the tag files are read sequentially and each on the WET file line holding the condition of being longer that 100 bytes is appended to a language file containing only plain text (tags are discarded). Finally the tag file and the WET files are deleted.

Only when one of these processes finishes another can be launched. This means that one can at most process and download as many files as cores the machine has. That is, if for example a machine has 24 cores, only 24 WET files can be downloaded and processed simultaneously, moreover, the 25$^{\text{th}}$ file won't be downloaded until one of the previous 24 files is completely processed.

When all the WET files are classified, one would normally get around 160 language files, each file holding just plain text written in its corresponding language. These files still need to be filtered in order to get rid of all files containing invalid UTF-8 characters, so again a number of processes are launched, this time depending on the amount of memory of the machine. Each process reads a language file, first filters for invalid UTF-8 characters and then performs deduplication. A simple non-collision resistant hashing algorithm is used to deduplicate the files.

The fastText linear classifier works by representing sentences for classification as Bags of Words (BoW) and training a linear classifier. A weight matrix A is used as a look-up table over the words and the word representations are then averaged into a text representation which is fed to the linear classifier. The architecture is in general similar to the CBoW model of Mikolov et al. (2013) but the middle word is replaced by a label. They uses a softmax function f to compute the probability distribution over the classes. For a set of N documents, the model is trained to minimise the negative log-likelihood over the classes:

$$-\frac{1}{N}\sum_{n=1}^{N} y_n \log(f(BAx_n)),$$

where x_n is the normalised bag of features of the n-th document, y_n is the n-th label, and A, B are the weight matrices. The pre-trained fastText model for language recognition (Grave et al., 2018) is capable of recognising around 176 different languages and was trained using 400 million tokens from Wikipedia as well as sentences from the Tatoeba website[11].

2.5 Asynchronous pipeline

We propose a new pipeline derived from the fastText one which we call `goclassy`, we reuse the fastText linear classifier (Joulin et al., 2016; Joulin et al., 2017) and the

[11] https://tatoeba.org/

	10 files			100 files			200 files		
	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
real									
fastText	2m50s	6m45s	3m31s	13m46s	38m38s	17m39s	26m20s	47m48s	31m4s
goclassy	1m23s	3m12s	1m42s	7m42s	12m43s	9m8s	15m3s	15m47s	15m16s
user									
fastText	26m45s	27m2s	26m53s	4h21m	4h24m	4h23m	8h42m	8h48m	8h45m
goclassy	10m26s	12m53s	11m0s	1h46m	1h54m	1h49m	3h37m	3h40m	3h38m
sys									
fastText	40.14s	40.85s	40.56s	6m14s	6m17s	6m15s	12m26s	12m45s	12m31s
goclassy	37.34s	45.98s	39.67s	5m7s	5m34s	5m16s	9m57s	10m14s	10m5s

Table 2.1: Benchmarks are done using the UNIX `time` tool, are repeated 10 times each and are done for random samples of 10, 100 and 200 WET files. Only the classifying and filtering part are benchmarked. The table shows the minimum, maximum and mean time for the user, real and sys time over the 10 runs. Here "fastText" is used as short for the pipeline.

pre-trained fastText model for language recognition (Grave et al., 2018), but we completely rewrite and parallelise their pipeline in an asynchronous manner.

The order of operations is more or less the same as in the fastText pre-processing pipeline but instead of clustering multiple operations into a single blocking process, we launch a worker for each operation and we bound the number of possible parallel operations at a given time by the number of available threads instead of the number of CPUs. We implement goclassy using the Go programming language[12] so we let the Go runtime[13] handle the scheduling of the processes. Thus in our pipeline we don't have to wait for a whole WET file to download, decompress and classify in order to start downloading and processing the next one, a new file will start downloading and processing as soon as the scheduler is able to allocate a new process.

When using electromechanical mediums of storage, I/O blocking is one of the main problems one encounters. To overcome this, we introduced buffers in all our I/O operations, a feature that is not present in the fastText pre-processing pipeline. We also create, from the start, a file for each of the 176 languages that the pre-trained fastText language classifier is capable of recognising, and we always leave them open, as we find that getting a file descriptor to each time we want to write, if we wanted leave them open just when needed, introduces a big overhead.

We also do the filtering and cleaning processes at line level before feeding each line to the classifier, which makes us create a new filtered file so that we can have a correspondence with the tag file, which in turn will consume more space, but that will also reduce the amount of unnecessary classifications performed by fastText. The filtered and file tags are then read and lines are appended to its corresponding

[12] https://golang.org/
[13] https://golang.org/src/runtime/mprof.go

language file. The writing in the classification step is asynchronous, meaning that process writing a line to the filtered files does not wait for the classifier to write a tag on the tag file. Figure 2.1 shows the pipeline up to this point.

After all WET files are processed, we then use Isaac Whitfield's deduplication tool runiq[14] which is based on Yann Collet's xxhash64[15], an extremely fast non-cryptographic hash algorithm that is resistant to collisions. We finally use the Mark Adler's pigz[16] for data compression, as opposed to the canonical UNIX tools proposed in the original fastText pipeline. We add both tools to our concurrent pipeline, executing multiple instances of them in parallel, in order to ensure we use the most of our available resources at a given time.

Beyond improving the computational time required to classify this corpus, we propose a simple improvement on the cleaning scheme in the fastText pre-processing pipeline. This improvement allows our pipeline to better take into account the multilingual nature of Common Crawl; that is, we count UTF-8 characters instead of bytes for setting the lower admissible bound for the length of a line to be fed into the classifier. This straightforward modification on the fastText pre-processing pipeline assures we take into account the multiple languages present in Common Crawl that use non-ASCII encoded characters.

Given that our implementation is written in Go, we release binary distributions [17] of goclassy for all major operating systems. Both pigz and runiq are also available for all major operating systems.

2.6 Benchmarks

We test both pipelines against one another in an infrastructure using traditional electromechanical storage mediums that are connected to the main processing machine via an Ethernet interface, that is, a low I/O speed environment as compared to an infrastructure where one would have an array of SSDs connected directly to the main processing machine via a high speed interface. We use a machine with an Intel® Xeon® Processor E5-2650 2.00 GHz, 20M Cache, and 203.1 GiB of RAM. We make sure that no other processes apart from the benchmark and the Linux system processes are run. We do not include downloading, decompression or deduplication in our benchmarks as downloading takes far too much time, and deduplication and compression were performed with third party tools that don't make part of our main contribution. We are mainly interested in seeing how the way the data is fed to the classifier impacts the overall processing time.

[14] https://github.com/whitfin/runiq
[15] https://github.com/Cyan4973/xxHash
[16] https://zlib.net/pigz/
[17] https://github.com/pjox/goclassy

Benchmarks in table 2.1 of our goclassy pipeline show a drastic reduction in processing time compared to the original fastText prepossessing pipeline. We show that in our particular infrastructure, we are capable of reducing the *real* time as measured by the `time` UNIX tool almost always by half. The *user* time which represents the amount of CPU time spent in user-mode code (outside the kernel) within the process is almost three times lower for our goclassy pipeline, this particular benchmark strongly suggest a substantial reduction in energy consumption of goclassy with respect to the fastText pipeline.

As we understand that even an infrastructure with more than 20TB of free space in traditional electromechanical storage is not available to everyone and we propose a simple parametrization in our pipeline that actively deletes already processed data and that only downloads and decompresses files when needed, thus ensuring that no more than 10TB of storage are used at a given time. We nevertheless note that delaying decompression increases the amount of computation time, which is a trade-off that some users might make as it might be more suitable for their available infrastructure.

2.7 OSCAR

Finally, we are aware that some users might not even have access to a big enough infrastructure to run our pipelines or just to store all the Common Crawl data. Moreover, even if previously used and cited in NLP and Machine Learning research, we note that there is currently no public distribution of Common Crawl that is filtered, classified by language and ready to use for Machine Learning or NLP applications. Thus we decide to publish a pre-processed version of the November 2018 copy of Common Crawl which is comprised of usable data in 166 different languages, we publish[18] our version under the name OSCAR which is short for *Open Super-large Crawled ALMAnaCH*[19] *coRpus*.

After processing all the data with goclassy, the size of the whole Common Crawl corpus 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 such as Chinese and Japanese do not use spaces. The sizes in bytes for both the original and the deduplicated versions of OSCAR can be found in table 2.2. OSCAR is published under the *Creative Commons CC0 license* (*"no rights reserved"*)[20], so it is free to use for all applications.

[18] https://team.inria.fr/almanach/oscar/
[19] https://team.inria.fr/almanach/
[20] http://creativecommons.org/publicdomain/zero/1.0/

2.8 Conclusions

We are sure that our work will greatly benefit researchers working on an either constrain infrastructure or a low budget setting. We are also confident, that by publishing a classified version of Common Crawl, we will substantially increase the amount of available public data for medium to low resource languages, thus improving and facilitating NLP research for them. Furthermore, as our pipeline speeds-up and simplifies the treatment of Common Crawl, we believe that our contribution can be further parallelised and adapted to treat multiple snapshots of Common Crawl opening the door to what would be otherwise costly diachronic studies of the use of a given language throughout the internet.

Finally, we note that both our proposed pipeline is data independent, which means that they can be reused to process, clean and classify any sort of big multilingual corpus that is available in plain text form and that is UTF-8 encoded; meaning that the impact of our work goes way beyond a single corpus.

Language	Size		Words		Language	Size		Words	
	Orig	Dedup	Orig	Dedup		Orig	Dedup	Orig	Dedup
Afrikaans	241M	163M	43,482,801	29,533,437	Lower Sorbian	13K	7.1K	1,787	966
Albanian	2.3G	1.2G	374,196,110	186,856,699	Luxembourgish	29M	21M	4,403,577	3,087,650
Amharic	360M	206M	28,301,601	16,086,628	Macedonian	2.1G	1.2G	189,289,873	102,849,595
Arabic	82G	32G	8,117,162,828	3,171,221,354	Maithili	317K	11K	69,161	874
Aragonese	1.3M	801K	52,896	45,669	Malagasy	21M	13M	3,068,360	1,872,044
Armenian	3.7G	1.5G	273,919,388	110,196,043	Malay	111M	42M	16,696,882	6,045,753
Assamese	113M	71M	6,956,663	4,366,570	Malayalam	4.9G	2.5G	189,534,472	95,892,551
Asturian	2.4M	2.0M	381,005	325,237	Maltese	24M	17M	2,995,654	2,163,358
Avaric	409K	324K	24,720	19,478	Marathi	2.7G	1.4G	162,609,404	82,130,803
Azerbaijani	2.8G	1.5G	322,641,710	167,742,296	Mazanderani	691K	602K	73,870	64,481
Bashkir	128M	90M	9,796,764	6,922,589	Minangkabau	608K	310K	5,682	4,825
Basque	848M	342M	120,456,652	45,359,710	Mingrelian	5.8M	4.4M	299,098	228,629
Bavarian	503	503	399	399	Mirandese	1.2K	1.1K	171	152
Belarusian	1.8G	1.1G	144,579,630	83,499,037	Modern Greek	62G	27G	5,479,180,137	2,412,419,435
Bengali	11G	5.8G	623,575,733	363,766,143	Mongolian	2.2G	838M	181,307,167	68,362,013
Bihari	110K	34K	8,848	2,875	Nahuatl languages	12K	11K	1,234	1,193
Bishnupriya	4.1M	1.7M	198,286	96,940	Neapolitan	17K	13K	5,282	4,147
Bosnian	447K	116K	106,448	20,485	Nepali	1.8G	1.2G	107,448,208	71,628,317
Breton	29M	16M	5,013,241	2,890,384	Newari	5.5M	4.1M	564,697	288,995
Bulgarian	32G	14G	2,947,648,106	1,268,114,977	Northern Frisian	4.4K	4.4K	1,516	1,516
Burmese	1.9G	1.1G	56,111,184	30,102,173	Northern Luri	76K	63K	8,022	6,740
Catalan	8.0G	4.3G	1,360,212,450	729,333,440	Norwegian	8.0G	4.7G	1,344,326,388	804,894,377
Cebuano	39M	24M	6,603,567	3,675,024	Norwegian Nynorsk	85M	54M	14,764,980	9,435,139
Central Bikol	885	885	312	312	Occitan	5.8M	3.7M	750,301	512,678
Central Khmer	1.1G	581M	20,690,610	10,082,245	Oriya	248M	188M	14,938,567	11,321,740
Central Kurdish	487M	226M	48,478,334	18,726,721	Ossetian	13M	11M	1,031,268	878,765
Chavacano	520	520	130	130	Pampanga	760	304	130	52
Chechen	8.3M	6.7M	711,051	568,146	Panjabi	763M	460M	61,847,806	37,555,835
Chinese	508G	249G	14,986,424,850	6,350,215,113	Persian	79G	38G	9,096,554,121	4,363,505,319
Chuvash	39M	26M	3,041,614	2,054,810	Piemontese	2.1M	1.9M	362,013	337,246
Cornish	44K	14K	8,329	2,704	Polish	109G	47G	15,277,255,137	6,708,709,674
Croatian	226M	110M	34,232,765	16,727,640	Portuguese	124G	64G	20,641,903,898	10,751,156,918
Czech	53G	24G	7,715,977,441	3,540,997,509	Pushto	361M	242M	46,559,441	31,347,348
Danish	16G	9.5G	2,637,463,889	1,620,091,317	Quechua	78K	67K	10,186	8,691
Dhivehi	126M	79M	7,559,472	4,726,660	Romanian	25G	11G	3,984,317,058	1,741,794,069
Dimli	146	146	19	19	Romansh	7.4K	6.5K	1,093	960
Dutch	78G	39G	13,020,136,373	6,598,786,137	Russia Buriat	13K	11K	963	809
Eastern Mari	7.2M	6.0M	565,992	469,297	Russian	1.2T	568G	92,522,407,837	46,692,691,520
Egyptian Arabic	66M	33M	7,305,151	3,659,419	Sanskrit	93M	37M	4,331,569	1,713,930
Emilian-Romagnol	25K	24K	6,376	6,121	Scottish Gaelic	1.9M	1.3M	310,689	207,110
English	2.3T	1.2T	418,187,793,408	215,841,256,971	Serbian	3.9G	2.2G	364,395,411	207,561,168
Erzya	1.4K	1.2K	90	78	Serbo-Croatian	25M	5.8M	5,292,184	1,040,573
Esperanto	299M	228M	48,486,161	37,324,446	Sicilian	3.3K	2.8K	554	468
Estonian	4.8G	2.3G	643,163,730	309,931,463	Sindhi	347M	263M	43,530,158	33,028,015
Finnish	27G	13G	3,196,666,419	1,597,855,468	Sinhala	1.4G	802M	93,053,465	50,864,857
French	282G	138G	46,896,036,417	23,206,776,649	Slovak	9.1G	4.5G	1,322,247,763	656,346,179
Galician	620M	384M	102,011,291	63,600,602	Slovenian	2.5G	1.3G	387,399,700	193,926,684
Georgian	3.6G	1.9G	171,950,621	91,569,739	Somali	61K	16K	1,202	472
German	308G	145G	44,878,908,446	21,529,164,172	South Azerbaijani	27M	19M	2,175,054	1,528,709
Goan Konkani	2.2M	1.8M	124,277	102,306	Spanish	278G	149G	47,545,122,279	25,928,290,729
Guarani	36K	24K	7,382	4,680	Sundanese	211K	141K	30,321	20,278
Gujarati	1.1G	722M	72,045,701	50,023,432	Swahili	13M	8.1M	2,211,927	1,376,963
Haitian	3.9K	3.3K	1,014	832	Swedish	44G	25G	7,155,994,312	4,106,120,608
Hebrew	20G	9.8G	2,067,753,528	1,032,018,056	Tagalog	573M	407M	98,949,299	70,121,601
Hindi	17G	8.9G	1,372,234,782	745,774,934	Tajik	379M	249M	31,758,142	21,029,893
Hungarian	40G	18G	5,163,936,345	2,339,127,555	Tamil	9.3G	5.1G	420,537,132	226,013,330
Icelandic	1.5G	846M	219,900,094	129,818,331	Tatar	670M	305M	51,034,893	23,825,695
Ido	147K	130K	25,702	22,773	Telugu	2.5G	1.6G	123,711,517	79,094,167
Iloko	874K	636K	142,942	105,564	Thai	36G	16G	951,743,087	368,965,202
Indonesian	30G	16G	4,574,692,265	2,394,957,629	Tibetan	187M	138M	1,483,589	936,556
Interlingua	662K	360K	180,231	100,019	Tosk Albanian	5.0M	2.8M	841,750	459,001
Interlingue	24K	1.6K	5,352	602	Turkish	60G	27G	7,577,388,700	3,365,734,289
Irish	88M	60M	14,483,593	10,017,303	Turkmen	11M	6.8M	1,113,869	752,326
Italian	137G	69G	22,248,707,341	11,250,012,896	Tuvinian	12K	7.9K	759	540
Japanese	216G	106G	4,962,979,182	1,123,067,063	Uighur	122M	83M	8,657,141	5,852,225
Javanese	659K	583K	104,896	86,654	Ukrainian	53G	28G	4,204,381,276	2,252,380,351
Kalmyk	113K	112K	10,277	10,155	Upper Sorbian	4.2M	1.8M	545,351	236,867
Kannada	1.7G	1.1G	81,186,863	49,343,462	Urdu	2.7G	1.7G	331,817,982	218,030,228
Karachay-Balkar	2.6M	2.3M	185,436	166,496	Uzbek	21M	12M	2,450,256	1,381,644
Kazakh	2.7G	1.5G	191,126,469	108,388,743	Venetian	18K	17K	3,492	3,199
Kirghiz	600M	388M	44,194,823	28,982,620	Vietnamese	68G	32G	12,036,845,359	5,577,159,843
Komi	2.3M	1.2M	201,404	95,243	Volapük	2.0M	2.0M	321,121	318,568
Korean	24G	12G	2,368,765,142	1,120,375,149	Walloon	273K	203K	50,720	37,543
Kurdish	94M	60M	15,561,003	9,946,440	Waray	2.5M	2.2M	397,315	336,311
Lao	174M	114M	4,133,311	2,583,342	Welsh	213M	133M	37,422,441	23,574,673
Latin	26M	8.3M	4,122,201	1,328,038	Western Frisian	35M	26M	5,691,077	4,223,816
Latvian	4.0G	1.8G	520,761,977	236,428,905	Western Mari	1.2M	1.1M	93,338	87,780
Lezghian	3.3M	3.0M	247,646	224,871	Western Panjabi	12M	9.0M	1,426,986	1,111,112
Limburgan	29K	27K	4,730	4,283	Wu Chinese	109K	32K	11,189	4,333
Lithuanian	8.8G	3.9G	1,159,661,742	516,183,525	Yakut	42M	26M	2,547,623	1,789,174
Lojban	736K	678K	154,330	141,973	Yiddish	141M	84M	13,834,320	8,212,970
Lombard	443K	433K	75,229	73,665	Yoruba	55K	27K	8,906	3,518
Low German	18M	13M	2,906,347	2,146,417	Yue Chinese	3.7K	2.2K	186	128
Total	6.3T	3.2T	844,315,434,723	425,651,344,234					

Table 2.2: Size of the OSCAR corpus by language measured in bytes and number of words. Standard UNIX human-readable notation is used for the size in byte. We define "words" as spaced separated tokens, which gives a good estimate of the size of each corpus for languages using Latin or Cyrillic alphabets, but might give a misleading size for other languages such as Chinese or Japanese.

3 Quality at Glance

3.1 Introduction

Access to multilingual datasets for NLP research has vastly improved over the past years. A variety of web-derived collections for hundreds of languages is available for anyone to download, such as ParaCrawl (Esplà et al., 2019; Bañón et al., 2020), WikiMatrix (Schwenk et al., 2021) CCAligned (El-Kishky et al., 2020), OSCAR (Ortiz Suárez et al., 2019; Ortiz Suárez et al., 2020), and several others. These have in turn enabled a variety of highly multilingual models, like mT5 (Xue et al., 2021), M2M-100 (Fan et al., 2020), M4 (Arivazhagan et al., 2019).

Curating such datasets relies on the websites giving clues about the language of their contents (e.g. a language identifier in the URL) and on automatic language classification (LangID). It is commonly known that these automatically crawled and filtered datasets tend to have overall lower quality than hand-curated collections (Koehn et al., 2020), but their quality is rarely measured directly, and is rather judged through the improvements they bring to downstream applications (Schwenk et al., 2021).

Building NLP technologies with automatically crawled datasets is promising. This is especially true for low-resource languages, because data scarcity is one of the major bottlenecks for deep learning approaches. However, there is a problem: There exists very little research on evaluating both data collections and automatic crawling and filtering tools for low-resource languages. As a result, although many low-resource languages are covered by the latest multilingual crawl data releases, their quality and thus usability is unknown.

To shed light on the quality of data crawls for the lowest resource languages, we perform a manual data audit for 230 per-language subsets of five major crawled multilingual datasets: CCAligned (El-Kishky et al., 2020), ParaCrawl (Esplà et al., 2019; Bañón et al., 2020), WikiMatrix (Schwenk et al., 2021), OSCAR (Ortiz Suárez et al., 2019; Ortiz Suárez et al., 2020) and mC4 (Xue et al., 2021). We propose solutions for effective, low-effort data auditing (section 3.4), including an error taxonomy. Our quantitative analysis reveals surprisingly low amounts of valid in-language data, and identifies systematic issues across datasets and languages. In addition, we find that a large number of datasets is labeled with nontransparent or incorrect language codes (section 3.5). This leads us to reflect on the potential harm of low-quality data releases for low-resource languages (section 3.6), and provide a set of recommendations for future multilingual data releases (section 3.8).

3.2 Related Work

Corpora collected by web crawlers are known to be noisy (Junczys-Dowmunt, 2019). In highly multilingual settings, past work found that web-crawls of lower-resource languages have serious issues, especially with segment-level LangID (Caswell et al.,

	Parallel			Monolingual	
	CCAligned	**ParaCrawl v7.1**	**WikiMatrix**	**OSCAR**	**mC4**
#languages	137	41	85	166	101
Source	CC 2013–2020	selected websites	Wikipedia	CC 11/2018	CC all
Filtering level	document	sentence	sentence	document	document
Langid	FastText	CLD2	FastText	FastText	CLD3
Alignment	LASER	Vec/Hun/BLEU-Align	LASER	-	-
Evaluation	TED-6	WMT-5	TED-45	POS/DEP-5	XTREME

Table 3.1: Comparison of parallel and monolingual corpora extracted from web documents, including their downstream evaluation tasks. All parallel corpora are evaluated through machine translation evaluation with BLEU. TED-6: `da`, `cr`, `sl`, `sk`, `lt`, `et`; TED-45: 45-language subset of (Qi et al., 2018); WMT-5: `cs`, `de`, `fi`, `lv`, `ro`. POS/DEP-5: part-of-speech labeling and dependency parsing for `bg`, `ca`, `da`, `fi`, `id`.

2020). Cleaning and filtering web-crawls can boost general language modeling (Gao et al., 2020; Brown et al., 2020; Raffel et al., 2020) and downstream task performance (Moore and Lewis, 2010; Xu and Koehn, 2017; Khayrallah and Koehn, 2018; Brown et al., 2020).

As the scale of ML research grows, it becomes increasingly difficult to validate automatically collected and curated datasets (Biderman and Scheirer, 2020; Birhane and Prabhu, 2021; Bender et al., 2021). Several works have focused on advancing methodologies and best practices to address these challenges. Bender and Friedman (2018) introduced data statements, a documentary framework for NLP datasets that seeks to provide a universal minimum bar for dataset description. Similar work has focused on online news (Kevin et al., 2018), data ethics (Sun et al., 2019), and data exploration (Holland et al., 2018), as well as generalist work such as (Gebru et al., 2018). There is a large literature on filtering text data for various NLP tasks, e.g. (Axelrod et al., 2011; Moore and Lewis, 2010; Wang et al., 2018; Kamholz et al., 2014; Junczys-Dowmunt, 2018; Caswell et al., 2020).

Closest to our work is the analysis of a highly multilingual (non-publicly available) web-crawl and LangID related quality issues by Caswell et al. (2020). They perform a brief analysis of the quality of OSCAR with the focus only on the presence of in-language content.

3.3 Multilingual Corpora

Table 3.1 provides an overview of the corpora of interest in this work. We selected the corpora for their multilinguality and the inclusion of understudied languages in

NLP. With the exception of WikiMatrix and Paracrawl, all corpora are derived from CommonCrawl (CC).[1]

CCAligned (El-Kishky et al., 2020) is a parallel dataset built off 68 CC snapshots. Documents are aligned if they are in the same language according to FastText LangID (Joulin et al., 2016; Joulin et al., 2017), and have the same URL but for a differing language code. These alignments are refined with cross-lingual LASER embeddings (Artetxe and Schwenk, 2019). For sentence-level data, they split on newlines and align with LASER, but perform no further filtering. Human annotators evaluated the quality of document alignments for six languages (`de`, `zh`, `ar`, `ro`, `et`, `my`) selected for their different scripts and amount of retrieved documents, reporting precision of over 90%. The quality of the extracted parallel sentences was evaluated in a machine translation (MT) task on six European (`da`, `cr`, `sl`, `sk`, `lt`, `et`) languages of the TED corpus (Qi et al., 2018), where it compared favorably to systems built on crawled sentences from WikiMatrix and ParaCrawl v6.

Multilingual C4 (mC4) (Xue et al., 2021) is a document-level dataset used for training the mT5 language model. It consists of monolingual text in 101 languages and is generated from 71 CC snapshots. It filters out pages that contain less than three lines of at least 200 characters and pages that contain bad words (Emerick, 2018). Since this is a document-level dataset, we split it by sentence and deduplicate it before rating. For language identification, it uses CLD3 (Botha et al., 2017), a small feed-forward neural network that was trained to detect 107 languages. The mT5 language model pre-trained on mC4 is evaluated on 6 tasks of the XTREME benchmark (Hu et al., 2020) covering a variety of languages and outperforms other multilingual pre-trained language models such as mBERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020).

OSCAR (Ortiz Suárez et al., 2019; Ortiz Suárez et al., 2020) is a set of monolingual corpora extracted from CC snapshots, specifically from the plain text *WET* format distributed by CC which removes all the HTML tags and converts the text formatting to UTF-8. It is deduplicated and follows the same approach as Grave et al. (2018) by using FastText LangID on a line-level. For five languages (`bg`, `ca`, `da`, `fi`, `id`) OSCAR corpora were used to train ELMo (Peters et al., 2018) embeddings for POS tagging and dependency parsing, outperforming those trained on Wikipedia (Ortiz Suárez et al., 2020).

ParaCrawl v7.1 is a parallel dataset with 41 language pairs primarily aligned with English (39 out of 41) and mined using the parallel-data-crawling tool Bitextor

[1] `http://commoncrawl.org/`

(Esplà et al., 2019; Bañón et al., 2020) which includes downloading documents, preprocessing and normalization, aligning documents and segments, and filtering noisy data via Bicleaner. ParaCrawl focuses on European languages, but also includes 9 lower-resource, non-European language pairs in v7.1. Sentence alignment and sentence pair filtering choices were optimized for five languages (`mt`, `et`, `hu`, `cs`, `de`) by training and evaluating MT models on the resulting parallel sentences. An earlier version, ParaCrawl v5, was shown to improve translation quality on WMT benchmarks for `cs`, `de`, `fi`, `lv`, `ro`.

WikiMatrix (Schwenk et al., 2021) is a public dataset containing 135M parallel sentences in 1620 language pairs (85 languages) mined from Wikipedia. Out of the 135M parallel sentences, 34M are aligned with English. The text is extracted from Wikipedia pages, split into sentences, and duplicate sentences are removed. FastText LangID is used before identifying bitext with LASER's distance-based mining approach. The margin threshold is optimized by training and evaluating downstream MT models on four WMT benchmarks (`de-en`, `de-fr`, `cs-de`, `cs-fr`). The final dataset is evaluated through TED translation models between 45 languages, with highest quality for translations between English and e.g. `pt`, `es`, `da`, and lowest for `sr`, `ja`, `mr`, `zh_TW`. We focus on language pairs with English on one side.

3.4 Auditing Data Quality

None of the above datasets has been evaluated for quality on the sentence level (exception: several languages in ParaCrawl v3), and downstream evaluations are centered around a small fraction of higher-resource languages. This is insufficient for drawing conclusions about the quality of individual or aligned sentences, and about the entirety of languages. To close this gap, we conduct a data quality audit that focuses on the lowest-resource and most under-evaluated languages, but also covers mid- and high-resource languages for comparison.

3.4.1 Auditing Process

Participants We recruited 51 volunteers from the NLP community, covering about 70 languages with proficient language skills. To verify our hypothesis that those annotations can largely done by non-native speakers, we repeat a set of language expert annotations by a non-expert, and measure the accuracy of the non-expert.

Sample selection For each language in each dataset, we took a random sample of 100 lines, which may be anywhere from single words to short paragraphs depending on segmentation.We manually annotated them according to the error taxonomy described below. For WikiMatrix and CCAligned, we selected those languages that

Correct Codes	
`CC`: *Correct translation, natural sentence*	
`en` The Constitution of South Africa	`nso` Molaotheo wa Rephabliki ya Afrika Borwa
`en` Transforming your swimming pool into a pond	`de` Umbau Ihres Swimmingpools zum Teich
`CB`: *Correct translation, Boilerplate or low quality*	
`en` Reference number: 13634	`ln` Motango ya référence: 13634
`en` Latest Smell Stop Articles	`fil` Pinakabagong mga Artikulo Smell Stop
`CS`: *Correct translation, Short*	
`en` movies, dad	`it` cinema, papà
`en` Halloween - without me	`ay` Hallowen – janiw nayampejj
Error Codes	
X: *Incorrect translation, but both correct languages*	
`en` A map of the arrondissements of Paris	`kg` Paris kele mbanza ya kimfumu ya Fwalansa.
`en` Ask a question	`tr` Soru sor Kullanıma göre seçim
`WL`: *Source OR target wrong language, but both still linguistic content*	
`en` The ISO3 language code is zho	`zza` Táim eadra bracach mar bhionns na frogannaidhe.
`en` Der Werwolf — sprach der gute Mann,	`de` des Weswolfs, Genitiv sodann,
NL: *Not a language: at least one of source and target are not linguistic content*	
`en` EntryScan 4 _	`tn` TSA PM704 _
`en` organic peanut butter	`ckb` ������

Table 3.2: Annotation codes for parallel data with sentence pair examples. The language code before each sentence indicates the language it is supposed to be in.

are paired with English, and for ParaCrawl, we also included those paired with Spanish ("total" counts in table 3.3). We did not annotate all languages, but focused on the ones with the least number of sentences in each dataset (at least the smallest 10) and languages for which we found proficient speakers.

Non-expert labeling strategies Although many of the volunteers were familiar with the languages in question or spoke related languages, in cases where no speaker of a relevant language could be found, volunteers used dictionaries and internet search to form educated guesses. We discuss this deeper in appendix 3.14 to highlight how much of this low-resource focused evaluation can actually be done by non-proficient speakers with relatively low effort. In general, we aim to find an upper bound on quality, so we encouraged annotators to be forgiving of translation mistakes when the overall meaning of the sentence or large parts thereof are conveyed, or when most of the sentence is in the correct language.

Taxonomy In order to quantify errors, we developed a simple error taxonomy. Sentences and sentence pairs were annotated according to a simple rubric with error classes of Incorrect Translation (`X`, excluded for monolingual data), Wrong Language (`WL`), and Non-Linguistic Content (`NL`). Of correct sentences (`C`), we further mark single words or phrases (`CS`) and boilerplate contents (`CB`). The appendix contains the

detailed instructions, and table 3.2 provides examples for parallel data. In addition, we asked annotators to flag offensive or pornographic content.

		Parallel			Monolingual	
		CCAligned	**ParaCrawl v7.1**	**WikiMatrix**	**OSCAR**	**mC4**
	#langs audited / total	65 / 119	21 / 38	20 / 78	51 / 166	48 / 108
	%langs audited	54.62%	55.26%	25.64%	30.72%	44.44%
	#sents audited / total	8037 / 907M	2214 / 521M	1997 / 95M	3517 / 8.4B	5314 / 8.5B
	%sents audited	0.00089%	0.00043%	0.00211%	0.00004%	0.00006%
macro	C	29.25%	76.14%	23.74%	87.21%	72.40%
	X	29.46%	19.17%	68.18%	-	-
	WL	9.44%	3.43%	6.08%	6.26%	15.98%
	NL	31.42%	1.13%	1.60%	6.54%	11.40%
	offensive	0.01%	0.00%	0.00%	0.14%	0.06%
	porn	5.30%	0.63%	0.00%	0.48%	0.36%
micro	C	53.52%	83.00%	50.58%	98.72%	92.66%
	X	32.25%	15.27%	47.10%	-	-
	WL	3.60%	1.04%	1.35%	0.52%	2.33%
	NL	10.53%	0.69%	0.94%	0.75%	5.01%
	offensive	0.00%	0.00%	0.00%	0.18%	0.03%
	porn	2.86%	0.33%	0.00%	1.63%	0.08%
	#langs =0% C	7	0	1	7	0
	#langs <50% C	44	4	19	11	9
	#langs >50% NL	13	0	0	7	1
	#langs >50% WL	1	0	0	3	4

Table 3.3: Averages of sentence-level annotations across datasets and selected languages. Macro-avg: Each language is weighted equally in the aggregation, regardless of its size. Micro-avg: Each label is weighted by the fraction of sentences for that language in the overall annotated corpus, i.e., the annotations for higher-represented languages are upweighted, and annotations for lower-represented languages are downweighted. The bottom rows contain the number of languages that have 0% sentences labeled C etc.

3.4.2 Human Audit Results

Interpretation of Results For each language, we compute the percentage of each label within the 100 audited sentences. Then, we either aggregate the labels across languages with equal weights (macro-average), or weight them according to their presence in the overall dataset (micro-average). Note that the number of languages, the numbers of sentences per language and the choice of languages differ across datasets, both in the original release and in the selection for our audit, so the comparison of numbers across datasets has to be taken with a grain of salt. Our audit captures a decent ratio of languages (25–55%, 2nd row in table 3.3), but only a

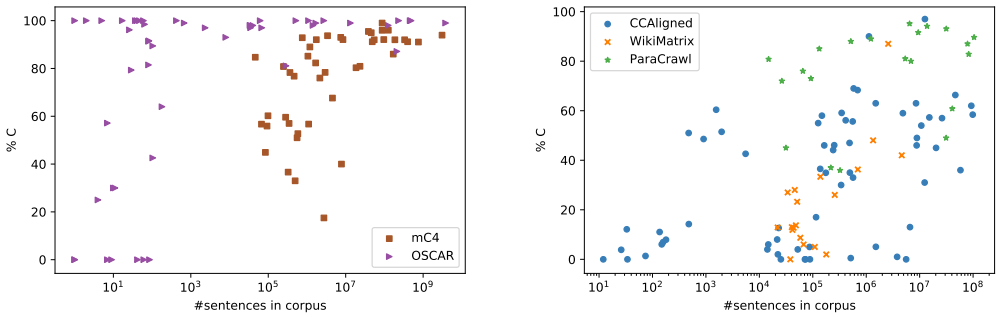
tiny fraction of the overall number of sentences (0.00004–0.002%). Appendix 3.16 contains the detailed audit results for each language and dataset. When we speak of "low-" and "high"-resource languages, we mean languages with smaller or larger representation in the datasets at hand. When reporting language-specific results we use the original language identifiers of the datasets.

WHICH DATASETS HAVE QUALITY ISSUES? The macro-averaged results show that the ratio of correct samples ("C") ranges from 24% to 87%, with a large variance across the five audited datasets. Particularly severe problems were found in CCAligned and WikiMatrix, with 44 of the 65 languages that we audited for CCAligned containing under 50% correct sentences, and 19 of the 20 in WikiMatrix. In total, 15 of the 205 language specific samples (7.3%) contained not a single correct sentence. For the parallel datasets we are also interested in the quantity of misaligned/mistranslated sentences (X). For WikiMatrix, two-thirds of the audited samples were on average misaligned. We noticed that sentences were often similar in structure, but described different facts (see table 3.10).

While Table 3.3 gives means and numbers of corpora passing certain thresholds, Figure 3.2 illustrates per-corpus correctness more completely, showing for each dataset what percent of audited corpora are under each possible threshold of correctness.

WHY HAVEN'T THESE PROBLEMS BEEN REPORTED BEFORE? The findings above are averaged on a per-language basis (i.e. macro-average), and therefore give low and high-resource languages equal weight. If we instead estimate the quality on a per-sentence basis, i.e. down-weight the the lower-resource languages in the computation of the average, the numbers paint a more optimistic picture ("micro" block in table 3.3). This is especially relevant for the monolingual datasets because they contain audits for English, which makes up for 43% of all sentences in OSCAR and 36% in mC4. To illustrate the effect of this imbalance: A random sample from the entire mC4 dataset will with over 63% chance be from one of the 8 largest languages (`en`, `ru`, `es`, `de`, `fr`, `it`, `pt`, `pl`, >100M sentences each),of which all have near perfect quality. Analogously, evaluation and tuning of web mining pipelines and resulting corpora in downstream applications focused largely on higher-resource languages (section 3.3), so the low quality of underrepresented languages might go unnoticed if there is no dedicated evaluation, or no proficient speakers are involved in the curation (Nekoto et al., 2020).

HOW MUCH CONTENT IS NONLINGUISTIC OR IN THE WRONG LANGUAGE? In general, non-linguistic content was a larger problem than wrong-language content. Among the parallel datasets, CCAligned contains the highest percentage of nonlinguistic content, at 31.42% on average across all rated corpora, and also the highest percent of



(a) Monolingual corpora (b) Parallel corpora

Figure 3.1: Percentage of sentences labeled as correct vs. log N sentences for all audited languages.

wrong-language content, at 9.44% on average. Among the monolingual datasets, mC4 contains the highest ratio both of sentences in incorrect languages (15.98% average) and nonlinguistic content (11.40% average), with 4 of the 48 audited languages having more than 50% contents in other languages. The low amount of wrong language in ParaCrawl shows the benefits of selecting domains by the amount in-language text, but the dataset also covers the smallest amount of languages. The relatively low ratio of wrong language samples in OSCAR may reflect the success of line-level LangID filtering. These numbers provide evidence that more research in improved LangID could improve the overall quality, especially with respect to nonlinguistic content.

WHICH LANGUAGES GOT CONFUSED? The languages that were confused were frequently related higher-resource languages. However, there were also a significant number of "out-of-model cousin" cases, where languages not supported by the LangID model ended up in a similar-seeming language. For instance in mC4, much of the Shona (`sn`) corpus is actually Kinyarwanda (`rw`) – and, peculiarly, much of the Hawaiian (`haw`) is actually Twi (`tw/ak`).

DO LOW-RESOURCE LANGUAGES HAVE LOWER QUALITY? Low-resource datasets tend to have lower human-judged quality. The Spearman rank correlation between quality (%C) and size is positive in all cases. The trend is strongest for mC4 ($r = 0.66$), and gradually declines for CCAligned ($r = 0.53$), WikiMatrix ($r = 0.49$), ParaCrawl ($r = 0.43$), and OSCAR ($r = 0.37$). Figure 3.1 compares the number of sentences for each language against the proportion of correct sentences that we found during the audit: Not all high-resource languages have high quality, in particular for CCAligned (e.g. `en-jv_ID` with 5%C, or `en-tl_XX` with 13%C). For mid-resource languages (10^4–10^6 sentences) the picture is rather inconclusive.

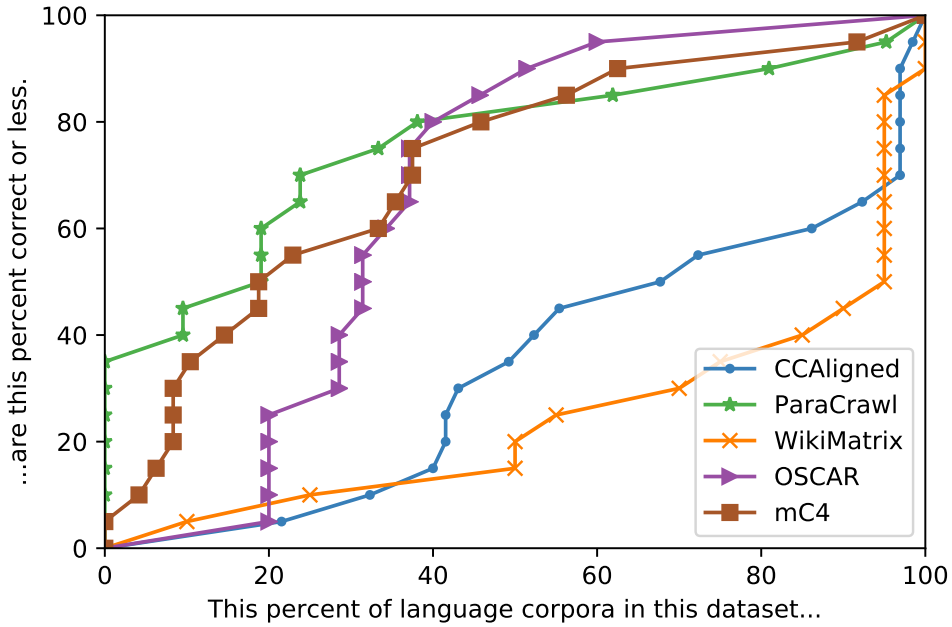


Figure 3.2: Fraction of languages in each dataset below a given quality threshold (percent correct).

WHICH LANGUAGES HAVE THE LOWEST QUALITY? Across datasets we observe that the quality is particularly poor for languages that are included in the datasets in romanized script, but are more commonly written in other scripts, e.g., Urdu (`ur`), Hindi (`hi`), Arabic (`ar`). In terms of geography, the poorest quality is found for African languages (`bm`, `ff`, `kg`, `lg`, `ln`, `nso`, `om`, `sn`, `so`, `tn`, `wo`), minority languages in Europe and the Middle East that are closely related to higher-resource languages (`az-IR`, `frr`, `nap`, `szl`, `zza`), lesser spoken Chinese languages sharing a script with Mandarin (`yue`, `wuu`), and four major Austronesian languages (`bcl`, `cbk`, `jv`, `su`).

WHAT IS THE INCIDENCE OF OFFENSIVE AND PORNOGRAPHIC CONTENT? Overall, the sampled sentences did not contain a large amount of offensive contents. However, there were notable amounts of pornographic content ($> 10\%$) found in CCAligned for 11 languages.

ANNOTATION QUALITY For six audited languages from OSCAR and ten from CCAligned we measure the accuracy of the labels assigned by non-proficient speakers against the labels assigned by proficient speakers for all audited sentences. With the full 6-class taxonomy we find a mean accuracy of 0.66 for CCAligned audits, and 0.98 for OSCAR audits (see appendix 3.11 for language-specific results). With a binary

taxonomy distinguishing C from the rest, the accuracy further increases to 0.79 for CCAligned. This provides strong evidence that good quality annotations are not limited to those proficient in a language.

3.4.3 Automatic Filtering

Given the frequency of WL and NL annotations, it might be tempting to use open-source LangID models to post-filter data on a per-sentence(-pair) level, as OSCAR does. Unfortunately, this turns out to have its own issues.

Sentence-level n-gram filtering We classify all sentence pairs of CCAligned with CLD3. By comparing its predictions to the audit labels, we evaluate its quality on the subset of annotated samples: the classifier should detect both correct languages when the pair is annotated as C and X, and should detect incorrect languages in the pair when WL and NL. On this task, the CLD3 classifier achieves an average precision of only 40.6%.

Transformer-based LangID filtering N-gram LangID models like CLD3 have known problems. However, Caswell et al. (2020) demonstrate that semi-supervised transformer-based LangID models strongly out-perform them. We train a comparable transformer-based LangID model and apply it to our annotated CCAligned data. We find that filtering noisy corpora ($< 50\%$ correct) on LangID for both source and target leads to gains in median precision, rising from 13.8% pre-filter to 43.9% post-filter. However, this comes at a steep cost of 77.5% loss in recall. The biggest winners were Lingala, whose precision climbs from 8% to 80%, and Oromo, which soars from 2% to 33% in-language. Both of these, however, come at the cost of losing 50% of the correct in-language sentences. The moral is that, at least at the current stage, there is no one-size-fits-all approach for sentence-level LangID.

3.5 Dataset Mis-labeling

Standardized and unambiguous representations of language codes are important for practical data use and exchange. The standard for unambiguous language codes used by most academic and industry applications is BCP-47 (Phillips and Davis, 2005), which would enhance transparency and interoperability if adopted consistently.

We find a variety of errors in language code usage, ranging from serious mislabelings to small transgressions against standard conventions. . For this analysis, we also include the JW300 (Agić and Vulić, 2019) dataset, a multilingual dataset crawled from jw.org. In summary, we find 8 nonstandard codes in CCAligned, 3 in OSCAR, 1 in mC4, 1 in WikiMatrix, and 70 in JW300, for 83 in total. This does not include the 59 codes affected by superset issues. Full details are given in appendix 3.9.

Inconsistent Language Codes One common issue is simply using nonstandard or invented codes. For example, CCAligned uses only two-letter codes, so when the BCP-47 code for a language is three letters it is either shortened (e.g. `zza` → `zz`) or invented (`shn` → `qa`) Similarly, OSCAR contains data labeled as `als` (BCP-47 for Tosk Albanian) that is actually in `gsw` (Allemanic). 22 additional language codes in JW300 have similar issues, including 12 codes that start with `jw_` but are not Javanese.

False sign languages 12% (48/417) of JW300 carry language codes for sign languages. Instead of sign language transcripts they are texts in another high resource language, mostly English or Spanish—for example, the `en-zsl` data is actually English-English parallel data (i.e. copies).

Mysterious supersets When datasets contain language codes that are supersets of other language codes, it is difficult to determine which particular language the data are in. WikiMatrix has Serbian (`sr`), Croatian (`hr`), Bosnian (`bs`), and Serbo-Croatian (`sh`)—their superset. In some cases this may not be an issue, as with Arabic, where `ar` conventionally refers to Modern Standard Arabic, even though the code technically encompasses all dialects. But in many cases, the nature of the data in the superset code remains a mystery.

Deprecated codes Finally, there are several deprecated codes that are used: `sh` in Wikimatrix, `iw` in mC4, `sh` and `eml` in Oscar, and `daf` in JW300.

3.6 Risks of Low-Quality Data

Low quality in downstream applications Text corpora today are building blocks for many downstream NLP applications like question answering and text summarization—for instance, a common approach is to first train translation models on such data and then automatically translate training data for downstream models (Conneau et al., 2018). If the data used for the original systems is flawed, derived technology may fail for those languages far down the line without knowing the causes.

Representation washing Since there are datasets which contain many low-resource languages, the community may feel a sense of progress and growing equity, despite the actual quality of the resources for these languages. Similarly, if low-quality datasets are used as benchmarks they may exaggerate model performance, making low-resource NLP appear more solved than it is—or conversely, if models perform poorly when trained with such data, it may be wrongly assumed that the task of learning models for these languages is harder than it actually is or infeasible given current resources. These effects could result in productive effort being redirected away from these tasks and languages.

Trust in incorrect "facts" We found many instances of parallel-looking sentences that are actually not semantically similar (appendix table 3.10). They can cause models to produce plausible "translations" that are factually wrong, but users may still trust them (*algorithmic trust*) without verifying the information. Similarly, *automation bias* (Skitka et al., 1999), referring to humans favoring decisions made by automated systems over decisions made by humans, might amplify the issues of inaccurate translations caused by misaligned sentences.

3.7 Future Work

There are a variety of ways to improve both the ease and accuracy of human evaluation, as well a few classes of issues we ignored in this paper, like close dialects. We present a slightly improved suggested rubric in appendix 3.15.

Ideally there can be a standard suite of automatic metrics for datasets, but more study is necessary to determine what the appropriate metrics would be. One important area missing from our analyses however is the estimated portion of a dataset which has been generated by MT, LM systems, or bots/templates. The information captured in machine-generated content might still be useful for modeling, but might falsely overrepresent typical generation patterns and introduce linguistic errors or unnatural artifacts.

3.8 Conclusion & Recommendations

Of the five multilingual corpora evaluated, we consistently found severe issues with quality, especially in the lower-resource languages. We rated samples of 205 languages, and found that 87 of them had under 50% usable data, with a full 15 languages at 0% in-language. We furthermore found consistent issues with mislabeled data and nonstandard language codes, particularly in the JW300 dataset, and identified 83 affected corpora, at least 48 of which were entirely spurious (Section 3.5). While there might have been anecdotal evidence of insufficient quality for some of the datasets, the majority of these quality issues had not been reported, nor been investigated in depth. These issues might go unnoticed for languages that are not represented in the evaluation of the crawling methods, and cause harm in downstream applications.

We therefore strongly recommend looking at samples of any dataset before using it or releasing it to the public. As we have shown, one does not need to be proficient in a language to see when there are serious quality issues, and a quick scan of 100 (or fewer!) sentences can be sufficient to detect major problems. Moreover, going through and annotating a small sample of data can bring useful insights about new ways to filter or use it.

If data quality issues are found, a wide variety of techniques can be explored, like filtering on length-ratio, LangID, TF-IDF wordlists (Caswell et al., 2020) or dictionaries (Kamholz et al., 2014); to neural approaches like LM scoring (Axelrod et al., 2011; Moore and Lewis, 2010; Wang et al., 2018). Unfortunately, none of these provides a quick and easy fix, especially for low-resource languages – data cleaning is no trivial task!

Noisy datasets are however by no means useless, at least if they contain some usable content. Therefore an alternative to filtering can be documentation (Bender et al., 2021). This can take the form of a per-language quality score and notes about known issues, a datasheet (Gebru et al., 2018) or nutrition label (Holland et al., 2018). However, we suggest researchers not release corpora with near-zero in-language content, as this may give the mistaken impression of usable resources.

Finally, we encourage the community to continue conducting evaluations and audits of public datasets – similar to system comparison papers.

3.9 Details on Language Code Issues

Section 3.5 describes a variety of issues surrounding language codes that are unclear or incorrect. This section provides more details, focusing on the JW300 dataset.

In table 3.4 we provide a complete table of the datasets where one code is defined as a superset of the other by the ISO standard, and in table 3.5 we provide a complete list of the language codes in JW300 which purport to be sign language but are actually unrelated high-resource languages.

Special attention needs to be given to the JW300 dataset, which, in addition to the sign languages and superset code issues, has a variety of other peculiarities. These problems seem to originate in the codes used by jw.org[2], which were apparently not checked in the creation of the JW300 dataset. An overview is provided in Table 3.6, and the following paragraphs give specifics.

Twelve languages in JW300 have codes starting in `jw_`, suggesting they are varieties of Javanese (ISO639-1 `jw`), but are instead attempts to represent language dialects for which there are not BCP-47 codes. These codes seem to have been updated in jw.org to appropriate BCP-47 private-use extensions in the form `<supercode>_x_<tag>`, which are provided in Table 3.6.

In addition to the `jw_` tags, there are two other mis-used private subtags: `hy_arevmda`, which in addition to lacking the mandatory `_x_` appears to represent standard Western Armenian (`hyw`); and `rmy_AR`, which, rather than being Romany from Argentina, is Kalderash Romany.

There are also a few anomalies where private use extensions should have been used but other methods were found to convey the distinctions. Three codes appear

[2]The jw.org website seems to use correct BCP-47 extensions now, however, and entering a code such as “jw_dmr” redirects to “naq_x_dmr”

Dataset	supercode	subcode(s)
JW300	kg	kwy
JW300	mg	tdx
JW300	qu	que,qug,qus,quw,quy,quz,qvi,qvz
JW300	sw	swc
OSCAR	ar	arz
OSCAR	az	azb
OSCAR	sh	bs,hr,sr
OSCAR	ku	ckb
OSCAR	ms	id,min
OSCAR	no	nn
OSCAR	sq	als*
OSCAR	zh	yue,wuu
Wikimatrix	ar	arz
Wikimatrix	sh	bs,hr,sr
Wikimatrix	zh	wuu

Table 3.4: Situations where two language codes are represented, but one is a superset of another by the ISO standard, leading to unclarity about the data in the supercode dataset. *The `als` dataset is actually in `gsw`.

in addition to equivalent ISO codes, making it unclear which languages they are. Two of these are equivalencies between ISO639-2 and ISO639-3 (`nya` and `ny` are both Chichewa, `qu` and `que` are both Quechua). and one is a script equivalency (`kmr` and `kmr_latn` are both in Latin script). In these three cases the two codes do represent different languages — so a private use extension would have been appropriate.

Finally, there is the more minor issue that three languages use the ISO639-3 code instead of the ISO639-2 code, and therefore are not BCP-47.

In addition to the JW300-specific tables, Table 3.9 summarizes misc errors in CCAligned and OSCAR that were detailed in Section 3.5.

3.10 Complete Error Taxonomy and Instructions

In addition to the table given in table 3.2, raters were provided with the following verbal notes on the error codes

- `CC`: **Correct translation, natural sentence:** It's OK if it's a sentence fragment instead of a whole sentence, as long as it is not too short (about 5 words or greater). The translation does not have to be perfect.
- `CS`: **Correct Translation, but single word or short phrase:** Also includes highly repeated short phrases, like "the cat the cat the cat the cat the cat ..."

Actual language	Code in JW300
cs	cse
de	gsg
el	gss
en	ase,asf,bfi,ins,psp,sfs,zib,zsl
es	aed,bvl,csf,csg,csn,csr,ecs,esn, gsm,hds,lsp,mfs,ncs,prl,pys,ssp,vsl
fi	fse
fr	fcs,fsl
hu	hsh
id	inl
it	ise
ja	jsl
ko	kvk
pl	pso
pt	bzs,mzy,psr,sgn_AO
ro	rms
ru	rsl
sk	svk
sq	sql
st	jw_ssa
zh	csl,tss

Table 3.5: There are 48 languages in the JW300 corpus with language codes that correspond to sign languages, but in reality are unrelated high-resource languages (usually the most spoken language in the country of origin of the sign language). This table shows the actual language of the data corresponding to each sign language code.

- `CB`**: Correct translation, but boilerplate:** This can be auto-generated or formulaic content, or content that one deems "technically correct but generally not very useful to NLP models". Unfortunately, it's often not clear what should be counted as boilerplate...do your best.

- `X`**: Incorrect translation** [for parallel sentences] both source and target are in the correct language, but they are not adequate translations.

- `WL`**: Wrong language** For short sentences, especially with proper nouns, there is often a fine line between "Wrong language" and "Not language". Do your best.

- `NL`**: Not language** At least one of source and target are not linguistic content. Any sentence consisting only of a proper noun (e.g. "Tyrone Ping") should be marked as `NL`.

- `U:` **Unknown** for sentences that need verification by a native speaker. This is an auxiliary label that is resolved in most cases.

Finally, for future work please consider using the aspirational error taxonomy in appendix 3.15, rather than the one presented above.

Code in JW300	BCP-47 code	Actual Language Name
Incorrect private-use extensions		
hy_arevmda	hyw	Western Armenian
jw_dgr	os_x_dgr	Digor Ossetian
jw_dmr	naq_x_dmr	Damara Khoekhoe
jw_ibi	yom_x_ibi	Ibinda Kongo
jw_paa	pap_x_paa	Papiamento (Aruba)
jw_qcs	qxl	Salasaca Highland Kichwa
jw_rmg	rmn_x_rmg	Greek Romani (South)
jw_rmv	rmy_x_rmv	Vlax Romani, Russia
jw_spl	nso_x_spl	Sepulana
jw_ssa	st_ZA	Sesotho (South Africa)
jw_tpo	pt_PT	Portuguese (Portugal)
jw_vlc	ca_x_vlc	Catalan (Valencia)
jw_vz	skg_x_vz	Vezo Malagasy
rmy_AR	rmy_x_?	Kalderash
Equivalent codes used in place of extensions		
kmr_latn	kmr_x_rdu	Kurmanji (Caucasus)
nya	ny_x_?	Chinyanja (Zambia)
que	qu_x_?	Quechua (Ancash)
Deprecated codes		
daf	dnj/lda	Dan
ISO-693-3 used in place of ISO-693-2		
cat	ca	Catalan
gug	gn	Guarani
run	rn	Kirundi
tso_MZ	ts_MZ	Changana (Mozambique)

Table 3.6: Language code issues in the JW300 datasets for 22 language varieties not covered by Tables 3.4 and 3.5. Twelve languages have codes starting in `jw_`, suggesting they are varieties of Javanese, but are instead mis-parsed private-use extensions. Three codes appear in addition to equivalent ISO codes, making it unclear which languages they are. One language uses a deprecated ISO code. Four languages use the ISO639-3 code instead of the ISO639-2 code, and therefore are not BCP-47. (Note: in this table, private use extensions are given as they appear in jw.org, and specified as '?' if they are absent from jw.org.)

	es_XX	bm_ML	yo_NG	tr_TR	ku_TR	zh_CN	af_ZA	jv_ID	zh_TW	it_IT	**mean**
Acc-6	0.58	0.73	0.41	0.45	0.43	0.55	0.65	0.55	0.46	0.55	0.66
Acc-4	0.77	0.73	0.60	0.55	0.56	0.72	0.72	0.57	0.58	0.66	0.72
Acc-2	0.91	0.96	0.72	0.64	0.71	0.79	0.77	0.92	0.81	0.69	0.79

Table 3.7: Rater evaluation for a subset of audits from **CCAligned** (translated from English) measured by the accuracy (Acc-n) of labels assigned by non-proficient speaker against those assigned by proficient speakers. n indicates the granularity of the classes. For $n = 6$ all classes of the taxonomy were distinguished, for $n = 4$ the `C` subclasses were combined, and for $n = 2$ it is binary decision between `C` and the rest of the error classes.

	tyv	rm	bar	eml	zh	la	**mean**
Acc-6	1.0	0.98	1.0	1.0	0.86	1.0	0.98
Acc-4	1.0	1.0	1.0	1.0	0.87	1.0	0.98
Acc-2	1.0	1.0	1.0	1.0	0.87	1.0	0.98

Table 3.8: Rater evaluation for a subset of audits from **OSCAR** measured by the accuracy (Acc-n) of labels assigned by non-proficient speaker against those assigned by proficient speakers. n indicates the granularity of the classes. For $n = 6$ all classes of the taxonomy were distinguished, for $n = 4$ the C subclasses were combined, and for $n = 2$ it is binary decision between C and the rest of the error classes.

corpus	code in corpus	correct code
CCAligned	zz	zza
CCAligned	sz	szl
CCAligned	ns	nso
CCAligned	cb	ckb
CCAligned	tz	ber
CCAligned	qa	shn
CCAligned	qd	kac
CCAligned	cx	ceb
mC4	iw	he
OSCAR	eml	egl
OSCAR	als	gsw
OSCAR	sh	hbs
Wikimatrix	sh	hbs

Table 3.9: Miscellaneous errors in language codes not in other tables (mentioned in the text in Section 3.5).

3.11 Non-proficient Rater Evaluation

Tables 3.7 and 3.8 show the detailed rating accuracy scores for all selected languages for several levels of annotation granularity. We can see that for the CCAligned data, reducing the labels to a binary scale naturally increases the accuracy (except for tr_TR), so a binary interpretation ("correct" sentence vs. error) is the most reliable. For monolingual data, the accuracy appears exceptionally high since the bar and tyv corpora contain < 100 sentences each (4 and 25, respectively).

en	The prime minister of the **UK** is **Boris Johnson**.
nl	De minister-president van **Nederland** is **Mark Rutte**.
en	**24 March** 2018
pt	**14 Novembro** 2018
en	The current local time in **Sarasota** is **89** minutes.
nn	Den lokale tiden i **Miami** er **86** minutt.
en	In **1932** the highway was extended **north to LA**.
bar	**1938** is de Autobahn bei **Inglstod** fertig gstellt.

Table 3.10: Examples of "parallel" data where the translation has a different meaning than the source, but the form looks the same. Such data may encourage hallucinations of fake "facts".

3.12 Not-So-Parallel Data

Table 3.10 contains a list of examples from the audited datasets that were misaligned (`X`). These examples in particular illustrate that structurally similar sentences can easily describe very different facts. Translation models trained on such examples might hallucinate such fact-altering translations.

3.13 Quality vs Size

To understand the relation between the amount of data available for each language in each corpus and the quality as estimated by our audit, we plot the ratio of `X`, `NL` and `WL` labels against the number of sentences in figures 3.3, 3.4, 3.5, 3.6.

3.14 Methodological Notes

A surprising amount of work can be done without being an expert in the languages involved. The easiest approach is simply to search the internet for the sentence, which usually results in finding the exact page the sentence came from, which in turn frequently contains clues like language codes in the URL, or a headline like *News in X language*, sometimes with references to a translated version of the same page. However, for the cases where this is insufficient, here are a few tips, tricks, and observations.

No Skills Required

Things that do not require knowledge of the language(s) in question.

1. "Not language" can usually be identified by anyone who can read the script, though there are tricky cases with proper nouns.

2. Frequently, "parallel" sentences contain different numbers in the source and target (especially autogenerated content), and are easy to disqualify

3. Errors tend to repeat. If a word is mistranslated once, it will often be mistranslated many more times throughout a corpus, making it easy to spot

Basic Research Required

Things that do not require knowledge of the language(s) in question but can be done with basic research.

1. If it's written in the wrong script it's considered wrong language. (Sometimes the writing system is indicated in the published corpus, e.g. `bg-Latn`, but usually the language has a "default" script defined by ISO.)

2. Some types of texts come with inherent labels or markers, such as enumerators or verse numbers.

3. When all else fails, search the internet for the whole sentence or n-grams thereof! If the whole sentence can be found, frequently the language is betrayed by the webpage (the language's autonym is useful in this case).

3.15 Aspirational Error Taxonomy

Although the error taxonomy used in this paper did the job, there are a variety of ways to improve both the ease and accuracy of human evaluation, as well as the ease of automatically detecting issues and fixing them. With respect to improved annotations, the error taxonomy presented in this paper lacks at least one significant category of error, namely "correct/in-language but unnatural". Similarly, the definition of "correct-short" and "correct-boilerplate" were not understood equally by all annotators, leading us to collapse the categories into one for most analyses. Similarly, a concept like "correct-short" has potential issues for agglutinative languages like Turkish. Finally, it was unclear what to do with related dialects, e.g. when a sentence is "almost correct but wrong dialect" or when it is unclear which dialect a sentence belongs to.

Therefore, we present here a slightly modified version which we hope is both more explicit and finer-grained. The main changes are 1) replacing "CB" and "CS" with a catch-all for lower-quality sentences "CL", and 2) incorporating two codes for languages with related dialects. This is also by no means a perfect rubric, and would benefit from some fine-tuning and workshopping based on the particular dataset or application in question.

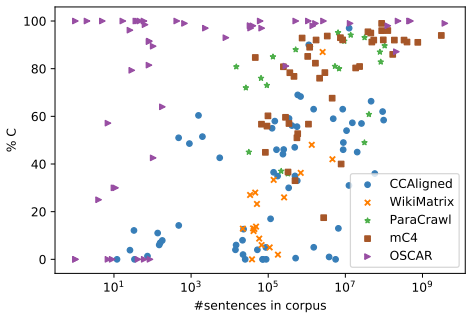


Figure 3.3: Ratio of `C` ratings vs size.

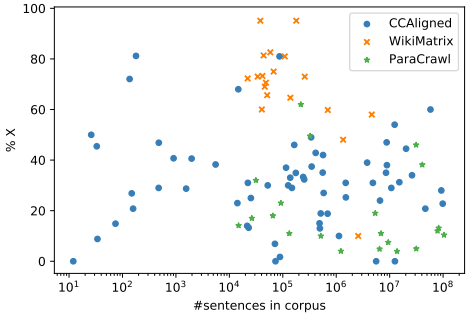


Figure 3.4: Ratio of `X` ratings vs size.

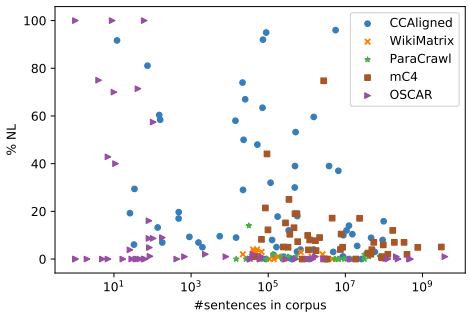


Figure 3.5: Ratio of `NL` ratings vs size.

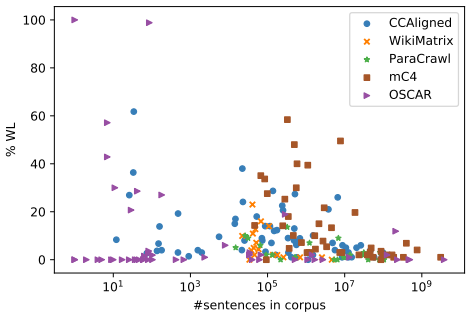


Figure 3.6: Ratio of `WL` ratings vs size.

- `CC`: **Correct:** Natural in-language sentence. It's ok if it has a few small issues, like spelling errors or a few words from another language, or if it's a sentence fragment of reasonable length (about 5 words or more). For translations, there may be minor mistakes in the translation.

- `CL`: **Correct Low-quality:** In-language sentence, but low-quality. This could be ungrammatical text, boilerplate, or very short fragments. For translations, this is the appropriate code for a low-quality translation.

- `X`: **Incorrect translation** [for parallel sentences] both source and target are in the correct language, but they are not adequate translations.

- `DW`: **Wrong Dialect** *This code is only applicable for dialects that are closely related to other languages/dialects.* This sentence is in a related but different dialect to the language it's supposed to be in. For instance, it's supposed to be in Sa'idi Arabic but it's in Egyptian Arabic.

- `DA`: **Ambiguous Dialect** *This code is only applicable for dialects that are closely related to other languages/dialects.* Correct but ambiguous whether it's in the correct language. For instance, many short sentences in Gulf Arabic may also be valid in MSA, and many written Cantonese sentences might also be valid in Mandarin.
- `WL`: **Wrong language** This sentence is not in the language it's supposed to be. For short sentences, especially with proper nouns, there is often a fine line between "Wrong language" and "Not language". Do your best.
- `NL`: **Not language** At least one of source and target are not linguistic content. Any sentence consisting only of a proper noun (e.g. "Ibuprofin", "Calvin Klein", or "Washington DC") should be marked as `NL`
- `U`: **Unknown** for sentences that need verification by a native speaker. This is an auxiliary label that is resolved in most cases.

Special note on Boilerplate: "Boilerplate" generally refers to autogenerated text found on websites. It's not always clear when a sentence is boilerplate or not. If you see a lot of similar formulaic sentences in the sample, however, that's a good sign that they are boilerplate, and you can mark them all as "CL" ! Common types of boilerplate include sentences like "Convert Euro to Pound", "Online gambling games", "Download Game of Thrones Free Torrent" and so on.

Special note on Mixed Language: Some samples are mixed between the right language and some other language. Some out-of-language content is fine, but a majority out-of-language content is not. We can mark the sentence "CC" if: 1) it is majority in-language, and 2) the in-language portion is more than a short phrase. For unclear border cases you can use "CL".

3.16 Complete Tables

Tables 3.11, 3.14, 3.15, 3.13, and 3.12 give the complete annotation percentages for CCAligned, MC4, OSCAR, Paracrawl, and Wikimatrix, respectively.

	C	CC	CS	CB	X	WL	NL	porn	#sentences	avg target length
en-sz_PL	0.00%	0.00%	0.00%	0.00%	0.00%	8.33%	91.67%	0.00%	12	71.42
en-mt_MT	3.85%	0.00%	3.85%	0.00%	50.00%	26.92%	19.23%	0.00%	26	12.58
en-tz_MA	12.12%	6.06%	6.06%	0.00%	45.45%	36.36%	6.06%	0.00%	33	57.33
en-zz_TR	0.00%	0.00%	0.00%	0.00%	8.82%	61.76%	29.41%	0.00%	34	46.53
en-kg_AO	1.35%	0.00%	1.35%	0.00%	14.86%	2.70%	81.08%	0.00%	74	29.20
en-qa_MM	11.03%	5.88%	3.68%	1.47%	72.06%	3.68%	13.24%	0.00%	136	55.28
en-bm_ML	6.04%	4.03%	2.01%	0.00%	26.85%	6.71%	60.40%	0.00%	149	32.19
en-az_IR	6.93%	6.93%	0.00%	0.00%	20.79%	13.86%	58.42%	0.00%	158	115.85
en-qd_MM	7.92%	4.95%	1.98%	0.99%	81.19%	3.96%	6.93%	0.00%	179	60.34
en-ay_BO	51.00%	33.00%	18.00%	0.00%	29.00%	3.00%	17.00%	0.00%	475	92.19
en-ak_GH	14.23%	13.60%	0.63%	0.00%	46.86%	19.25%	19.67%	0.00%	478	45.85
en-st_ZA	48.57%	42.14%	0.00%	6.43%	40.71%	1.43%	9.29%	0.00%	904	111.83
en-ve_ZA	60.40%	29.70%	21.78%	8.91%	28.71%	3.96%	6.93%	0.00%	1555	82.99
en-ts_ZA	51.49%	34.65%	11.88%	4.95%	40.59%	2.97%	4.95%	0.00%	1967	73.93
en-or_IN	42.61%	6.09%	24.35%	12.17%	38.26%	9.57%	9.57%	0.00%	5526	71.39
en-ns_ZA	4.00%	2.00%	0.00%	2.00%	23.00%	15.00%	58.00%	4.00%	14138	33.52
en-lg_UG	6.00%	0.00%	6.00%	0.00%	68.00%	17.00%	9.00%	2.00%	14701	15.83
en-ln_CD	8.00%	4.00%	3.00%	1.00%	14.00%	4.00%	74.00%	4.00%	21562	28.80
en-om_KE	2.00%	2.00%	0.00%	0.00%	31.00%	38.00%	29.00%	24.00%	22206	23.83
en-ss_SZ	12.65%	9.04%	3.61%	0.00%	13.25%	24.10%	50.00%	13.86%	22960	25.30
en-te_IN_rom	0.00%	0.00%	0.00%	0.00%	25.00%	8.00%	67.00%	5.00%	25272	24.21
en-cb_IQ	4.00%	1.00%	3.00%	0.00%	30.00%	18.00%	48.00%	11.00%	52297	30.04
en-tn_BW	0.00%	0.00%	0.00%	0.00%	6.90%	8.97%	63.45%	10.34%	71253	16.80
en-ff_NG	0.00%	0.00%	0.00%	0.00%	0.00%	8.00%	92.00%	2.00%	73022	33.59
en-sn_ZW	5.00%	1.00%	3.00%	1.00%	81.00%	14.00%	0.00%	0.00%	86868	102.59
en-wo_SN	0.00%	0.00%	0.00%	0.00%	1.71%	3.31%	94.98%	18.46%	88441	27.25
en-br_FR	17.00%	3.00%	1.00%	13.00%	37.00%	14.00%	32.00%	1.00%	115128	41.68
en-zu_ZA	55.00%	39.00%	3.00%	13.00%	30.00%	7.00%	8.00%	3.00%	126101	79.32
en-ku_TR	36.52%	12.17%	13.04%	11.30%	33.04%	28.70%	1.74%	1.74%	137874	90.51
en-ig_NG	58.00%	49.00%	3.00%	6.00%	29.00%	12.00%	1.00%	0.00%	148146	83.42
en-kn_IN	46.00%	9.00%	6.00%	31.00%	46.00%	2.00%	5.00%	4.00%	163921	70.20
en-yo_NG	34.93%	6.16%	10.96%	17.81%	34.93%	12.33%	17.81%	0.00%	175192	75.01
en-ky_KG	44.12%	24.51%	17.65%	1.96%	33.33%	22.55%	0.00%	0.98%	240657	69.56
en-tg_TJ	46.08%	18.63%	24.51%	2.94%	32.35%	20.59%	0.98%	4.90%	251865	75.31
en-ha_NG	30.00%	25.00%	3.00%	2.00%	49.00%	9.00%	12.00%	1.00%	339176	60.78
en-am_ET	59.11%	35.47%	2.46%	21.18%	37.44%	2.96%	0.49%	0.00%	346517	58.29
en-km_KH	56.12%	12.24%	33.67%	10.20%	42.86%	1.02%	0.00%	0.00%	412381	71.35
en-ne_NP	47.00%	10.00%	13.00%	24.00%	15.00%	8.00%	30.00%	14.00%	487155	79.14
en-su_ID	35.00%	15.00%	15.00%	5.00%	13.00%	13.00%	39.00%	0.00%	494142	57.08
en-ur_PK_rom	0.50%	0.00%	0.50%	0.00%	18.91%	27.36%	53.23%	5.47%	513123	18.41
en-ht_HT	55.67%	8.25%	10.31%	37.11%	35.05%	6.19%	3.09%	1.03%	558167	101.95
en-mn_MN	33.00%	8.00%	14.00%	11.00%	42.00%	7.00%	18.00%	12.00%	566885	44.43
en-te_IN	69.00%	42.00%	11.00%	16.00%	27.00%	1.00%	3.00%	1.00%	581651	97.95
en-kk_KZ	68.32%	40.59%	18.81%	8.91%	18.81%	8.91%	3.96%	1.98%	689651	72.36
en-be_BY	90.00%	57.00%	13.00%	20.00%	10.00%	0.00%	0.00%	2.00%	1125772	118.45
en-af_ZA	63.00%	40.00%	23.00%	0.00%	31.00%	2.00%	4.00%	12.00%	1504061	105.45
en-jv_ID	5.05%	1.01%	1.01%	3.03%	25.25%	10.10%	59.60%	8.08%	1513974	18.34
en-nl_NL	46.00%	27.00%	19.00%	0.00%	49.00%	2.00%	3.00%	0.00%	36324231	85.95
en-hi_IN_rom	1.00%	0.00%	0.00%	1.00%	39.00%	21.00%	39.00%	8.00%	3789571	18.13
en-lv_LV	59.00%	37.00%	9.00%	13.00%	31.00%	7.00%	3.00%	14.00%	4850957	83.67
en-ar_AR_rom	0.00%	0.00%	0.00%	0.00%	0.00%	4.00%	96.00%	4.00%	5584724	16.69
en-tl_XX	13.00%	6.00%	3.00%	4.00%	24.00%	26.00%	37.00%	5.00%	6593250	37.03
en-uk_UA	63.00%	42.00%	8.00%	13.00%	35.00%	1.00%	1.00%	5.00%	8547348	67.88
en-zh_TW	46.00%	11.00%	31.00%	4.00%	47.00%	6.00%	1.00%	1.00%	8778971	24.89
en-el_GR	49.00%	15.00%	5.00%	29.00%	38.00%	3.00%	10.00%	8.00%	8878492	54.90
en-da_DK	54.00%	31.00%	18.00%	5.00%	29.00%	5.00%	12.00%	7.00%	10738582	73.99
en-vi_VN	31.00%	18.00%	0.00%	13.00%	54.00%	1.00%	14.00%	6.00%	12394379	74.19
en-sv_SE	97.00%	91.00%	3.00%	3.00%	0.00%	3.00%	0.00%	0.00%	12544075	103.91
en-zh_CN	57.29%	22.92%	12.50%	21.88%	31.25%	1.04%	10.42%	1.04%	15181410	33.55
en-tr_TR	45.00%	14.50%	14.00%	16.50%	44.50%	5.00%	5.50%	4.00%	20282339	83.80
en-ja_XX	57.00%	35.00%	21.00%	1.00%	34.00%	6.00%	0.00%	0.00%	26201214	34.44
en-pt_XX	66.34%	36.63%	10.89%	18.81%	20.79%	3.96%	8.91%	0.00%	46525410	87.20
en-it_IT	36.00%	14.00%	18.00%	4.00%	60.00%	1.00%	3.00%	0.00%	58022366	97.44
en-de_DE	62.00%	29.00%	14.00%	19.00%	28.00%	2.00%	8.00%	2.00%	92597196	78.08
en-es_XX	58.42%	16.83%	25.74%	15.84%	22.77%	2.97%	15.84%	4.95%	98351611	72.18
mean	27.01%	29.35%	8.62%	28.97%	14.48%	6.49%	5.89%	0.00%	5.26%	

Table 3.11: Audit results for a sample of 100 sentences from **CCAligned** for each language pair, compared to the number of sentences available in the dataset. If fewer than 100 sentences were available, all sentences were audited. Language codes are as originally published. The length is measured in number of characters and averaged across the audited portion of each corpus. Languages with less than 20% correct sentences are boldfaced.

	C	CC	CS	CB	X	WL	NL	porn	# sentences	avg target length
en-ug	12.87%	8.91%	1.98%	1.98%	72.28%	9.90%	1.98%	0.00%	22012	95.55
en-mwl	27.00%	26.00%	0.00%	1.00%	73.00%	0.00%	0.00%	0.00%	33899	135.26
en-tg	0.00%	0.00%	0.00%	0.00%	95.10%	3.92%	0.98%	0.00%	37975	88.87
en-ne	13.00%	7.00%	6.00%	0.00%	60.00%	23.00%	4.00%	0.00%	40549	69.26
en-ka	11.88%	2.97%	2.97%	5.94%	73.27%	10.89%	2.97%	0.00%	41638	144.74
en-lmo	12.75%	11.76%	0.00%	0.98%	81.37%	4.90%	0.98%	0.00%	43790	89.38
en-io	28.00%	27.00%	0.00%	1.00%	69.00%	2.00%	1.00%	0.00%	45999	83.26
en-jv	13.73%	9.80%	0.00%	3.92%	70.59%	12.75%	2.94%	0.00%	48301	91.87
en-wuu	23.23%	14.14%	7.07%	2.02%	65.66%	7.07%	4.04%	0.00%	51024	34.77
br-en	8.70%	7.61%	1.09%	0.00%	82.61%	4.35%	0.00%	0.00%	58400	90.68
bar-en	6.00%	6.00%	0.00%	0.00%	75.00%	16.00%	3.00%	0.00%	67394	103.51
en-kk	5.00%	2.00%	2.00%	1.00%	81.00%	14.00%	0.00%	0.00%	109074	56.03
en-sw	33.33%	27.27%	4.04%	2.02%	64.65%	2.02%	0.00%	0.00%	138590	111.61
en-nds	1.96%	1.96%	0.00%	0.00%	95.10%	1.96%	0.98%	0.00%	178533	91.95
be-en	26.00%	24.00%	2.00%	0.00%	73.00%	1.00%	0.00%	0.00%	257946	121.22
en-hi	36.27%	32.35%	0.98%	2.94%	59.80%	0.98%	2.94%	0.00%	696125	96.77
en-ko	48.04%	33.33%	2.94%	11.76%	48.04%	2.94%	0.98%	0.00%	1345630	55.18
en-uk	87.00%	84.00%	2.00%	1.00%	10.00%	1.00%	2.00%	0.00%	2576425	104.39
en-it	42.00%	42.00%	0.00%	0.00%	58.00%	0.00%	0.00%	0.00%	4626048	140.27
en-simple	37.62%	24.75%	0.00%	12.87%	56.44%	2.97%	2.97%	0.00%	nan	77.53

Table 3.12: Audit results for a sample of 100 sentences from **WikiMatrix** for each language pair, compared to the number of sentences available in the dataset. Language codes are as originally published. The length is measured in number of characters and averaged across the audited portion of each corpus. Languages with less than 20% correct sentences are boldfaced.

	C	CC	CS	CB	X	WL	NL	porn	# sentences	avg target length
en-so	80.81%	61.62%	1.01%	18.18%	14.14%	5.05%	0.00%	0.00%	14879	189.83
en-ps	72.00%	53.00%	9.00%	10.00%	17.00%	10.00%	0.00%	0.00%	26321	141.01
en-my	45.00%	9.00%	16.00%	20.00%	32.00%	9.00%	14.00%	0.00%	31374	147.07
en-km	76.00%	51.00%	13.00%	12.00%	18.00%	6.00%	0.00%	0.00%	65113	121.20
en-ne	73.00%	48.00%	1.00%	24.00%	23.00%	2.00%	0.00%	0.00%	92084	153.42
en-sw	85.00%	60.00%	15.00%	10.00%	11.00%	2.00%	2.00%	0.00%	132517	167.34
en-si	37.00%	31.00%	6.00%	0.00%	62.00%	0.00%	1.00%	0.00%	217407	123.06
en-nn	35.92%	24.27%	8.74%	2.91%	49.51%	13.59%	0.97%	0.00%	323519	56.24
es-eu	88.00%	66.00%	15.00%	7.00%	10.00%	1.00%	1.00%	0.00%	514610	121.31
es-gl	89.00%	46.00%	6.00%	37.00%	4.00%	7.00%	0.00%	0.00%	1222837	107.88
en-ru	81.00%	73.00%	6.00%	2.00%	19.00%	0.00%	0.00%	6.00%	5377911	101.28
en-bg	95.15%	85.44%	0.97%	8.74%	4.85%	0.00%	0.00%	0.97%	6470710	112.29
es-ca	80.00%	54.00%	19.00%	7.00%	11.00%	9.00%	0.00%	5.00%	6870183	107.21
en-el	91.59%	68.22%	0.93%	22.43%	7.48%	0.93%	0.00%	0.00%	9402646	135.66
en-pl	94.12%	76.47%	0.98%	16.67%	3.92%	1.96%	0.00%	0.98%	13744860	95.95
en-nl	49.00%	32.00%	17.00%	0.00%	46.00%	3.00%	2.00%	0.00%	31295016	95.05
en-pt	93.07%	92.08%	0.00%	0.99%	4.95%	1.98%	0.00%	0.00%	31486963	108.68
en-it	60.82%	36.08%	16.49%	8.25%	38.14%	0.00%	1.03%	0.00%	40798278	127.55
en-es	87.00%	54.00%	20.00%	13.00%	12.00%	0.00%	1.00%	0.50%	78662122	119.72
en-de	82.83%	64.65%	13.13%	5.05%	13.13%	3.03%	1.01%	0.00%	82638202	111.43
en-fr	89.62%	82.08%	4.72%	2.83%	10.38%	0.00%	0.00%	0.00%	104351522	144.20

Table 3.13: Audit results for a sample of 100 sentences from **ParaCrawl** for each language pair, compared to the number of sentences available in the dataset. Language codes are as originally published. The length is measured in number of characters and averaged across the audited portion of each corpus.

	C	CC	CS	CB	WL	NL	porn	# sentences	avg length
yo	84.69%	71.43%	2.04%	11.22%	14.29%	1.02%	0.00%	46214	117.71
st	56.70%	42.27%	14.43%	0.00%	35.05%	8.25%	0.00%	66837	132.13
haw	44.90%	34.69%	1.02%	9.18%	33.67%	21.43%	1.02%	84312	129.99
ig	55.91%	41.73%	10.24%	3.94%	0.00%	44.09%	0.79%	92909	98.03
sm	60.20%	58.16%	2.04%	0.00%	27.55%	12.24%	0.00%	98467	126.42
ha	80.81%	79.80%	1.01%	0.00%	14.14%	5.05%	2.02%	247479	155.76
su	59.60%	58.59%	1.01%	0.00%	25.25%	15.15%	2.02%	280719	107.10
sn	36.63%	32.67%	2.97%	0.99%	58.42%	4.95%	0.00%	326392	145.59
mg	57.00%	57.00%	0.00%	0.00%	18.00%	25.00%	0.00%	345040	116.23
pa	78.30%	68.87%	3.77%	5.66%	4.72%	10.38%	0.00%	363399	134.43
ga	76.77%	58.59%	6.06%	12.12%	10.10%	13.13%	0.00%	465670	147.35
co	33.00%	29.00%	2.00%	2.00%	48.00%	19.00%	0.00%	494913	195.30
zu	51.00%	48.00%	2.00%	1.00%	30.00%	19.00%	0.00%	555458	137.81
jv	52.73%	19.09%	19.09%	14.55%	40.00%	7.27%	1.82%	581528	97.96
km	92.86%	92.86%	0.00%	0.00%	7.14%	0.00%	0.00%	756612	162.57
kn	85.15%	73.27%	3.96%	7.92%	2.97%	9.90%	0.00%	1056849	105.39
fy	56.73%	50.00%	3.85%	2.88%	39.42%	3.85%	0.00%	1104359	234.25
te	89.00%	76.00%	9.00%	4.00%	3.00%	8.00%	0.00%	1188243	108.49
la	82.31%	65.38%	6.15%	10.77%	10.00%	7.69%	0.00%	1674463	67.25
be	92.04%	86.73%	2.65%	2.65%	4.42%	3.54%	0.00%	1742030	110.86
af	76.00%	76.00%	0.00%	0.00%	15.00%	9.00%	0.00%	2152243	99.52
lb	17.48%	17.48%	0.00%	0.00%	7.77%	74.76%	0.00%	2740336	481.68
ne	78.35%	77.32%	1.03%	0.00%	21.65%	0.00%	0.00%	2942785	102.88
sr	93.69%	85.59%	7.21%	0.90%	5.41%	0.00%	0.00%	3398483	131.72
gl	67.62%	57.14%	10.48%	0.00%	13.33%	17.14%	0.00%	4549465	151.45
bn	93.00%	86.00%	1.00%	6.00%	3.00%	4.00%	0.00%	7444098	92.60
mr	40.00%	35.24%	2.86%	1.90%	49.52%	10.48%	0.00%	7774331	281.94
sl	92.08%	82.18%	4.95%	4.95%	2.97%	4.95%	0.00%	8499456	149.45
hi	80.30%	76.77%	1.01%	2.53%	19.70%	0.00%	2.53%	18507273	105.54
bg	80.90%	75.88%	2.51%	2.51%	2.01%	17.09%	0.00%	23409799	93.86
uk	95.48%	81.41%	7.54%	6.53%	2.01%	2.51%	0.00%	38556465	116.79
ro	94.95%	78.79%	12.12%	4.04%	3.03%	2.02%	0.00%	45738857	130.08
sv	91.18%	84.31%	2.94%	3.92%	4.90%	3.92%	1.96%	48570979	114.45
zh	92.00%	87.00%	1.00%	4.00%	1.00%	7.00%	0.00%	54542308	94.77
ja	99.00%	89.00%	6.00%	4.00%	0.00%	1.00%	1.00%	87337884	59.94
tr	95.96%	88.89%	0.00%	7.07%	3.54%	0.51%	0.00%	87595290	152.75
nl	92.08%	85.15%	6.93%	0.00%	1.98%	5.94%	0.00%	96210458	103.67
pl	96.00%	82.00%	7.00%	7.00%	2.00%	2.00%	0.00%	126164277	170.70
pt	86.00%	79.00%	4.00%	3.00%	2.00%	12.00%	1.00%	169239084	133.51
it	92.00%	79.00%	9.00%	4.00%	1.00%	7.00%	0.00%	186404508	180.26
fr	92.00%	82.00%	7.00%	3.00%	1.00%	7.00%	0.00%	332674575	143.69
de	91.18%	77.45%	7.84%	5.88%	6.86%	1.96%	0.00%	397006993	107.71
ru	91.06%	69.11%	11.38%	10.57%	4.07%	4.88%	0.00%	755585265	109.28
en	93.94%	83.84%	8.08%	2.02%	1.01%	5.05%	0.00%	3079081989	130.97
bg_latn	9.09%	9.09%	0.00%	0.00%	51.52%	39.39%	1.01%	N/A	139.92
ja_latn	13.00%	7.00%	4.00%	2.00%	60.00%	27.00%	0.00%	N/A	218.92
ru_latn	36.45%	25.23%	10.28%	0.93%	34.58%	28.97%	0.93%	N/A	123.14
zh_latn	5.00%	4.00%	1.00%	0.00%	64.00%	31.00%	0.00%	N/A	186.84

Table 3.14: Audit results for a sample of 100 sentences from **mC4** for each language, compared to the number of sentences available in the dataset. Language codes are as originally published. The length is measured in number of characters and averaged across the audited portion of each corpus. Languages with less than 20% correct sentences are boldfaced.

	C	CC	CS	CB	WL	NL	porn	# sentences	avg length
diq	100.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1	131.00
bcl	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%	0.00%	1	623.00
cbk	0.00%	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	1	519.00
pam	100.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	2	139.00
bar	25.00%	25.00%	0.00%	0.00%	0.00%	75.00%	0.00%	4	53.50
myv	100.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	5	127.00
yue	0.00%	0.00%	0.00%	0.00%	57.14%	42.86%	0.00%	7	177.00
mwl	57.14%	57.14%	0.00%	0.00%	42.86%	0.00%	0.00%	7	141.00
frr	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%	0.00%	9	231.56
ht	30.00%	30.00%	0.00%	0.00%	0.00%	70.00%	0.00%	10	329.10
ie	30.00%	30.00%	0.00%	0.00%	30.00%	40.00%	0.00%	11	121.70
scn	100.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	17	155.59
tyv	96.15%	96.15%	0.00%	0.00%	0.00%	3.85%	0.00%	26	167.96
mai	79.31%	75.86%	0.00%	3.45%	20.69%	0.00%	0.00%	29	141.17
bxr	100.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	37	160.76
dsb	100.00%	97.56%	0.00%	2.44%	0.00%	0.00%	0.00%	41	155.15
so	0.00%	0.00%	0.00%	0.00%	28.57%	71.43%	0.00%	42	208.24
rm	100.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	47	137.66
nah	100.00%	96.67%	0.00%	3.33%	0.00%	0.00%	0.00%	60	164.53
nap	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%	0.00%	61	152.11
yo	98.46%	96.92%	0.00%	1.54%	1.54%	0.00%	0.00%	64	281.57
gn	81.48%	81.48%	0.00%	0.00%	2.47%	16.05%	0.00%	81	234.95
vec	91.36%	91.36%	0.00%	0.00%	0.00%	8.64%	0.00%	81	184.90
kw	91.57%	90.36%	0.00%	1.20%	3.61%	4.82%	0.00%	83	162.75
wuu	0.00%	0.00%	0.00%	0.00%	98.84%	1.16%	0.00%	86	157.15
eml	42.57%	42.57%	0.00%	0.00%	0.00%	57.43%	0.00%	104	177.88
bh	89.42%	21.15%	0.00%	68.27%	1.92%	8.65%	0.00%	104	137.17
min	64.00%	6.00%	0.00%	58.00%	27.00%	9.00%	0.00%	180	649.85
qu	100.00%	98.97%	0.00%	1.03%	0.00%	0.00%	0.00%	425	167.27
su	99.00%	99.00%	0.00%	0.00%	0.00%	1.00%	0.00%	676	221.00
jv	97.00%	86.00%	0.00%	11.00%	1.00%	2.00%	0.00%	2350	203.08
als	93.00%	93.00%	0.00%	0.00%	6.00%	1.00%	0.00%	7997	375.44
la	98.00%	98.00%	0.00%	0.00%	2.00%	0.00%	0.00%	33838	224.11
uz	98.00%	98.00%	0.00%	0.00%	2.00%	0.00%	0.00%	34244	369.99
nds	97.03%	95.05%	0.00%	1.98%	2.97%	0.00%	0.00%	35032	344.74
sw	98.00%	98.00%	0.00%	0.00%	0.00%	2.00%	0.00%	40066	196.70
br	100.00%	96.00%	0.00%	4.00%	0.00%	0.00%	0.00%	61941	239.56
fy	97.00%	97.00%	0.00%	0.00%	2.00%	1.00%	0.00%	67762	340.23
am	81.09%	79.10%	0.00%	1.99%	18.91%	0.00%	0.00%	287142	267.43
af	100.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	517353	339.18
eu	100.00%	98.00%	0.00%	2.00%	0.00%	0.00%	0.00%	1099498	330.93
mn	98.00%	94.00%	0.00%	4.00%	2.00%	0.00%	0.00%	1430527	309.94
te	98.99%	93.94%	1.01%	4.04%	0.00%	1.01%	1.01%	1685185	412.31
kk	100.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	2719851	318.93
ca	99.00%	91.00%	0.00%	8.00%	1.00%	0.00%	0.00%	13292843	333.38
nl	98.00%	94.00%	2.00%	2.00%	2.00%	0.00%	4.00%	126067610	305.01
it	87.13%	71.29%	1.98%	13.86%	11.88%	0.99%	1.98%	210348435	393.66
zh	100.00%	97.00%	0.00%	3.00%	0.00%	0.00%	1.00%	232673578	195.60
fr	100.00%	93.00%	0.00%	7.00%	0.00%	0.00%	5.00%	461349575	306.62
es	100.00%	94.00%	0.00%	6.00%	0.00%	0.00%	3.00%	488616724	268.07
en	99.00%	96.00%	0.00%	3.00%	0.00%	1.00%	1.00%	3809525119	364.65

Table 3.15: Audit results for a sample of 100 sentences from **OSCAR** for each language, compared to the number of sentences available in the dataset. If fewer than 100 sentences were available, all sentences were audited Language codes are as originally published. Length is measured in number of characters. Languages with less than 20% correct sentences are boldfaced.

corpus	language	C	CC	CS	CB	X	WL	NL	porn	#sentences	avg target length
CCAligned	**en-tz_MA**	12.12%	6.06%	6.06%	0.00%	45.45%	36.36%	6.06%	0.00%	33	57.33
CCAligned	**en-kg_AO**	1.35%	0.00%	1.35%	0.00%	14.86%	2.70%	81.08%	0.00%	74	29.20
CCAligned	**en-bm_ML**	6.04%	4.03%	2.01%	0.00%	26.85%	6.71%	60.40%	0.00%	149	32.19
CCAligned	**en-ak_GH**	14.23%	13.60%	0.63%	0.00%	46.86%	19.25%	19.67%	0.00%	478	45.85
CCAligned	en-st_ZA	48.57%	42.14%	0.00%	6.43%	40.71%	1.43%	9.29%	0.00%	904	111.83
CCAligned	en-ve_ZA	60.40%	29.70%	21.78%	8.91%	28.71%	3.96%	6.93%	0.00%	1555	82.99
CCAligned	en-ts_ZA	51.49%	34.65%	11.88%	4.95%	40.59%	2.97%	4.95%	0.00%	1967	73.93
CCAligned	**en-ns_ZA**	4.00%	2.00%	0.00%	2.00%	23.00%	15.00%	58.00%	4.00%	14138	33.52
CCAligned	**en-lg_UG**	6.00%	0.00%	6.00%	0.00%	68.00%	17.00%	9.00%	2.00%	14701	15.83
CCAligned	**en-ln_CD**	8.00%	4.00%	3.00%	1.00%	14.00%	4.00%	74.00%	4.00%	21562	28.80
CCAligned	**en-om_KE**	2.00%	2.00%	0.00%	0.00%	31.00%	38.00%	29.00%	24.00%	22206	23.83
CCAligned	**en-ss_SZ**	12.65%	9.04%	3.61%	0.00%	13.25%	24.10%	50.00%	13.86%	22960	25.30
CCAligned	**en-tn_BW**	0.00%	0.00%	0.00%	0.00%	6.90%	8.97%	63.45%	10.34%	71253	16.80
CCAligned	**en-ff_NG**	0.00%	0.00%	0.00%	0.00%	0.00%	8.00%	92.00%	2.00%	73022	33.59
CCAligned	**en-sn_ZW**	5.00%	1.00%	3.00%	1.00%	81.00%	14.00%	0.00%	0.00%	86868	102.59
CCAligned	**en-wo_SN**	0.00%	0.00%	0.00%	0.00%	1.71%	3.31%	94.98%	18.46%	88441	27.25
CCAligned	en-zu_ZA	55.00%	39.00%	3.00%	13.00%	30.00%	7.00%	8.00%	3.00%	126101	79.32
CCAligned	en-ig_NG	58.00%	49.00%	3.00%	6.00%	29.00%	12.00%	1.00%	0.00%	148146	83.42
CCAligned	en-yo_NG	34.93%	6.16%	10.96%	17.81%	34.93%	12.33%	17.81%	0.00%	175192	75.01
CCAligned	en-ha_NG	30.00%	25.00%	3.00%	2.00%	49.00%	9.00%	12.00%	1.00%	339176	60.78
CCAligned	en-am_ET	59.11%	35.47%	2.46%	21.18%	37.44%	2.96%	0.49%	0.00%	346517	58.29
CCAligned	en-af_ZA	63.00%	40.00%	23.00%	0.00%	31.00%	2.00%	4.00%	12.00%	1504061	105.45
CCAligned	**en-ar_AR_rom**	0.00%	0.00%	0.00%	0.00%	0.00%	4.00%	96.00%	4.00%	5584724	16.69
Wikimatrix	en-sw	33.33%	27.27%	4.04%	2.02%	64.65%	2.02%	0.00%	0.00%	138590	111.61
ParaCrawl	en-so	80.81%	61.62%	1.01%	18.18%	14.14%	5.05%	0.00%	0.00%	14879	189.83
ParaCrawl	en-sw	85.00%	60.00%	15.00%	10.00%	11.00%	2.00%	2.00%	0.00%	132517	167.34
MC4	yo	84.69%	71.43%	2.04%	11.22%	N/A	14.29%	1.02%	0.00%	46214	117.71
MC4	st	56.70%	42.27%	14.43%	0.00%	N/A	35.05%	8.25%	0.00%	66837	132.13
MC4	ig	55.91%	41.73%	10.24%	3.94%	N/A	0.00%	44.09%	0.79%	92909	98.03
MC4	ha	80.81%	79.80%	1.01%	0.00%	N/A	14.14%	5.05%	2.02%	247479	155.76
MC4	sn	36.63%	32.67%	2.97%	0.99%	N/A	58.42%	4.95%	0.00%	326392	145.59
MC4	mg	57.00%	57.00%	0.00%	0.00%	N/A	18.00%	25.00%	0.00%	345040	116.23
MC4	zu	51.00%	48.00%	2.00%	1.00%	N/A	30.00%	19.00%	0.00%	555458	137.81
MC4	af	76.00%	76.00%	0.00%	0.00%	N/A	15.00%	9.00%	0.00%	2152243	99.52
OSCAR	**so**	0.00%	0.00%	0.00%	0.00%	N/A	28.57%	71.43%	0.00%	42	208.24
OSCAR	yo	98.46%	96.92%	0.00%	1.54%	N/A	1.54%	0.00%	0.00%	64	281.57
OSCAR	sw	98.00%	98.00%	0.00%	0.00%	N/A	0.00%	2.00%	0.00%	40066	196.70
OSCAR	am	81.09%	79.10%	0.00%	1.99%	N/A	18.91%	0.00%	0.00%	287142	267.43
OSCAR	af	100.00%	100.00%	0.00%	0.00%	N/A	0.00%	0.00%	0.00%	517353	339.18

Table 3.16: Results on African languages.

4 Ungoliant

With the increasing interest in language modeling in recent years in Natural Language Processing (NLP) (Rogers et al., 2020), particularly concerning contextualized word representations[1] (Peters et al., 2018; Devlin et al., 2019), there has also been an explosion in interest for large raw corpora, as some of these latest models require almost 1TiB of raw text for pre-training (Raffel et al., 2020; Brown et al., 2020).

While most of these language models were initially trained in English (Devlin et al., 2019; Yang et al., 2019; Clark et al., 2020; Zaheer et al., 2020; Xiong et al., 2021) and consequently most of the large corpora used to pre-train them were in English, there has been a recent push to produce larger high quality corpora for other languages, namely those of Grave et al. (2018), CCNet (Wenzek et al., 2020), Multilingual C4 (mC4) (Xue et al., 2021) and OSCAR (Ortiz Suárez et al., 2019; Ortiz Suárez et al., 2020) for pre-training language models, as well as, Paracrawl (Esplà et al., 2019; Bañón et al., 2020), CCAligned (El-Kishky et al., 2020) and WikiMatrix (Schwenk et al., 2021) which are parallel corpora for training Machine Translation (MT) models. Of these, only OSCAR, Paracrawl, CCaligned and WikiMatrix are freely available and easily downloadable.

In this paper we propose a new multilingual corpus for language modeling, and for that we take inspiration in the OSCAR corpus and its pipeline goclassy[2] (Ortiz Suárez et al., 2019; Ortiz Suárez et al., 2020), but we propose a new pipeline *Ungoliant*[3] that is faster, modular, parametrizable and well-documented. We then use it to produce a new corpus similar to OSCAR, yet larger, based on recent data containing mentions of last years' events such as the COVID-19 pandemic, the 2020–2021 United States racial unrest, the Australian wildfires, the Beirut explosion and Brexit among others. Moreover, contrarily to OSCAR, our corpus retains metadata information at the document level. We release our pipeline under an Apache 2.0 open source license and we publish the corpus under a research-only use license following the licensing schemes proposed by OSCAR (Ortiz Suárez et al., 2019; Ortiz Suárez et al., 2020) and Paracrawl (Esplà et al., 2019; Bañón et al., 2020).

4.1 Limitations of the OSCAR Corpus and its Generation Pipeline

4.1.1 OSCAR

OSCAR is a multilingual corpus derived from CommonCrawl[4], a project that provides web crawl data for everyone on a periodic manner, usually each month. CommonCrawl provides data in several formats, from raw HTML source code to pure text.

[1] In which one takes a unnanotated large textual corpus in a particular language and tries to predict a missing word in order to learn a vector space representation for it.

[2] https://github.com/oscar-corpus/goclassy

[3] https://github.com/oscar-corpus/ungoliant

[4] https://commoncrawl.org

OSCAR was generated from the pure text data version (WET files) of the November 2018 crawl, distributed in the form of 56,000 *shards*, that were then filtered and classified by language (Ortiz Suárez et al., 2019; Ortiz Suárez et al., 2020). OSCAR is available through several means, and has been used in numerous projects (Ortiz Suárez et al., 2019). OSCAR's generation pipeline also suffers from numerous issues, which we plan to address simultaneously with the release of a new, more powerful, stable, and higher quality pipeline

Simply put, OSCAR is composed of single language files that contain textual data (`ta.txt` for the Tamil language, for example). However, due to the often huge sizes of these files, and subsequently the impracticality of storage and distribution, OSCAR files are split and compressed in equally sized parts.

OSCAR comes in four different versions, each suited differently for different tasks, and allows less limited ways of sharing the corpus more widely. These versions are either *unshuffled* or *shuffled* (that is, for each language, lines have been shuffled, destroying records integrity), and *non-deduplicated* or *deduplicated* (since duplicate lines account for more than half of the total data[5] generated by the pipeline). For the unshuffled versions, each language file contains paragraphs that come from the same record, and each paragraph is separated by a newline.

OSCAR is inherently linked to its generation pipeline, and as such its quality partly depends on the pipeline's quality. While OSCAR is considered to be one of the cleanest multilingual corpora available (Caswell et al., 2020; Caswell et al., 2021), several problems have been described, and the state of the publicly available code raises questions about maintenance and maintenability of the pipeline itself.

Apart from the fact that its content dates back to 2018, the current OSCAR corpus suffers from quality issues discussed in (Caswell et al., 2020; Caswell et al., 2021), including:

- **Language label mismatches and inconsistencies**, which occurs earlier in the pipeline and would be fixable downstream,
- **Representation washing** as defined by Caswell et al. (2021), whereby low resource languages, while present in the corpus, are of a significantly lower quality than higher resource languages without any quality metric available publicly.

The most recent Common Crawl dump contains 64,000 shards. Each shard is composed of numerous records, and each record holds textual content along with metadata. While CommonCrawl shards hold document-level metadata that could be useful downstream, they were discarded and do not appear in OSCAR, whereas other corpora generated from the same source include them, e.g. CCNet (Wenzek et al., 2020). This limits OSCAR users to the textual content only, whereas metadata could have been distributed along with the corpus itself.

[5] OSCAR-orig: 6.3TB, OSCAR-dedup: 3.2TB

4.1.2 goclassy

OSCAR was built using *goclassy*, a high-performance asynchronous pipeline written in Go (Ortiz Suárez et al., 2019). However, it suffers from several caveats that makes the re-generation and update of the corpus relatively complex in practice.

While goclassy's source code is easily readable thanks to the choice of an uncluttered language and a pragmatic approach, the lack of structure in both the source and the project itself makes goclassy difficult to extend and maintain.

The pipeline is not functional out-of-the-box, as the user has to provide the compressed shards from CommonCrawl, manually install fasttext (Joulin et al., 2016; Joulin et al., 2017) and create specific directories by themselves, since only partial instructions are given in the supplied README file.

goclassy also makes heavy use of I/O, as data is saved and loaded repeatedly between steps; as an example, the identification step stores language identification data and individual sentences in two files, before generating the final files (one per language). Despite these limitations, goclassy's performance is good due to Go's emphasis on easy and efficient parallelization and inherent speed. The pipeline uses clever handling of file descriptors, limiting I/O calls cost in some parts.

4.2 Building a new OSCAR-like corpus

We introduce *Ungoliant*, a new corpus generation pipeline that, like goclassy, creates a large-scale multilingual text corpus from a CommonCrawl dump. Contrarily to goclassy, Ungoliant is fully modular, better structured, and highly parametrizable; thereby allowing comparisons between several parallelization strategies. A specific effort was put in testing and documentation. Parts of Ungoliant are heavily inspired by goclassy, although it is implemented in Rust rather than in Go, which is sometimes faster.[6]

Additionally, we use Ungoliant to generate a new corpus from a recent Common Crawl dump. The new corpus includes metadata information while retaining backward compatibility with the OSCAR corpus.

4.2.1 Ungoliant

Rationale and scope

While Ungoliant is heavily inspired by goclassy, it provides a better set of tools to download, process, filter and aggregate textual and contextual data from CommonCrawl. These operations can be sequential, parallel or both, depending on contexts and performance requirements.

[6] https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/rust-go.html

Platform	#shards	goclassy	Ungoliant	Approx. speedup
Desktop	1	30s	13s	×2.3
	10	3m6s	2m12s	×1.3
	25	9m10s	5m47s	×1.5
HPC	1	40s	6s	×6.6
	25	2m40s	1m6s	×2.4
	100	7m59s	4m14s	×1.8

Table 4.1: Comparison of approximate generation times depending on platform and number of shards.

We provide both batch and streaming processing, so that the whole pipeline could be run either online, with every step running on streams of data, or offline, with every step running on tangible files, or a mix of both, using already downloaded CommonCrawl dumps but streaming the rest of the process. Moreover, we embed numerous filtering and deduplication utilities directly inside Ungoliant, making these features available for pipeline composition and post-processing.

Ungoliant features a loosely defined pipeline interface, on which we re-implement goclassy's one, while improving performance by threading more aggressively and avoiding I/O where it is not necessary: While goclassy uses intermediate files for tags and sentences, we try to keep everything in memory in order to avoid losing time loading or writing files. The Rust language provides constructs that helps us build complex abstractions and pipelines while limiting proactive file I/O or computing, since nearly all the reimplemented pipeline is built around lazy evaluation. File I/O is only used when loading shards, and when writing sentences in language files.

Through benchmarking we found that the best parallelization strategy is to use rayon[7], a work-stealing (Blumofe and Leiserson, 1999) parallel and concurrent library enabling massive parallelization. We parallelize on shard-, record- and sentence-level processing.

To evaluate Ungoliant performance, we run both goclassy and Ungoliant's implementation on 1, 10, 25 and 100 Common Crawl shards both on a middle-range laptop computer (i5-7200u, 8GB RAM, NVMe SSD) and a HPC node (Xeon 5218 (64 Threads), 180GB RAM). Results are shown in Table 4.1.

Ungoliant performs better than goclassy on all tasks, independently of the platform or number of shards processed. However, we can note that Ungoliant's speedup is higher on short tasks, which is explained by its aggressive multithreading strategy, while goclassy uses a record-scope multithreading at its finest granularity.

4.2.2 Iterating on the goclassy pipeline

CommonCrawl dumps contain metadata that hold useful information such as related records, recognized language(s), or origin URLs. Since OSCAR pipeline discards

[7] https://github.com/rayon-rs/rayon

metadata and sentences can be shuffled, we lose the ability to investigate those metadata themselves, as well as working on potentially multilingual documents, since we separate text from metadata.

The new pipeline (and the resulting new corpus schema) aims to establish a first link between textual data and metadata from CommonCrawl, while staying backward compatible with the existing OSCAR schema.

In other words, switching from the original OSCAR corpus and the newly generated one should be a drop-in operation.

Metadata extraction and linking

Our choice of keeping the corpus backward compatible with the original OSCAR introduces changes in the way the corpus is generated, namely regarding metadata: a record's body is composed of sentences that aren't guaranteed to be of the same language. Since OSCAR merges sentences from multiple records into a single file, special attention has to be paid to the metadata dispatch too.

Approaches to tackle this problem range from (1) storing all metadata in a single location to (2) having language-specific metadata files that contain the metadata for each line in the language file.

Both (1) and (2) have their strengths and weaknesses, namely:

1. Having all metadata at the same place may facilitate wide queries about whole metadata, but at a cost of a very large size (which harms both accessibility and performance).

2. Getting the metadata for a given line is fast since line numbers are synchronized, but there is repeated information and a potentially important increase in size.

We choose a hybrid approach which keeps metadata local to each language, while trying to limit the information repetition by keeping an entry by group of chunks rather than by line, where a chunk is a series of contiguous sentences that share the same language from the same document.

An overview of the pipeline can be seen in Figure 4.1, with a more precise view on record processing and metadata extraction in Figure 4.1.

Metadata are distributed via JSON-encoded files holding an ordered list of metadata entries, along with offsets (o) and paragraph lengths (l), enabling any user to get the content of a said metadata by querying for lines $(o, o + l]$ in the content file.

This approach still has drawbacks, in particular when looking for the corresponding metadata of a given sentence/paragraph, where one has to perform a search on the metadata file, or when working with multilingual documents. Another drawback is the resulting cost of potentially merging back numerous language parts: Since metadata query is offset-based, merging back metadata files implies updating those offsets.

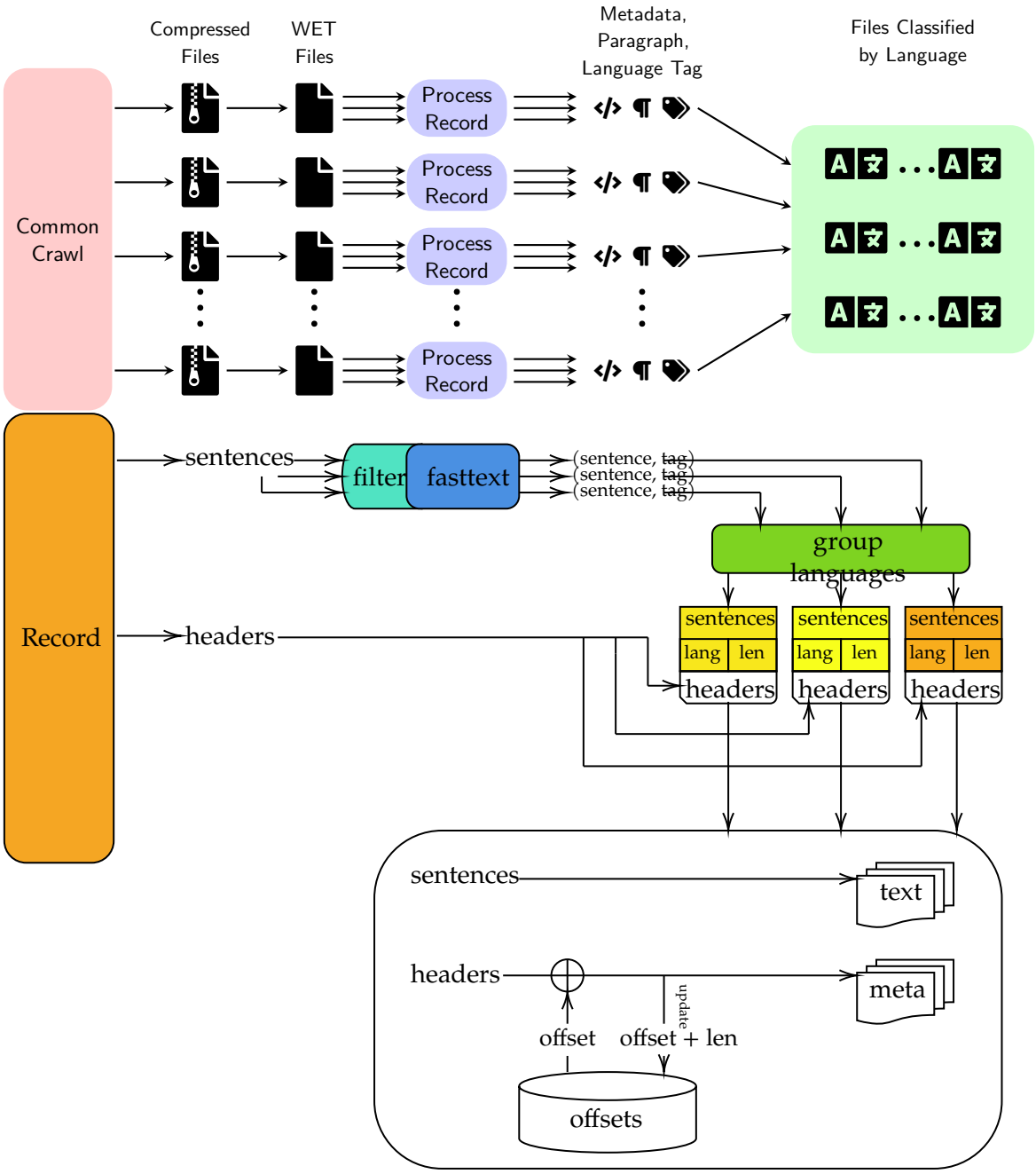


Figure 4.1: Record processing with metadata extraction. Headers are kept aside while sentences are identified and grouped into same-language bins. Headers are then cloned for each bin, and are sequentially stamped with an offset that is recorded for the whole operation, and written to disk into text and metadata files by language.

Platform	#shards	OSCAR	With Metadata	Speedup
Desktop	1	13s	12s	×1.1
	10	2m12s	1m55s	×1.1
	25	5m47s	4m50s	×1.2
HPC	1	6s	7s	×0.9
	25	1m6s	1m12s	×0.9
	100	4m14s	4m36s	×0.9

Table 4.2: Comparison of approximate generation times with and without metadata generation.

Version	Source	Textual (dedup)	Metadata	Total (increase)
2018	7.42TB	6.3TB (3.2TB)	N/A	6.3TB
2021	8.06TB	7.2TB (3.3TB)	1.2TB	8.4TB (+33%)

Table 4.3: Comparison of CommonCrawl and OSCAR sizes between 2018 and 2021 versions. Compressed (CommonCrawl) sources are from November 2018 and February 2021. Total is Textual + Metadata without deduplication.

Having paragraphs and metadata linked by offsets in a highly parallelized pipeline implies to take special care at the offset level. The solution is to use shard-scoped offsets (starting from 0 for each language), and to keep global offsets protected by a mutex guard. This way, when a given shard is done processing and is ready to be written on disk, we convert shard-scoped offsets to global-scoped ones, update the global-scoped ones and then write text and metadata on disk.

We compare running times for the reimplementation of the goclassy pipeline, and our new pipeline adding metadata extraction, using both desktop and HPC contexts. The results are reported in Table 4.2.

Metadata generation does not seem to influence generation time dramatically. However, we can notice a slight performance difference between HPC and Desktop contexts. These differences may lie in the storage medium differences, I/O layout, or algorithmic peculiarities benefiting desktop contexts because of other bottlenecks.

4.2.3 Characteristics of our new backward compatible OSCAR-like corpus

We evaluate the newly generated corpus, assessing its ability to reflect events that occurred after the publication of OSCAR 2018 and detail the metadata format and potential use.

Comparison with OSCAR

While it is expected that our new corpus has a larger file size than OSCAR since CommonCrawl itself grew from 7.42TB to 8.06TB, metadata quickly adds up and take for nearly 15% of the whole uncompressed data.

The size augmentation is not the same for each language, and while the whole corpus is bigger now, some languages are smaller than they were before.

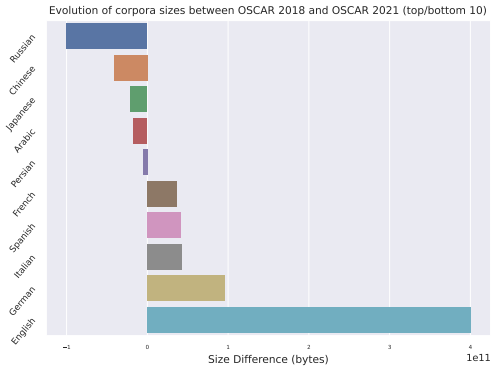


Figure 4.2: Comparison of language size (in bytes) between OSCAR 2018 and OSCAR 2021 (top/bottom 5 only).

Results show that already largely represented languages gain more and more data (like the English language, which constitutes more than a third of the original OSCAR), except for the Russian language which loses approximately 100Gb of textual content. These results are summarized in Figure 4.2.

However, in a context where the number of languages is very high (higher than 150) and of varying sizes, evolution can't be analyzed via a mere size evaluation. By computing, for each language, the relative size difference between the 2018 and 2021 releases of OSCAR, less resourced languages do appear, hinting at a better representation of some of them. These results can be found in Figure 4.3.

Numerous languages have been omitted from Figure 4.3, either:

- because they were present in the original OSCAR and are now absent (*Central Bikol* and *Cantonese*)
- because they were absent in the original OSCAR and are now present (*Manx*, *Rusyn*, *Scots* and *West Flemish*)

Precautions have to be taken when using these corpora and further work has to be done to correctly assess the quality of low-to-mid resource languages in order to better reflect the quality of each corpus to the OSCAR users. Some languages exhibited either a particularly low number of sentences or a very low quality, and as such couldn't be usable, while still accounting for a language in the total language count of the original OSCAR.

Metadata

Metadata provides new contextual data that is useful to evaluate the corpus and draw metrics.

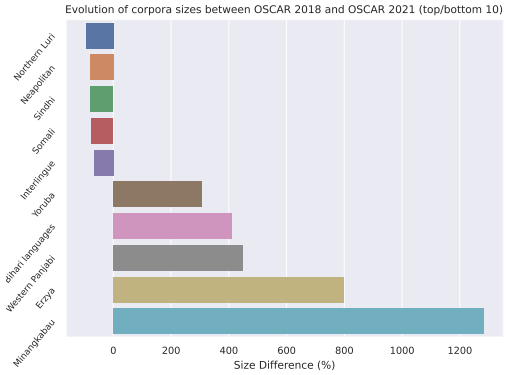


Figure 4.3: Comparison of language percentage between OSCAR 2018 and OSCAR 2021 (top/bottom 5 only).

The total size of metadata is 1.2TB, ranging from 4Kb to 500Gb, depending on the number of lines. Relative size varies from 100% to 20%, diminishing with the textual data size, which is expected.

Metadata are provided in single files for now, but split versions of both textual and contextual data will be released soon after the release of the corpus, enabling easy access.

Our choice of keeping metadata aside from the main content adds some complexity when working with both textual and contextual data:

- When trying to get the metadata of given sentence, one has to get the line number k, then sequentially (or use a search algorithm since offsets are sorted) look for the record (with offset o and length l), where $k \in [o, o + l]$.
- Looking for lines corresponding to a particular metadata entry is easier: one has to read the textual file, skipping until the o-th line, then read l lines.

Presence of events

Using a sample of an English part of our corpus, we perform a simple search of terms in order to assess and compare the presence of pre- and post- 2018 events and persons in both corpora. Terms and frequency are grouped in Table 4.4.

Our corpus keeps around the same number of occurrences for pre-2018 events or public figures such as Barack Obama, while increasing the occurrence of people linked to more recent events (Joe Biden).

We include search terms linked to post-2018 events in French and Arabic which are smaller corpora (resp. 200 and 80 GB), and in Burmese, a mid-resource language (approximately 2GB). We observe a term occurrences evolution that reflects the linked events' timing and importance.

Language	Term	2018	2021
Arabic	Beirut port explosion	0	31
Burmese*	Min Aung Hlaing	387	3439
English	Obama	30039	27639
English	Biden	990	19299
French	Yellow Vests	2	96

Table 4.4: Comparison of occurrences of news-related terms between OSCAR and our corpus in a sample of 100 CommonCrawl shards. For the Burmese language, we use the whole 2018 and 2021 corpus since it is a low resource language. Terms are translated in the corpus language.

4.2.4 License

This new corpus will be released under a research-only license that is compliant with the EU's exceptions for research in text and data mining. Contrarily to the original OSCAR, no shuffled version of the corpus will be distributed, instead we will put in place an authentication system that will allow us to verify that requests for the corpus come from research institutions. A contact form will be also provided for independent researchers so that we can study their particular cases and determine if the utilization of the corpus corresponds to a legitimate research use.

Moreover, the introduction of metadata makes our corpus far more queryable, thus simplifying and speeding up the handling of take-down GDPR requests. For this reason, we will be releasing the complete set of metadata under a CC0 public domain license, so that any individual can check if their personal or even copyrighted data is in our new corpus and make a request accordingly.

4.3 Conclusion

We show that our solution is able to generate an OSCAR-like corpus that is augmented with metadata without breaking compatibility, while being faster, better tested and thoroughly documented. We believe our new pipeline and corpus will be useful for applications in computational linguistics as well as in corpus linguistics in general.

The generated corpus is of a larger size when including metadata and without deduplication. However, deduplicated textual content is of the same magnitude between OSCAR 2018 and OSCAR 2021, while reflecting topic changes from all over the world. This fact suggests that old data may be lost with the time passing, and could be resolved by using CommonCrawl releases to build an incremental corpus, with every version augmenting the corpus size.

Metadata enables queries and statistics on the generated data, and we believe that it can be used to filter OSCAR to generate corpora that respond to certain criteria.

We plan to make this new version of OSCAR available under research constraints, with split versions of both textual content and metadata along with tools to operate on the corpus, enabling fast and easy operation on the corpus for researchers.

Part II

Data

5 CaBeRnet

5.1 Introduction

The question of quality versus size of training corpora is increasingly gaining attention and interest in the context of the latest developments in neural language models' performance. The longstanding issue of corpora "representativeness" is here addressed, in order to grasp to what extent a linguistically balanced cross-genre language sample is sufficient for a language model to gain in accuracy for contextualized word-embeddings on different NLP tasks.

Several increasingly larger corpora are nowadays compiled from the web, i.e. frWAC (Baroni et al., 2009), CCNet (Wenzek et al., 2020) and OSCAR-fr (Ortiz Suárez et al., 2019). However, does large size necessarily go along with better performance for language model training? Their alleged lack of representativeness has called for inventive ways of building a French balanced corpus offering new insights into language variation and NLP.

Following Biber's definition, "representativeness refers to the extent to which a sample includes the full range of variability in a population" (Biber, 1993). We adopt a balanced approach by sampling a wide spectrum of language use and its cross-genre variability, be it situational (e.g. format, author, addressee, purposes, settings or topics) or linguistic, e.g. linked to distributional parameters like frequencies of word classes and genres. In this way, we developed two newly built corpora. The French Balanced Reference Corpus - *CaBeRnet* - includes a wide-ranging and balanced coverage of cross-genre language use to be maximally representative of French language and therefore yield good generalizations from. The second corpus, the *French Children Book Test* (CBT-fr), includes both narrative material and oral language use as present in youth literature, and will be used for domain-specific language model training. Both are inspired by existing American and English corpora, respectively COCA, the balanced Corpus of Contemporary American English (Davies, 2009, 2010), and the Children Book Test (Hill et al., 2016, CBT).

The second main contribution of this paper lies in the evaluation of the quality of the word-embeddings obtained by pre-training and fine-tuning on different corpora, that are made here publicly available. Based on the underlying assumption that a linguistically representative corpus would possibly generate better word-embeddings. We provide an evaluation-based investigation of how a balanced cross-genre corpus can yield improvements in the performance of neural language models like ELMo (Peters et al., 2018) on various downstream tasks. The two corpora, CaBeRnet and CBT-fr, and the ELMos will be distributed freely under Creative Commons License.

Specifically, we want to investigate the contribution of oral language use as present in different corpora. Through a series of comparisons, we contrast a more domain-specific and written corpus like Wikipedia-fr with the newly built domain-specific CBT-fr corpus which additionally features oral style dialogues, like the ones one can find in youth literature. To test for the effect of corpus size, we further compare a wide ranging corpora characterized by a variety of linguistic phenomena crawled

from internet, like OSCAR (Ortiz Suárez et al., 2019), with our newly built French Balanced Reference Corpus CaBeRnet. Our aim is assess the benefits that can be gained from a balanced, multi-domain corpus such as CaBeRnet, despite its being 34 times smaller than the web-based OSCAR.

The paper is organized as follows. Sections 5.2 and 5.3 are dedicated to a descriptive overlook of the building of our two newly brewed corpora CaBeRnet and CBT-fr, including quantitative measures like type-token ratio and morphological richness. Section 5.4 presents the evaluation methods for POS-tagging, NER and dependency Parsing tasks, while results are introduced in §5.5 Finally, we conclude in §5.6 on the computational relevance of word-embeddings obtained through a balanced and representative corpus, and broaden the discussion on the benefits of smaller and noiseless corpora in neural NLP.

5.2 Corpora Building

5.2.1 CaBeRnet

CaBeRnet corpus was inspired by the genre partition of the American balanced corpus COCA, which currently contains over 618 million words of text (20 million words each year 1990-2019) and is equally divided among spoken, fiction, popular magazines, newspapers, and academic texts (Davies, 2009, 2010). A second reference, guiding our approach and sampling method, is one of the earliest precursors of balanced reference corpora: the BNC (Consortium et al., 2007), first covered a wide variety of genres, with the intention to be a representative sample of spoken and written language.

CaBeRnet was obtained by compiling existing data-sets and web-text extracted from different sources as detailed in this section. As shown in Table 5.1, genres sources are evenly divided (~120 million words each) into spoken, fiction, magazine, newspaper, academic to achieve genre-balanced between oral and written modality in newspapers or popular written style, technical reports and Wikipedia entries, fiction, literature or academic production).

CaBeRnet Oral The oral sub-portion gathers both oral transcriptions (ORFEO and Rhapsodie[1]) and Films subtitles (Open Subtitles.org), pruned from diacritics, interlocutors tagging and time stamps. To these transcriptions, the French European Parliament Proceedings (1996-2011), as presented in Koehn (2005), contributed a sample of more complex oral style with longer sentences and richer vocabulary.

[1]ORFEO corpus available at www.cocoon.huma-num.fr/exist/crdo/ ; Rhapsodie corpus at www.projet-rhapsodie.fr.

CaBeRnet Popular Press The whole sub-portion of Popular Press is gathered from an open data-set from the *Est Républicain* (1999, 2002 and 2003), a regional press format[2]. It was selected to match popular style as it is characterized by easy-to-read press style and a wide range of every-day topics characterizing local regional press.

CaBeRnet Fiction & Literature The Fiction & Literature sub-portion was compiled from march 2019's Wiki Source and WikiBooks dump and extracted using WikiExtractor.py, a script that extracts and cleans text from a WikiMedia database dumps, by performing template expansion and preprocessing of template definitions.[3]

CaBeRnet News The News sub-portion builds upon web crawled elements, including Wikimedia's NewsComments and WikiNews reports from may 2019 WikiMedia dump, collected with a custom version of WikiExtractor.py. Newspaper's content gathered by the Chambers-Rostand Corpus (i.e. Le Monde 2002-2003, La Dépèche 2002-2003, L'Humanité 2002-2003) and *Le Monde diplomatique* open-source corpus were assembled to represent a higher register of written news style from different political and thematic horizons. Several months of French Press Agency reports (AFP, 2007-2011-2012) competed with more simple and telegraphic style the newspaper written sample of the corpus.[4]

CaBeRnet Academic The academic genre was also built from different sources including technical and educational texts from WikiBooks and Wikipedia dump (prior to 2016) for their thematic variety of highly specialized written production. ORFEO Corpus offered a small sample of academic writings like PHD dissertations and scientific articles encompassing a wide choice of disciplinary topics, and TALN Corpus[5] was included to represent more concise written style characterizing scientific abstracts and proceedings.

For all sub-portions of CaBeRnet, visual inspection was performed to remove section titles, redundant meta-information linked to publishing schemes of each of the six news editor includes. This was manually achieved by compiling a rich set of regular expressions specific of each textual source to obtain clean plain text as an outcome.

[2]Corpus available at www.cnrtl.fr/corpus/estrepublicain/.
[3]Script available at https://github.com/attardi/wikiextractor.
[4]At the time being, this part of CaBeRnet corpus is still subject to Licence restrictions. This restricted amount of AFP news reports can reasonably fall in the public domain.
[5]TALN proceedings corpus (about 2 million) builds on a subset of 586 scientific articles (from 2007 to 2013), namely TALN and RECITAL. Available at redac.univ-tlse2.fr/corpus/taln_en.html.

CaBeRnet Sub-set	Tokens	Unique Forms	TTR
Oral	122 864 888	291 744	0.0024
Popular	131 444 017	458 521	0.0035
News	132 708 943	462 971	0.0035
Fiction	198 343 802	983 195	0.0050
Academic	126 431 211	1 433 663	0.0113
Total	711 792 861	2 558 513	0.0036

Table 5.1: Comparison of number of unique forms in the different genres represented by CaBeRnet partition. TTR: Type-Token Ration. Lemmatization and tokenization was performed as described in §5.3.

5.2.2 French Children Book Test (CBT-fr)

The French Children Book Test (CBT-fr) was built upon its original English version, the Children Book Test (CBT) Hill et al. (2016)[6], which consists of books freely available on www.gutenberg.orgProject Gutenberg.

Using youth literature and children books guarantees a clear narrative structure, and a large amount of dialogues, which enrich with oral register the literary style of this corpus. The English version of this corpus was originally built as benchmark data-set to test how well language models capture meaning in context. It contains 108 books, and a vocabulary size of 53,628.

French version of CBT, named CBT-fr, was constructed to guarantee enough linguistic similarities between the collected books in the two languages. 104 freely available books were included. One third of the books were purposely chosen because they were classical translations of English literary classics. Chapter heads, titles, notes and all types of editorial information were removed to obtain a plain narrative text. The effort of keeping proportion, genre, domain, and time as equal as possible yields a multilingual set of comparable corpora with a similar balance and representativeness.

5.3 Corpora Descriptive Comparison

We used two different tokenizers: SEM, Segmenteur-Étiqueteur Markovien standalone Dupont (2017) and TreeTagger. Both are based on cascades of regular expressions, and both perform tokenization and sentence splitting. The first was used

[6]This data-set can be found at www.fb.ai/babi/.

Children Book Test - fr	words
number of different lemmas	25 139
total number of forms	95 058
mean number of forms per lemma	3.78
Number of lemmas having more than one form :	14 128
Percentage of lemmas with multiple forms	56.20

Table 5.2: Lexical statistics of French CBT, performed as described in §5.3

for descriptive purposes because it technically allowed to segment and tokenize all corpora including OSCAR (23 billion words). Hence, all corpora were entirely segmented into sentences and tokenized using SEM.

The second tokenization method was run only on 3 million words samples to automatically tag them with TreeTagger into part-of-speech and lemmatize them.[7] All corpora were randomly shuffled by sentence to then select samples of 3 million words, to be able to compare them in terms of lexical composition (Type-Token Ratio, see Table 5.4).

5.3.1 Corpora Size and Composition

Length of sentences is a simple measure to quantify both sentence syntactic complexity and genre. Hence, the number of sentences reported in Table 5.3 shows interesting patterns of distributions across genres, consider the comparison between CaBeRnet an Wiki-fr. In our effort to evaluate the impact of corpora pre-training on ELMo-based contextualized word-embedding, we introduce here our two terms of comparison, namely the crawled corpus OSCAR-fr and the Wikipedia-fr one.

OSCAR fr

As it has been shown that pre-trained language models can be significantly improved by using more data (Liu et al., 2019; Raffel et al., 2020), we decided to include in our comparison a corpus of French text extracted from Common Crawl[8]. We leverage on a recently published corpus, OSCAR (Ortiz Suárez et al., 2019), which offers a pre-classified and pre-filtered version of the November 2018 Common Craw snapshot.

[7] Based on the tag-set available at https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/french-tagset.html.

[8] More information available at https://commoncrawl.org/about/.

OSCAR gathers a set of monolingual text extracted from Common Crawl - in plain text *WET* format - where all HTML tags are removed and all text encodings are converted to UTF-8. It follows a similar approach to (Grave et al., 2018) by using a language classification model based on the fastText linear classifier (Joulin et al., 2016; Joulin et al., 2017) pre-trained on Wikipedia, Tatoeba and SETimes, supporting 176 different languages.

After language classification, a deduplication step is performed without introducing a specialized filtering scheme: paragraphs containing 100 or more UTF-8 encoded characters are kept. This makes OSCAR an example of unfiltered data that is nearly as noisy as to the original Crawled data.

FrWIKI

This corpus collects a selection of pages from Wikipedia-fr from a dump executed in April 2019, where HTML tags and tables were removed, together with template expansion using Attardi's tool (WikiExtractor, §5.2.1). As reported on Table 5.3, in this data-set (660 million words) sentences are relatively longer compared to other corpora. It has the advantage of having a comparable size to CaBeRnet, but its homogeneity in terms of written genre is set to Wikipedia entries descriptive style.

CORPUS	WORDFORMS	TOKENS	SENTENCES
OSCAR-fr	23 212 459 287	27 439 082 933	1 003 261 066
Wiki-fr	665 599 545	802 283 130	21 775 351
CaBeRnet	697 119 013	830 894 133	54 216 010
CBT-fr	5 697 584	6 910 201	317 239

Table 5.3: Comparing the corpora under study.

5.3.2 Corpora Lexical Variety

Focusing on a useful measure of complexity that documents lexical richness or variety in vocabulary, we present the type-token ration (TTR) of the corpora under analysis. Generally used to asses language use aspects like the variety of different words used to communicate by learners or children, it represents the total number of unique words (types/forms) divided by the total number of tokens in a given sample of language production. Hence, the closer the TTR ratio is to 1, the greater the lexical richness of the corpus. Table 5.1 summarizes the lexical variety of the five sub-portions of CaBeRnet, respectively taken as representative of Oral, Popular, Fiction, News, and Academic genres. Domain diversity of texts can be observed in the lexical statistics showing a gradual increase in the number of distinct lexical forms (cf. TTR). This pattern reflects a generally acknowledged distributional pattern of

vocabulary-size across genres. Oral style shows a poorer lexical variety compared to newspapers/magazines' textual typology. The lexically rich fictional/classic literature is outreached by academic writing-style with its wide-ranging specialized vocabulary. All in all, Table 5.1 quantitatively demonstrates that the selected textual and oral materials are indeed representative of the five types of genres of CaBeRnet.

5.3.3 Corpora Morphological richness

To select a measure that would help quantifying the different corpora morphological richness, we follow (Bonami and Beniamine, 2015). Hence, the proportion of lemmas with multiple forms in a given vocabulary size was evaluated on randomly selected samples of 3-million-words from each corpus under analysis (see Table 5.4).

3 M samples	CBT-fr	CaBeRnet	Fr-Wiki	OSCAR
nb of diff. lemmas	25 139	30 488	31 385	31 204
tot. nb forms	95 058	180 089	238 121	190 078
mean nb forms/lemma	3.78	6.19	7.85	6.40
nb lemmas $>$ 1 form	14 128	15 927	15 182	16 480
% lemmas $>$ 1 form	56.20	52.24	48.37	52.81

Table 5.4: Lexical statistics on morphological richness over randomly selected samples of 3 million words from each corpus. nb : number

Table 4 reports some more in-depth lexical and morphological statistics across corpora. Although OSCAR is 34 times bigger than CaBeRnet, their total number of forms and the proportion of lemmas having more than one form in a 3-million-word sample are comparable. FrWiki shows a radically different lexical distribution with numerous hapaxes but a lower morphological richness. Although its total number of forms is more than one third higher than in OSCAR and CaBeRnet samples, the proportion of lemmas having more than one distinct form is around four points below CaBeRnet and OSCAR. Comparatively, youth literature in CBT-fr shows the greatest morphological richness, around 56% of lemmas have more than one form.

5.4 Corpora Evaluation Tasks

This section reports the method of experiments designed to better understand the computational impact of the quality, size and linguistic balance of ELMo's (Peters et al., 2018) pre-training (§5.4.1) and their evaluations tasks (§5.4.3).

Embeddings from Language Models ELMo is an LSTM-based language model. More precisely, it uses a bidirectional language model, which combines a both

forward and a backward LSTM-based language models. ELMo also computes a context-independent token representation via a CNN over characters. Methodologically, we selected ELMo which not only performs generally better on sequence tagging than other architectures, but which is also better suited to pre-train on small corpora because of its smaller number of parameters (93.6 million) compared to the RoBERTa-base architecture used for CamBERT (BERTbase, 12,110 million - Transformer) (Martin et al., 2020).

5.4.1 ELMo Pre-traing & Fine-tuning Method

Two protocols were carried out to evaluate the impact of corpora characteristics on the tasks under analysis. *Method 1* implies a full pre-training ELMo-based language models for each of the corpora mentioned in Table 5.3. While *Method 2* is based on pre-training OSCAR + fine-tuning with our French Balanced Reference Corpus CaBeRnet, yielding $\text{ELMo}_{\text{OSCAR+CaBeRnet}}$. Hence, the pure pre-traing (i.e. Method 1) yields the following four language models which were pre-trained on the four corpora under comparison : $\text{ELMo}_{\text{OSCAR}}$, $\text{ELMo}_{\text{Wikipedia}}$, $\text{ELMo}_{\text{CaBeRnet}}$ and ELMo_{CBT}.

5.4.2 Base evaluation systems

UDPipe Future (Straka, 2018) is an LSTM based model ranked 3$^{\text{rd}}$ in dependency parsing and 6$^{\text{th}}$ in POS tagging during the CoNLL 2018 shared task (Seker et al., 2018). We report the scores as they appear in Kondratyuk and Straka (2019)'s paper. We add to UDPipe Future, five differently trained ELMo language model pre-trained on the qualitatively and quantitatively different corpora under comparison. Additionally, we also test the impact of the CaBeRnet Corpus on ELMo fine-tuning.

The LSTM-CRF is a model originally concived by Lample et al. (2016) is just a Bi-LSTM pre-appended by both character level word embeddings and pre-trained word embeddings and pos-appended by a CRF decoder layer. For our experiments, we use the implementation of (Straková et al., 2019) which is readily available[9] and it is designed to easily pre-append contextualized word-embeddings to the model.

5.4.3 Evaluation Tasks

We distinguish three main evaluation tasks that were performed to asses the lexical and syntactic quality of contextualized word-embeddings obtained from different pre-training corpora under comparison.Crucially, comparing them with and ELMo pre-trained on OSCAR and fine-tuned with CaBeRnet, i.e. $\text{ELMo}_{\text{OSCAR+CaBeRnet}}$, will allow to control for the presence of oral transcriptions and proceeding in order to understand its impact on the accuracy of our language model and on the development experiments after fine-tuning.

[9] Available at https://github.com/ufal/acl2019_nested_ner.

Syntactic tasks The evaluation tasks were selected to probe to what extent corpus "representativeness" and balance is impacting syntactic representations, in both (1) low-level syntactic relations in POS-tagging tasks, and (2) higher level syntactic relations at constituent- and sentence-level thanks to dependency-parsing evaluation task. Namely, POS-tagging is a low-level syntactic task, which consists in assigning to each word its corresponding grammatical category. Dependency-parsing consists of higher order syntactic task like predicting the labeled syntactic tree capturing the syntactic relations between words. We evaluate the performance of our models using the standard UPOS accuracy for POS-tagging, and Unlabeled Attachment Score (UAS) and Labeled Attachment Score (LAS) for dependency parsing. We assume gold tokenisation and gold word segmentation as provided in the UD treebanks.

Lexical tasks To test for word-level representation obtained through the different pre-training corpora and fine-tunings, Named Entity Recognition task (NER) was retained (5.4.3). As it involves a sequence labeling task that consists in predicting which words refer to real-world objects, such as people, locations, artifacts and organizations, it directly probes the quality and specificity of semantic representations issued by the more or less balanced corpora under comparison.

POS-tagging and dependency parsing

Experiments were run using the Universal Dependencies (UD) paradigm and its corresponding UD POS-tag set (Petrov et al., 2012) and UD treebank collection version 2.2 (Nivre et al., 2018), which was used for the CoNLL 2018 shared task.

Different terms of comparisons were considered on the two downstream tasks of part-of-speech (POS) tagging and dependency parsing.

Treebanks test data-set We perform our work on the four freely available French UD treebanks in UD v2.2: GSD, Sequoia, Spoken, and ParTUT, presented in Table 5.5.

GSD treebank (McDonald et al., 2013) is the second-largest tree-bank available for French after the FTB (described in subsection 5.4.3), it contains data from blogs, news, reviews, and Wikipedia.

Sequoia tree-bank (Candito et al., 2014) comprises more than 3000 sentences, from the French Europarl, the regional newspaper *L'Est Républicain*, the French Wikipedia and documents from the European Medicines Agency.

Spoken was automatically converted from the Rhapsodie tree-bank (Lacheret et al., 2014) with manual corrections. It consists of 57 sound samples of spoken French with phonetic transcription aligned with sound (word boundaries, syllables, and phonemes), syntactic and prosodic annotations.

Finally, **ParTUT** is a conversion of a multilingual parallel treebank developed at the University of Turin, and consisting of a variety of text genres, including talks,

Treebank	Tokens	Words	Sentences	Genre
GSD	389 363	400 387	16 342	News Wiki. Blogs
Sequoia	68 615	70 567	3 099	Pop. Wiki. Med. EuroParl
Spoken	34 972	34 972	2 786	Oral transcip.
ParTUT	27 658	28 594	1 020	Oral Wiki. Legal

Table 5.5: Sizes of the 4 treebanks used in the evaluations of POS-tagging and dependency parsing.

Model	GSD			Sequoia			Spoken			ParTUT		
	UPOS	UAS	LAS	UPOS	UAS	LAS	UPOS	UAS	LAS	UPOS	UAS	LAS
Baseline UDPipe Future	97.63	90.65	88.06	98.79	92.37	90.73	95.91	82.90	77.53	96.93	92.17	89.63
+ELMo$_{\text{CBT}}$	97.49	90.21	87.37	98.40	92.18	90.56	96.60	85.05	79.82	97.27	92.55	90.44
+ELMo$_{\text{Wikipedia}}$	97.92	92.13	89.77	99.22	94.28	92.97	97.28	85.61	80.79	**97.62**	94.01	91.78
+ELMo$_{\text{CaBeRnet}}$	97.87	92.02	89.62	99.33	94.42	93.14	***97.30***	85.39	80.63	97.43	94.02	91.86
+ELMo$_{\text{OSCAR}}$	97.85	92.41	90.05	99.30	94.43	93.25	97.10	85.83	**80.94**	97.47	**94.74**	**92.55**
+ELMo$_{\text{OSCAR+CaBeRnet}}$	**97.98**	**92.57**	**90.22**	**99.34**	**94.51**	**93.38**	97.24	**85.91**	80.93	97.58	94.47	92.05
State-of-the-art												
UDify	97.83	93.60	91.45	97.89	92.53	90.05	96.23	85.24	80.01	96.12	90.55	88.06
UDPipe Future + mBERT	97.98	92.55	90.31	*99.32*	94.88	93.81	97.23	*86.27*	*81.40*	*97.64*	94.51	92.47
CamemBERT	*98.19*	*94.82*	*92.47*	99.21	*95.56*	*94.39*	96.68	86.05	80.07	97.63	95.21	*92.90*

Table 5.6: Final POS and dependency parsing scores on 4 French treebanks (French GSD, Spoken, Sequoia and ParTUT), reported on test sets (4 averaged runs) assuming gold tokenisation. Best scores in bold, second to best underlined, state-of-the-art results in italics.

legal texts, and Wikipedia articles, among others; ParTUT data is derived from the already-existing parallel treebank, Par(allel)TUT (Sanguinetti and Bosco, 2015). Table 5.5 contains a summary comparing the sizes of the treebanks.

State-of-the-art For POS-tagging and Parsing we select as a baseline UDPipe Future (2.0), without any additional contextualized embeddings (Straka, 2018). This model was ranked 3rd in dependency parsing and 6th in POS-tagging during the CoNLL 2018 shared task (Seker et al., 2018). Notably, UDPipe Future provides us a strong baseline that does not make use of any pre-trained contextual embedding.

We report on Table 5.6 the published results on UDify by (Kondratyuk and Straka, 2019), a multitask and multilingual model based on mBERT that is near state-of-the-art on all UD languages including French for both POS-tagging and dependency parsing.

Finally, it is also relevant to compare our results with CamemBERT on the selected tasks, because compared to UDify it is the work that pushed the furthest the performance in fine-tuning end-to-end a BERT-based model.

Named Entity Recognition

Treebanks test data-set The benchmark data set from the French Treebank (FTB) (Abeillé et al., 2003) was selected in its 2008 version, as introduced by Candito and Crabbé (2009) and complemented with NER annotations by Sagot et al. (2012)[10]. The tree-bank, shows a large proportion of the entity mentions that are multi-word entities. We therefore report the three metrics that are commonly used to evaluate models: precision, recall, and F1 score.

NER State-of-the-art English has received the most attention in NER in the past, with some recent developments in German, Dutch and Spanish by Straková et al. (2019). In French, no extensive work has been done due to the limited availability of NER corpora. We compare our model with the stable baselines settled by (Dupont, 2017), who trained both CRF and BiLSTM-CRF architectures on the FTB and enhanced them using heuristics and pre-trained word-embeddings.

And additional term of comparison was identified in a recently released state-of-the-art language model for French, CamemBERT (Martin et al., 2020), based on the RoBERTa architecture pre-trained on the French sub-corpus of the newly available multilingual corpus OSCAR (Ortiz Suárez et al., 2019).

5.5 Results & Discussion

5.5.1 Dependency Parsing and POS-tagging

$\text{ELMo}_{\text{CaBeRnet}}$: a test for balance The word-embeddings representations offered by $\text{ELMo}_{\text{CaBeRnet}}$ are not only competitive but sometimes better than Wikipedia ones. One should keep in mind that almost all of the four treebanks we use in this section include Wikipedia data. $\text{ELMo}_{\text{CaBeRnet}}$ is reaching state-of-the-are results in POS-tagging on Spoken. Notably, it performs better than CamemBERT, the previous state of the art on this oral specialized tree-bank (cf. dark gray highlight on Table 5.6). We understand this results as a clear effect of balance when testing upon a purely spoken test-set. Importantly, this effect is difficultly explainable by the size of oral-style data in CaBeRnet. The oral sub-part is only one fifth of the total, and in this one fifth, only an even smaller amount of data comes from purely oral transcripts comparable the ones in the Spoken tree-bank, namely 67,444 words from Rhapsodie corpus, and 575,894 words form ORFEO. Hence, CaBeRnet's balanced oral language use shows to pay off in POS-tagging. These results are extremely surprising especially

[10]The NER-annotated FTB contains approximately than 12k sentences, and more than 350k tokens were extracted from articles of *Le Monde* newspaper (1989 - 1995). As a whole, it encompasses 11,636 entity mentions distributed among 7 different types : 2025 mentions of "Person", 3761 of "Location", 2382 of "Organisation", 3357 of "Company", 67 of "Product", 15 of "POI" (Point of Interest) and 29 of "Fictional Character".

NER - Results on FTB	Precision	Recall	F1
Baselines Models			
SEM (CRF) (Dupont, 2017)	87.89	82.34	85.02
LSTM-CRF (Dupont, 2017)	87.23	83.96	85.57
LSTM-CRF test models	85.87	81.35	83.55
+FastText	88.53	84.63	86.53
+FastText+ELMo$_{CBT}$	79.77	77.63	78.69
+FastText+ELMo$_{Wikipedia}$	88.87	87.56	88.21
+FastText+ELMo$_{CaBeRnet}$	<u>88.91</u>	87.22	88.06
+FastText+ELMo$_{OSCAR}$	88.89	<u>88.43</u>	<u>88.66</u>
+FastText+ELMo$_{OSCAR+CaBeRnet}$	**90.70**	**89.12**	**89.93**
State-of-the-art Models			
CamemBERT (Martin et al., 2020)	88.35	87.46	87.93

Table 5.7: NER Results on French Treebank (FTB): **best scores**, <u>second to best</u>.

given the fact that our evaluation method was aiming at comparing the quality of word-embedding representations and not beating the state-of-the-art.

ELMo$_{CaBeRnet}$: a test for coverage From Table 5.6, we discover that not only balance, but also the broad and diverse genre converge of CaBeRnet may play a role in its POS-tagging success is we compare its results with ELMo$_{CBT}$ that also features oral dialogues in youth literature. The fact that ELMo$_{CBT}$ does not show a comparable performance in POS-tagging, can be interpreted as linked to its size, but possibly also to its lack of variety in genres, thus, suggesting the advantage of a comprehensive coverage of language use. This suggests that a balanced sample may enhance the convergence of generalization about oral-style from distinct genre that still imply oral-like dialogues like in fiction. In sum, broad coverage may contribute to enhancing representations about oral language.

The effect of balance on Fine-tuning For POS-tagging in GSD the results of ELMo$_{OSCAR}$ are in second place position compared to ELMo$_{OSCAR+CaBeRnet}$ that is extremely close to ELMo$_{Wikipedia}$. While in POS-tagging in ParTUT, ELMo$_{Wikipedia}$ exhibits better results than ELMo$_{OSCAR}$, and ELMo$_{OSCAR+CaBeRnet}$ is in second position.

Further comparing GSD and Sequoia scores from $\text{ELMo}_{\text{OSCAR}}$ and $\text{ELMo}_{\text{OSCAR+CaBeRnet}}$, we observe that fine-tuning with CaBeRnet the emdeddings that were pre-trained on OSCAR, yields better representations for the three tasks compared to both the original $\text{ELMo}_{\text{OSCAR}}$ and $\text{ELMo}_{\text{CaBeRnet}}$. However, fine-tuning does not always yield better findings than $\text{ELMo}_{\text{OSCAR}}$ on Spoken and ParTUT, where $\text{ELMo}_{\text{OSCAR+CaBeRnet}}$ places in second after $\text{ELMo}_{\text{OSCAR}}$ for parsing scores UAS/LAS (cf. Table 5.6).

A closer look on Parsing results reveals an interesting pattern of results across treebanks (see light gray highlights on Table 5.6). We see that for GSD and Sequoia the CaBeRnet fine-tuned version $\text{ELMo}_{\text{OSCAR+CaBeRnet}}$ compared to the pure OSCAR pre-trained $\text{ELMo}_{\text{OSCAR}}$ is achieving higher scores. While a reverse and less clear-cut pattern is observable for the other two treebanks, namely Spoken and ParTUT. This configuration can be explained if we understand this pattern as due to the reinforcement and unlearning of $\text{ELMo}_{\text{OSCAR}}$ representations during the process of fine-tuning. Specifically, we can observe that parsing scores are better on treebanks that share the kind of language use represented in CaBeRnet, while they are worst on corpora that are closer in language sample to OSCAR corpus, like Spoken and ParTuT. This calls for further developments of CaBeRnet (§5.6).

ELMo_{CBT}: SMALL BUT RELEVANT ELMo_{CBT} shows an intriguing pattern of results. Even if its scores are under the baseline on GSD and Sequoia, it yields over the baseline results for Spoken and ParTUT. Given its reduced size, one would expect it to overfit, this would explain the under baseline performance. However, this was not the case on Spoken and ParTUT treebanks, thus showing ELMo_{CBT} contribution in generating representations that are useful to UDPipe model to achieve better results in POS-tagging and parsing tasks on the ParTUT and Spoken tree-banks. The presence of oral dialogues is certainly playing a role in this results' pattern. This unexpected result calls for further investigation on the impact of pre-training with reduced-size, noiseless, domain-specific corpora.

5.5.2 NER

For named entity recognition, LSTM-CRF +FastText +$\text{ELMo}_{\text{OSCAR+CaBeRnet}}$ achieves a better precision, recall and F1 than the traditional CRF-based SEM architectures (§ 5.4.3) and CamemBERT, which is currently state-of-the-art.Importantly, LSTM-CRF +FastText +$\text{ELMo}_{\text{CaBeRnet}}$ reaches better results in finding entity mentions, than Wikipedia which is a highly specialized corpus in terms of vocabulary variety and size, as can be seen in the overwhelming total number of unique forms it contains (see Table 5.4). We can conclude that both pre-training and fine-tuning with CaBeRnet on ELMo OSCAR generates better word-embedding representations than Wikipedia in this downstream task.

CBT-fr NER results are under the LSTM-CRF baseline. This can possibly be explained by the distance in terms of topics and domain from FTB tree-bank (i.e.

newspaper articles), or by the reduced-size of the corpus to yield good-enough representation to perform entity mentions recognition.

All in all, our evaluations confirm the effectiveness of large ELMo-based language models fine-tuned or pre-trained with a balanced and linguistically representative corpus, like CaBeRnet as opposed to domain-specific ones, or to an extra-large and noisy one like OSCAR.

5.6 Perspectives & Conclusion

The paper investigates the relevance of different types of corpora on ELMo's pre-training and fine-tuning. It confirms the effectiveness and quality of word-embeddings obtained through balanced and linguistically representative corpora.

By adding to UDPipe Future 5 differently trained ELMo language models that were pre-trained on qualitatively and quantitatively different corpora, our French Balanced Reference Corpus CaBeRnet unexpectedly establishes a new state-of-the-art for POS-tagging over previous monolingual (Straka, 2018) and multilingual approaches (Straka et al., 2019; Kondratyuk and Straka, 2019).

The proposed evaluation methods are showing that the two newly built corpora that are published here are not only relevant for neural NLP and language modeling in French, but that corpus balance shows to be a significant predictor of ELMo's accuracy on Spoken test data-set and for NER tasks.

Other perspective uses of CaBeRnet involve it use as a corpus offering a reference point for lexical frequency measures, like association measures. Its comparability with English COCA further grants the cross-linguistic validity of measures like Point-wise Mutual Information or DICE's Coefficient. The representativeness probed through our experimental approach are key aspects that allow such measures to be tested against psycho-linguistic and neuro-linguistic data as shown in previous neuro-imaging studies (Bhattasali et al., 2018).

The results obtained for the parsing tasks on ParTUT open a new perspective for the development of the French Balanced Reference Corpus, involving the enhancement of the terminological coverage of CaBeRnet. A sixth sub-part could be included to cover technical domains like legal and medical ones, and thereby enlarge the specialized lexical coverage of CaBeRnet. Further developments of this resource would involve an extension to cover user-generated content, ranging from well written blogs, tweets to more variable written productions like newspaper's comment or forums, as present in the CoMeRe corpus (Chanier et al., 2014).The computational experiments conducted here also show that pre-training language models like ELMo on a very small sample like the French Children Book Test corpus or CaBeRnet yields unexpected results. This opens a perspective for languages that have smaller training corpora. ELMo could be a better suited language model for those languages than it is for others having larger size resources.

Results on the NER task show that size - usually presented as the more important factor to enhance the precision of representation of word-embeddings - matters less than linguistic representativeness, as achieved through corpus linguistic balance. $\text{ELMo}_{\text{OSCAR+CaBeRnet}}$ sets state-of-the art results in NER (i.e. Precision, Recall and F1) that are superior than those obtained with a 30 times larger corpus, like OSCAR.

To conclude, our current evaluations show that linguistic quality in terms of *representativeness* and balance is yielding better performing contextualized word-embeddings.

6 Modern French Data

6.1 LEM17

6.2 presto max

6.3 presto gold

7 Ancient/Medieval French Data

7.1 BERTrade Corpus

8 Other Data

Part III

Models

9 CamemBERT

9.1 Introduction

Pretrained word representations have a long history in Natural Language Processing (NLP), from non-contextual (Brown et al., 1992; Ando and Zhang, 2005; Mikolov et al., 2013; Pennington et al., 2014) to contextual word embeddings (Peters et al., 2018; Akbik et al., 2018). Word representations are usually obtained by training language model architectures on large amounts of textual data and then fed as an input to more complex task-specific architectures. More recently, these specialized architectures have been replaced altogether by large-scale pretrained language models which are *fine-tuned* for each application considered. This shift has resulted in large improvements in performance over a wide range of tasks (Devlin et al., 2019; Radford et al., 2019; Liu et al., 2019; Raffel et al., 2020).

These transfer learning methods exhibit clear advantages over more traditional task-specific approaches. In particular, they can be trained in an *unsupervized* manner, thereby taking advantage of the information contained in large amounts of raw text. Yet they come with implementation challenges, namely the amount of data and computational resources needed for pretraining, which can reach hundreds of gigabytes of text and require hundreds of GPUs (Yang et al., 2019; Liu et al., 2019). This has limited the availability of these state-of-the-art models to the English language, at least in the monolingual setting. This is particularly inconvenient as it hinders their practical use in NLP systems. It also prevents us from investigating their language modelling capacity, for instance in the case of morphologically rich languages.

Although multilingual models give remarkable results, they are often larger, and their results, as we will observe for French, can lag behind their monolingual counterparts for high-resource languages.

In order to reproduce and validate results that have so far only been obtained for English, we take advantage of the newly available multilingual corpora OSCAR (Ortiz Suárez et al., 2019) to train a monolingual language model for French, dubbed CamemBERT. We also train alternative versions of CamemBERT on different smaller corpora with different levels of homogeneity in genre and style in order to assess the impact of these parameters on downstream task performance. CamemBERT uses the RoBERTa architecture (Liu et al., 2019), an improved variant of the high-performing and widely used BERT architecture (Devlin et al., 2019).

We evaluate our model on four different downstream tasks for French: part-of-speech (POS) tagging, dependency parsing, named entity recognition (NER) and natural language inference (NLI). CamemBERT improves on the state of the art in all four tasks compared to previous monolingual and multilingual approaches including mBERT, XLM and XLM-R, which confirms the effectiveness of large pretrained language models for French.

We make the following contributions:

- First release of a monolingual RoBERTa model for the French language using recently introduced large-scale open source corpora from the Oscar collection and first outside the original BERT authors to release such a large model for an other language than English.[1]
- We achieve state-of-the-art results on four downstream tasks: POS tagging, dependency parsing, NER and NLI, confirming the effectiveness of BERT-based language models for French.
- We demonstrate that small and diverse training sets can achieve similar performance to large-scale corpora, by analysing the importance of the pretraining corpus in terms of size and domain.

9.2 Previous work

9.2.1 Contextual Language Models

From non-contextual to contextual word embeddings The first neural word vector representations were non-contextualized word embeddings, most notably word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and fastText (Mikolov et al., 2018), which were designed to be used as input to task-specific neural architectures. Contextualized word representations such as ELMo (Peters et al., 2018) and flair (Akbik et al., 2018), improved the representational power of word embeddings by taking context into account. Among other reasons, they improved the performance of models on many tasks by handling words polysemy. This paved the way for larger contextualized models that replaced downstream architectures altogether in most tasks. Trained with language modeling objectives, these approaches range from LSTM-based architectures such as (Dai and Le, 2015), to the successful transformer-based architectures such as GPT2 (Radford et al., 2019), BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and more recently ALBERT (Lan et al., 2020) and T5 (Raffel et al., 2020).

Non-English contextualized models Following the success of large pretrained language models, they were extended to the multilingual setting with multilingual BERT (hereafter mBERT) (Devlin et al., 2019), a single multilingual model for 104 different languages trained on Wikipedia data, and later XLM (Conneau and Lample, 2019), which significantly improved unsupervized machine translation. More recently XLM-R (Conneau et al., 2020), extended XLM by training on 2.5TB of data and outperformed previous scores on multilingual benchmarks. They show that multilingual models can obtain results competitive with monolingual models by leveraging higher quality data from other languages on specific downstream tasks.

[1]Released at: `https://camembert-model.fr` under the MIT open-source license.

A few non-English monolingual models have been released: ELMo models for Japanese, Portuguese, German and Basque[2] and BERT for Simplified and Traditional Chinese (Devlin et al., 2019) and German (Chan et al., 2019).

However, to the best of our knowledge, no particular effort has been made toward training models for languages other than English at a scale similar to the latest English models (e.g. RoBERTa trained on more than 100GB of data).

BERT AND ROBERTA Our approach is based on RoBERTa (Liu et al., 2019) which itself is based on BERT (Devlin et al., 2019). BERT is a multi-layer bidirectional Transformer encoder trained with a masked language modeling (MLM) objective, inspired by the Cloze task (Taylor, 1953). It comes in two sizes: the $\text{BERT}_{\text{BASE}}$ architecture and the $\text{BERT}_{\text{LARGE}}$ architecture. The $\text{BERT}_{\text{BASE}}$ architecture is 3 times smaller and therefore faster and easier to use while $\text{BERT}_{\text{LARGE}}$ achieves increased performance on downstream tasks. RoBERTa improves the original implementation of BERT by identifying key design choices for better performance, using dynamic masking, removing the next sentence prediction task, training with larger batches, on more data, and for longer.

9.3 Downstream evaluation tasks

In this section, we present the four downstream tasks that we use to evaluate CamemBERT, namely: Part-Of-Speech (POS) tagging, dependency parsing, Named Entity Recognition (NER) and Natural Language Inference (NLI). We also present the baselines that we will use for comparison.

TASKS POS tagging is a low-level syntactic task, which consists in assigning to each word its corresponding grammatical category. Dependency parsing consists in predicting the labeled syntactic tree in order to capture the syntactic relations between words.

For both of these tasks we run our experiments using the Universal Dependencies (UD)[3] framework and its corresponding UD POS tag set (Petrov et al., 2012) and UD treebank collection (Nivre et al., 2018), which was used for the CoNLL 2018 shared task (Seker et al., 2018). We perform our evaluations on the four freely available French UD treebanks in UD v2.2: GSD (McDonald et al., 2013), Sequoia[4] (Candito and Seddah, 2012; Candito et al., 2014), Spoken (Lacheret et al., 2014; Bawden et al., 2014)[5], and ParTUT (Sanguinetti and Bosco, 2015). A brief overview of the size and content of each treebank can be found in Table 9.1.

[2] https://allennlp.org/elmo
[3] https://universaldependencies.org
[4] https://deep-sequoia.inria.fr
[5] Speech transcript uncased that includes annotated disfluencies without punctuation

Treebank	#Tokens	#Sentences	Genres
GSD	389,363	16,342	Blogs, News Reviews, Wiki
Sequoia	68,615	3,099	Medical, News Non-fiction, Wiki
Spoken	34,972	2,786	Spoken
ParTUT	27,658	1,020	Legal, News, Wikis
FTB	350,930	27,658	News

Table 9.1: Statistics on the treebanks used in POS tagging, dependency parsing, and NER (FTB).

We also evaluate our model in NER, which is a sequence labeling task predicting which words refer to real-world objects, such as people, locations, artifacts and organisations. We use the French Treebank[6] (FTB) (Abeillé et al., 2003) in its 2008 version introduced by Candito and Crabbé (2009) and with NER annotations by Sagot et al. (2012). The FTB contains more than 11 thousand entity mentions distributed among 7 different entity types. A brief overview of the FTB can also be found in Table 9.1.

Finally, we evaluate our model on NLI, using the French part of the XNLI dataset (Conneau et al., 2018). NLI consists in predicting whether a hypothesis sentence is entailed, neutral or contradicts a premise sentence. The XNLI dataset is the extension of the Multi-Genre NLI (MultiNLI) corpus (Williams et al., 2018) to 15 languages by translating the validation and test sets manually into each of those languages. The English training set is machine translated for all languages other than English. The dataset is composed of 122k train, 2490 development and 5010 test examples for each language. As usual, NLI performance is evaluated using accuracy.

Baselines In dependency parsing and POS-tagging we compare our model with:

- *mBERT*: The multilingual cased version of BERT (see Section 9.2.1). We fine-tune mBERT on each of the treebanks with an additional layer for POS-tagging and dependency parsing, in the same conditions as our CamemBERT model.
- $XLM_{MLM\text{-}TLM}$: A multilingual pretrained language model from Conneau and Lample (2019), which showed better performance than mBERT on NLI. We

[6]This dataset has only been stored and used on Inria's servers after signing the research-only agreement.

use the version available in the Hugging's Face transformer library (Wolf et al., 2019); like mBERT, we fine-tune it in the same conditions as our model.

- *UDify* (Kondratyuk and Straka, 2019): A multitask and multilingual model based on mBERT, UDify is trained simultaneously on 124 different UD treebanks, creating a single POS tagging and dependency parsing model that works across 75 different languages. We report the scores from Kondratyuk and Straka (2019) paper.

- *UDPipe Future* (Straka, 2018): An LSTM-based model ranked 3rd in dependency parsing and 6th in POS tagging at the CoNLL 2018 shared task (Seker et al., 2018). We report the scores from Kondratyuk and Straka (2019) paper.

- *UDPipe Future + mBERT + Flair* (Straka et al., 2019): The original UDPipe Future implementation using mBERT and Flair as feature-based contextualized word embeddings. We report the scores from Straka et al. (2019) paper.

In French, no extensive work has been done on NER due to the limited availability of annotated corpora. Thus we compare our model with the only recent available baselines set by Dupont (2017), who trained both CRF (Lafferty et al., 2001) and BiLSTM-CRF (Lample et al., 2016) architectures on the FTB and enhanced them using heuristics and pretrained word embeddings. Additionally, as for POS and dependency parsing, we compare our model to a fine-tuned version of mBERT for the NER task.

For XNLI, we provide the scores of mBERT which has been reported for French by Wu and Dredze (2019). We report scores from $\text{XLM}_{\text{MLM-TLM}}$ (described above), the best model from Conneau and Lample (2019). We also report the results of XLM-R (Conneau et al., 2020).

9.4 CamemBERT: a French Language Model

In this section, we describe the pretraining data, architecture, training objective and optimisation setup we use for CamemBERT.

9.4.1 Training data

Pretrained language models benefits from being trained on large datasets (Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2020). We therefore use the French part of the OSCAR corpus (Ortiz Suárez et al., 2019), a pre-filtered and pre-classified version of Common Crawl.[7]

[7] https://commoncrawl.org/about/

OSCAR is a set of monolingual corpora extracted from Common Crawl snapshots. It follows the same approach as (Grave et al., 2018) by using a language classification model based on the fastText linear classifier (Joulin et al., 2016; Joulin et al., 2017) pretrained on Wikipedia, Tatoeba and SETimes, which supports 176 languages. No other filtering is done. We use a non-shuffled version of the French data, which amounts to 138GB of raw text and 32.7B tokens after subword tokenization.

9.4.2 Pre-processing

We segment the input text data into subword units using SentencePiece (Kudo and Richardson, 2018). SentencePiece is an extension of Byte-Pair encoding (BPE) (Sennrich et al., 2016) and WordPiece (Kudo, 2018) that does not require pre-tokenization (at the word or token level), thus removing the need for language-specific tokenisers. We use a vocabulary size of 32k subword tokens. These subwords are learned on 10^7 sentences sampled randomly from the pretraining dataset. We do not use subword regularisation (i.e. sampling from multiple possible segmentations) for the sake of simplicity.

9.4.3 Language Modeling

Transformer Similar to RoBERTa and BERT, CamemBERT is a multi-layer bidirectional Transformer (Vaswani et al., 2017). Given the widespread usage of Transformers, we do not describe them here and refer the reader to (Vaswani et al., 2017). CamemBERT uses the original architectures of $\text{BERT}_{\text{BASE}}$ (12 layers, 768 hidden dimensions, 12 attention heads, 110M parameters) and $\text{BERT}_{\text{LARGE}}$ (24 layers, 1024 hidden dimensions, 16 attention heads, 335M parameters). CamemBERT is very similar to RoBERTa, the main difference being the use of whole-word masking and the usage of SentencePiece tokenization (Kudo and Richardson, 2018) instead of WordPiece (Schuster and Nakajima, 2012).

Pretraining Objective We train our model on the Masked Language Modeling (MLM) task. Given an input text sequence composed of N tokens $x_1, ..., x_N$, we select 15% of tokens for possible replacement. Among those selected tokens, 80% are replaced with the special `<MASK>` token, 10% are left unchanged and 10% are replaced by a random token. The model is then trained to predict the initial masked tokens using cross-entropy loss.

Following the RoBERTa approach, we dynamically mask tokens instead of fixing them statically for the whole dataset during preprocessing. This improves variability and makes the model more robust when training for multiple epochs.

Since we use SentencePiece to tokenize our corpus, the input tokens to the model are a mix of whole words and subwords. An upgraded version of BERT[8] and Joshi

[8] https://github.com/google-research/bert/blob/master/README.md

et al. (2020) have shown that masking whole words instead of individual subwords leads to improved performance. Whole-word Masking (WWM) makes the training task more difficult because the model has to predict a whole word rather than predicting only part of the word given the rest. We train our models using WWM by using whitespaces in the initial untokenized text as word delimiters.

WWM is implemented by first randomly sampling 15% of the words in the sequence and then considering all subword tokens in each of this 15% for candidate replacement. This amounts to a proportion of selected tokens that is close to the original 15%. These tokens are then either replaced by `<MASK>` tokens (80%), left unchanged (10%) or replaced by a random token.

Subsequent work has shown that the next sentence prediction (NSP) task originally used in BERT does not improve downstream task performance (Conneau and Lample, 2019; Liu et al., 2019), thus we also remove it. j

Optimisation Following (Liu et al., 2019), we optimize the model using Adam (Kingma and Ba, 2015) ($\beta_1 = 0.9, \beta_2 = 0.98$) for 100k steps with large batch sizes of 8192 sequences, each sequence containing at most 512 tokens. We enforce each sequence to only contain complete paragraphs (which correspond to lines in the our pretraining dataset).

Pretraining We use the RoBERTa implementation in the fairseq library (Ott et al., 2019). Our learning rate is warmed up for 10k steps up to a peak value of 0.0007 instead of the original 0.0001 given our large batch size, and then fades to zero with polynomial decay. Unless otherwise specified, our models use the BASE architecture, and are pretrained for 100k backpropagation steps on 256 Nvidia V100 GPUs (32GB each) for a day. We do not train our models for longer due to practical considerations, even though the performance still seemed to be increasing.

9.4.4 Using CamemBERT for downstream tasks

We use the pretrained CamemBERT in two ways. In the first one, which we refer to as *fine-tuning*, we fine-tune the model on a specific task in an end-to-end manner. In the second one, referred to as *feature-based embeddings* or simply *embeddings*, we extract frozen contextual embedding vectors from CamemBERT. These two complementary approaches shed light on the quality of the pretrained hidden representations captured by CamemBERT.

Fine-tuning For each task, we append the relevant predictive layer on top of CamemBERT's architecture. Following the work done on BERT (Devlin et al., 2019), for sequence tagging and sequence labeling we append a linear layer that respectively takes as input the last hidden representation of the `<s>` special token and the last

hidden representation of the first subword token of each word. For dependency parsing, we plug a bi-affine graph predictor head as inspired by Dozat and Manning (2017). We refer the reader to this article for more details on this module. We fine-tune on XNLI by adding a classification head composed of one hidden layer with a non-linearity and one linear projection layer, with input dropout for both.

We fine-tune CamemBERT independently for each task and each dataset. We optimize the model using the Adam optimiser (Kingma and Ba, 2015) with a fixed learning rate. We run a grid search on a combination of learning rates and batch sizes. We select the best model on the validation set out of the 30 first epochs. For NLI we use the default hyper-parameters provided by the authors of RoBERTa on the MNLI task.[9] Although this might have pushed the performances even further, we do not apply any regularisation techniques such as weight decay, learning rate warm-up or discriminative fine-tuning, except for NLI. We show that fine-tuning CamemBERT in a straightforward manner leads to state-of-the-art results on all tasks and outperforms the existing BERT-based models in all cases. The POS tagging, dependency parsing, and NER experiments are run using Hugging Face's Transformer library extended to support CamemBERT and dependency parsing (Wolf et al., 2019). The NLI experiments use the fairseq library following the RoBERTa implementation.

Embeddings Following Straková et al. (2019) and Straka et al. (2019) for mBERT and the English BERT, we make use of CamemBERT in a feature-based embeddings setting. In order to obtain a representation for a given token, we first compute the average of each sub-word's representations in the last four layers of the Transformer, and then average the resulting sub-word vectors.

We evaluate CamemBERT in the embeddings setting for POS tagging, dependency parsing and NER; using the open-source implementations of Straka et al. (2019) and Straková et al. (2019).[10]

9.5 Evaluation of CamemBERT

In this section, we measure the performance of our models by evaluating them on the four aforementioned tasks: POS tagging, dependency parsing, NER and NLI.

POS tagging and dependency parsing For POS tagging and dependency parsing, we compare CamemBERT with other models in the two settings: *fine-tuning* and as *feature-based embeddings*. We report the results in Table 9.2.

[9] More details at https://github.com/pytorch/fairseq/blob/master/examples/roberta/README.glue.md.

[10] UDPipe Future is available at https://github.com/CoNLL-UD-2018/UDPipe-Future, and the code for nested NER is available at https://github.com/ufal/acl2019_nested_ner.

Model	GSD		Sequoia		Spoken		ParTUT	
	UPOS	LAS	UPOS	LAS	UPOS	LAS	UPOS	LAS
mBERT (fine-tuned)	97.48	89.73	98.41	91.24	96.02	78.63	97.35	91.37
$XLM_{MLM-TLM}$ (fine-tuned)	98.13	90.03	98.51	91.62	96.18	80.89	97.39	89.43
UDify (Kondratyuk and Straka, 2019)	97.83	91.45	97.89	90.05	96.23	80.01	96.12	88.06
UDPipe Future (Straka, 2018)	97.63	88.06	98.79	90.73	95.91	77.53	96.93	89.63
+ mBERT + Flair (emb.) (Straka et al., 2019)	97.98	90.31	**99.32**	93.81	**97.23**	81.40	97.64	92.47
CamemBERT (fine-tuned)	**98.18**	**92.57**	99.29	**94.20**	96.99	81.37	**97.65**	**93.43**
UDPipe Future + CamemBERT (embeddings)	97.96	90.57	99.25	93.89	97.09	**81.81**	97.50	92.32

Table 9.2: **POS** and **dependency parsing** scores on 4 French treebanks, reported on test sets assuming gold tokenization and segmentation (best model selected on validation out of 4). Best scores in bold, second best underlined.

Model	F1
SEM (CRF) (Dupont, 2017)	85.02
LSTM-CRF (Dupont, 2017)	85.57
mBERT (fine-tuned)	87.35
CamemBERT (fine-tuned)	89.08
LSTM+CRF+CamemBERT (embeddings)	**89.55**

Table 9.3: **NER** scores on the FTB (best model selected on validation out of 4). Best scores in bold, second best underlined.

CamemBERT reaches state-of-the-art scores on all treebanks and metrics in both scenarios. The two approaches achieve similar scores, with a slight advantage for the fine-tuned version of CamemBERT, thus questioning the need for complex task-specific architectures such as UDPipe Future.

Despite a much simpler optimisation process and no task specific architecture, fine-tuning CamemBERT outperforms UDify on all treebanks and sometimes by a large margin (e.g. +4.15% LAS on Sequoia and +5.37 LAS on ParTUT). CamemBERT also reaches better performance than other multilingual pretrained models such as mBERT and $XLM_{MLM-TLM}$ on all treebanks.

CamemBERT achieves overall slightly better results than the previous state-of-the-art and task-specific architecture UDPipe Future+mBERT +Flair, except for POS tagging on Sequoia and POS tagging on Spoken, where CamemBERT lags by 0.03% and 0.14% UPOS respectively. UDPipe Future+mBERT +Flair uses the contextualized string embeddings Flair (Akbik et al., 2018), which are in fact pretrained contextualized character-level word embeddings specifically designed to handle misspelled words as well as subword structures such as prefixes and suffixes. This design choice might explain the difference in score for POS tagging with CamemBERT, especially for the Spoken treebank where words are not capitalized, a factor that might pose a problem for CamemBERT which was trained on capitalized data, but that might be properly handle by Flair on the UDPipe Future+mBERT +Flair model.

Model	Acc.	#Params
mBERT (Devlin et al., 2019)	76.9	175M
$XLM_{MLM-TLM}$ (Conneau and Lample, 2019)	80.2	250M
$XLM-R_{BASE}$ (Conneau et al., 2020)	80.1	270M
CamemBERT (fine-tuned)	**82.5**	110M
Supplement: LARGE models		
$XLM-R_{LARGE}$ (Conneau et al., 2020)	85.2	550M
$CamemBERT_{LARGE}$ (fine-tuned)	**85.7**	335M

Table 9.4: **NLI** accuracy on the French XNLI test set (best model selected on validation out of 10). Best scores in bold, second best underlined.

Named-Entity Recognition For NER, we similarly evaluate CamemBERT in the fine-tuning setting and as input embeddings to the task specific architecture LSTM+CRF. We report these scores in Table 9.3.

In both scenarios, CamemBERT achieves higher F1 scores than the traditional CRF-based architectures, both non-neural and neural, and than fine-tuned multilingual BERT models.[11]

Using CamemBERT as embeddings to the traditional LSTM+CRF architecture gives slightly higher scores than by fine-tuning the model (89.08 vs. 89.55). This demonstrates that although CamemBERT can be used successfully without any task-specific architecture, it can still produce high quality contextualized embeddings that might be useful in scenarios where powerful downstream architectures exist.

Natural Language Inference On the XNLI benchmark, we compare CamemBERT to previous state-of-the-art multilingual models in the fine-tuning setting. In addition to the standard CamemBERT model with a BASE architecture, we train another model with the LARGE architecture, referred to as $CamemBERT_{LARGE}$, for a fair comparison with $XLM-R_{LARGE}$. This model is trained with the CCNet corpus, described in Sec. 9.6, for 100k steps.[12] We expect that training the model for longer would yield even better performance.

CamemBERT reaches higher accuracy than its BASE counterparts reaching +5.6% over mBERT, +2.3 over $XLM_{MLM-TLM}$, and +2.4 over $XLM-R_{BASE}$. CamemBERT also uses as few as half as many parameters (110M vs. 270M for $XLM-R_{BASE}$).

$CamemBERT_{LARGE}$ achieves a state-of-the-art accuracy of 85.7% on the XNLI benchmark, as opposed to 85.2, for the recent $XLM-R_{LARGE}$.

CamemBERT uses fewer parameters than multilingual models, mostly because of its smaller vocabulary size (e.g. 32k vs. 250k for XLM-R). Two elements might

[11] $XLM_{MLM-TLM}$ is a lower-case model. Case is crucial for NER, therefore we do not report its low performance (84.37%)

[12] We train our LARGE model with the CCNet corpus for practical reasons. Given that BASE models reach similar performance when using OSCAR or CCNet as pretraining corpus (Appendix Table 9.8), we expect an OSCAR LARGE model to reach comparable scores.

explain the better performance of CamemBERT over XLM-R. Even though XLM-R was trained on an impressive amount of data (2.5TB), only 57GB of this data is in French, whereas we used 138GB of French data. Additionally XLM-R also handles 100 languages, and the authors show that when reducing the number of languages to 7, they can reach 82.5% accuracy for French XNLI with their BASE architecture.

Summary of CamemBERT's results CamemBERT improves the state of the art for the 4 downstream tasks considered, thereby confirming on French the usefulness of Transformer-based models. We obtain these results when using CamemBERT as a fine-tuned model or when used as contextual embeddings with task-specific architectures. This questions the need for more complex downstream architectures, similar to what was shown for English (Devlin et al., 2019). Additionally, this suggests that CamemBERT is also able to produce high-quality representations out-of-the-box without further tuning.

Dataset	Size	GSD		Sequoia		Spoken		ParTUT		**Average**		NER	NLI
		UPOS	LAS	UPOS	LAS	UPOS	LAS	UPOS	LAS	UPOS	LAS	F1	Acc.
Fine-tuning													
Wiki	4GB	98.28	93.04	98.74	92.71	96.61	79.61	96.20	89.67	97.45	88.75	89.86	78.32
CCNet	4GB	98.34	93.43	98.95	93.67	96.92	**82.09**	96.50	**90.98**	97.67	**90.04**	90.46	**82.06**
OSCAR	4GB	98.35	93.55	98.97	93.70	96.94	81.97	96.58	90.28	97.71	89.87	90.65	81.88
OSCAR	138GB	**98.39**	**93.80**	**98.99**	**94.00**	**97.17**	81.18	**96.63**	90.56	**97.79**	89.88	**91.55**	81.55
Embeddings (with UDPipe Future (tagging, parsing) or LSTM+CRF (NER))													
Wiki	4GB	98.09	92.31	98.74	93.55	96.24	78.91	95.78	89.79	97.21	88.64	91.23	-
CCNet	4GB	**98.22**	**92.93**	99.12	94.65	97.17	**82.61**	**96.74**	89.95	97.81	90.04	**92.30**	-
OSCAR	4GB	98.21	92.77	99.12	**94.92**	97.20	82.47	**96.74**	**90.05**	**97.82**	**90.05**	91.90	-
OSCAR	138GB	98.18	92.77	**99.14**	94.24	**97.26**	82.44	96.52	89.89	97.77	89.84	91.83	-

Table 9.5: Results on the four tasks using language models pre-trained on data sets of varying homogeneity and size, reported on validation sets (average of 4 runs for POS tagging, parsing and NER, average of 10 runs for NLI).

9.6 Impact of corpus origin and size

In this section we investigate the influence of the homogeneity and size of the pre-training corpus on downstream task performance. With this aim, we train alternative version of CamemBERT by varying the pretraining datasets. For this experiment, we fix the number of pretraining steps to 100k, and allow the number of epochs to vary accordingly (more epochs for smaller dataset sizes). All models use the BASE architecture.

In order to investigate the need for homogeneous clean data versus more diverse and possibly noisier data, we use alternative sources of pretraining data in addition to OSCAR:

- **Wikipedia**, which is homogeneous in terms of genre and style. We use the official 2019 French Wikipedia dumps[13]. We remove HTML tags and tables using Giuseppe Attardi's *WikiExtractor*.[14]

- **CCNet** (Wenzek et al., 2020), a dataset extracted from Common Crawl with a different filtering process than for OSCAR. It was built using a language model trained on Wikipedia, in order to filter out bad quality texts such as code or tables.[15] As this filtering step biases the noisy data from Common Crawl to more Wikipedia-like text, we expect CCNet to act as a middle ground between the unfiltered "noisy" OSCAR dataset, and the "clean" Wikipedia dataset. As a result of the different filtering processes, CCNet contains longer documents on average compared to OSCAR with smaller—and often noisier—documents weeded out.

Table 9.6 summarizes statistics of these different corpora.

Corpus	Size	#tokens	#docs	Tokens/doc Percentiles:		
				5%	50%	95%
Wikipedia	4GB	990M	1.4M	102	363	2530
CCNet	135GB	31.9B	33.1M	128	414	2869
OSCAR	138GB	32.7B	59.4M	28	201	1946

Table 9.6: Statistics on the pretraining datasets used.

In order to make the comparison between these three sources of pretraining data, we randomly sample 4GB of text (at the document level) from OSCAR and CCNet, thereby creating samples of both Common-Crawl-based corpora of the same size as the French Wikipedia. These smaller 4GB samples also provides us a way to investigate the impact of pretraining data size. Downstream task performance for our alternative versions of CamemBERT are provided in Table 9.5. The upper section reports scores in the fine-tuning setting while the lower section reports scores for the embeddings.

9.6.1 Common Crawl vs. Wikipedia?

Table 9.5 clearly shows that models trained on the 4GB versions of OSCAR and CCNet (Common Crawl) perform consistently better than the the one trained on

[13] https://dumps.wikimedia.org/backup-index.html.
[14] https://github.com/attardi/wikiextractor.
[15] We use the HEAD split, which corresponds to the top 33% of documents in terms of filtering perplexity.

the French Wikipedia. This is true both in the fine-tuning and embeddings setting. Unsurprisingly, the gap is larger on tasks involving texts whose genre and style are more divergent from those of Wikipedia, such as tagging and parsing on the Spoken treebank. The performance gap is also very large on the XNLI task, probably as a consequence of the larger diversity of Common-Crawl-based corpora in terms of genres and topics. XNLI is indeed based on multiNLI which covers a range of genres of spoken and written text.

The downstream task performances of the models trained on the 4GB version of CCNet and OSCAR are much more similar.[16]

9.6.2 How much data do you need?

An unexpected outcome of our experiments is that the model trained "only" on the 4GB sample of OSCAR performs similarly to the standard CamemBERT trained on the whole 138GB OSCAR. The only task with a large performance gap is NER, where "138GB" models are better by 0.9 F1 points. This could be due to the higher number of named entities present in the larger corpora, which is beneficial for this task. On the contrary, other tasks don't seem to gain from the additional data.

In other words, when trained on corpora such as OSCAR and CCNet, which are heterogeneous in terms of genre and style, 4GB of uncompressed text is large enough as pretraining corpus to reach state-of-the-art results with the BASE architecure, better than those obtained with mBERT (pretrained on 60GB of text).[17] This calls into question the need to use a very large corpus such as OSCAR or CCNet when training a monolingual Transformer-based language model such as BERT or RoBERTa. Not only does this mean that the computational (and therefore environmental) cost of training a state-of-the-art language model can be reduced, but it also means that CamemBERT-like models can be trained for all languages for which a Common-Crawl-based corpus of 4GB or more can be created. OSCAR is available in 166 languages, and provides such a corpus for 38 languages. Moreover, it is possible that slightly smaller corpora (e.g. down to 1GB) could also prove sufficient to train high-performing language models. We obtained our results with BASE architectures. Further research is needed to confirm the validity of our findings on larger architectures and other more complex natural language understanding tasks. However, even with a BASE architecture and 4GB of training data, the validation loss is still decreasing beyond 100k steps (and 400 epochs). This suggests that we are still under-fitting the 4GB pretraining dataset, training longer might increase downstream performance.

[16] We provide the results of a model trained on the whole CCNet corpus in the Appendix. The conclusions are similar when comparing models trained on the full corpora: downstream results are similar when using OSCAR or CCNet.

[17] The OSCAR-4GB model gets slightly better XNLI accuracy than the full OSCAR-138GB model (81.88 vs. 81.55). This might be due to the random seed used for pretraining, as each model is pretrained only once.

9.7 Discussion

Since the pre-publication of this work (Martin et al., 2020), many monolingual language models have appeared, e.g. (Le et al., 2020; Virtanen et al., 2019; Delobelle et al., 2020), for as much as 30 languages (Nozza et al., 2020). In almost all tested configurations they displayed better results than multilingual language models such as mBERT (Pires et al., 2019). Interestingly, Le et al. (2020) showed that using their FlauBert, a RoBERTa-based language model for French, which was trained on less but more edited data, in conjunction to CamemBERT in an ensemble system could improve the performance of a parsing model and establish a new state-of-the-art in constituency parsing of French, highlighting thus the complementarity of both models.[18] As it was the case for English when BERT was first released, the availability of similar scale language models for French enabled interesting applications, such as large scale anonymization of legal texts, where CamemBERT-based models established a new state-of-the-art on this task (Benesty, 2019), or the first large question answering experiments on a French Squad data set that was released very recently (d'Hoffschmidt et al., 2020) where the authors matched human performance using $\text{CamemBERT}_{\text{LARGE}}$. Being the first pre-trained language model that used the open-source Common Crawl Oscar corpus and given its impact on the community, CamemBERT paved the way for many works on monolingual language models that followed. Furthermore, the availability of all its training data favors reproducibility and is a step towards better understanding such models. In that spirit, we make the models used in our experiments available via our website and via the `huggingface` and `fairseq` APIs, in addition to the base CamemBERT model.

9.8 Conclusion

In this work, we investigated the feasibility of training a Transformer-based language model for languages other than English. Using French as an example, we trained CamemBERT, a language model based on RoBERTa. We evaluated CamemBERT on four downstream tasks (part-of-speech tagging, dependency parsing, named entity recognition and natural language inference) in which our best model reached or improved the state of the art in all tasks considered, even when compared to strong multilingual models such as mBERT, XLM and XLM-R, while also having fewer parameters.

Our experiments demonstrate that using web crawled data with high variability is preferable to using Wikipedia-based data. In addition we showed that our models could reach surprisingly high performances with as low as 4GB of pretraining data, questioning thus the need for large scale pretraining corpora. This shows that state-

[18] We refer the reader to (Le et al., 2020) for a comprehensive benchmark and details therein.

of-the-art Transformer-based language models can be trained on languages with far fewer resources than English, whenever a few gigabytes of data are available. This paves the way for the rise of monolingual contextual pre-trained language-models for under-resourced languages. The question of knowing whether pretraining on small domain specific content will be a better option than transfer learning techniques such as fine-tuning remains open and we leave it for future work.

Pretrained on pure open-source corpora, CamemBERT is freely available and distributed with the MIT license via popular NLP libraries (`fairseq` and `huggingface`) as well as on our website `camembert-model.fr`.

Appendix

Dataset	Masking	Arch.	#Steps	GSD		Sequoia		Spoken		ParTUT		NER	NLI
				UPOS	LAS	UPOS	LAS	UPOS	LAS	UPOS	LAS	F1	Acc.
Fine-tuning													
OSCAR	Subword	Base	100k	**98.25**	92.29	99.25	93.70	96.95	79.96	97.73	**92.68**	89.23	81.18
OSCAR	Whole-word	Base	100k	98.21	92.30	99.21	94.33	96.97	80.16	**97.78**	92.65	89.11	81.92
CCNET	Subword	Base	100k	98.02	92.06	**99.26**	94.13	96.94	80.39	97.55	92.66	89.05	81.77
CCNET	Whole-word	Base	100k	98.03	**92.43**	99.18	94.26	96.98	80.89	97.46	92.33	89.27	81.92
CCNET	Whole-word	Base	500k	98.21	**92.43**	99.24	**94.60**	96.69	**80.97**	97.65	92.48	89.08	83.43
CCNET	Whole-word	Large	100k	98.01	91.09	99.23	93.65	**97.01**	80.89	97.41	92.59	**89.39**	**85.29**
Embeddings (with UDPipe Future (tagging, parsing) or LSTM+CRF (NER))													
OSCAR	Subword	Base	100k	**98.01**	90.64	**99.27**	94.26	97.15	**82.56**	**97.70**	92.70	**90.25**	-
OSCAR	Whole-word	Base	100k	97.97	90.44	99.23	93.93	97.08	81.74	97.50	92.28	89.48	-
CCNET	Subword	Base	100k	97.87	**90.78**	99.20	94.33	**97.17**	82.39	97.54	92.51	89.38	-
CCNET	Whole-word	Base	100k	97.96	90.76	99.23	**94.34**	97.04	82.09	97.39	**92.82**	89.85	-
CCNET	Whole-word	Base	500k	97.84	90.25	99.14	93.96	97.01	82.17	97.27	92.28	89.07	-
CCNET	Whole-word	Large	100k	**98.01**	90.70	99.23	94.01	97.04	82.18	97.31	92.28	88.76	-

Table 9.7: Performance reported on **Test sets** for all trained models (**average** over multiple fine-tuning seeds).

In the appendix, we analyse different design choices of CamemBERT (Table 9.8), namely with respect to the use of whole-word masking, the training dataset, the model size, and the number of training steps in complement with the analyses of the impact of corpus origin an size (Section 9.6. In all the ablations, all scores come from at least 4 averaged runs. For POS tagging and dependency parsing, we average the scores on the 4 treebanks. We also report all averaged test scores of our different models in Table 9.7.

9.9 Impact of Whole-Word Masking

In Table 9.8, we compare models trained using the traditional subword masking with whole-word masking. Whole-Word Masking positively impacts downstream

Dataset	Masking	Arch.	#Param.	#Steps	UPOS	LAS	NER	XNLI
Masking Strategy								
OSCAR	Subword	Base	110M	100k	97.78	89.80	**91.55**	81.04
OSCAR	Whole-word	Base	110M	100k	**97.79**	**89.88**	91.44	**81.55**
Model Size								
CCNet	Whole-word	Base	110M	100k	97.67	89.46	90.13	82.22
CCNet	Whole-word	Large	335M	100k	**97.74**	**89.82**	**92.47**	**85.73**
Dataset								
CCNet	Whole-word	Base	110M	100k	97.67	89.46	90.13	**82.22**
OSCAR	Whole-word	Base	110M	100k	**97.79**	**89.88**	**91.44**	81.55
Number of Steps								
CCNet	Whole-word	Base	110M	100k	**98.04**	89.85	90.13	82.20
CCNet	Whole-word	Base	110M	500k	97.95	**90.12**	91.30	**83.04**

Table 9.8: Comparing scores on the **Validation sets** of different design choices. POS tagging and parsing datasets are averaged. (average over multiple fine-tuning seeds).

performances for NLI (although only by 0.5 points of accuracy). To our surprise, this Whole-Word Masking scheme does not benefit much lower level task such as Name Entity Recognition, POS tagging and Dependency Parsing.

9.10 Impact of model size

Table 9.8 compares models trained with the BASE and LARGE architectures. These models were trained with the CCNet corpus (135GB) for practical reasons. We confirm the positive influence of larger models on the NLI and NER tasks. The LARGE architecture leads to respectively 19.7% error reduction and 23.7%. To our surprise, on POS tagging and dependency parsing, having three time more parameters doesn't lead to a significant difference compared to the BASE model. Tenney et al. (2019) and Jawahar et al. (2019) have shown that low-level syntactic capabilities are learnt in lower layers of BERT while higher level semantic representations are found in upper layers of BERT. POS tagging and dependency parsing probably do not benefit from adding more layers as the lower layers of the BASE architecture already capture what is necessary to complete these tasks.

9.11 Impact of training dataset

Table 9.8 compares models trained on CCNet and on OSCAR. The major difference between the two datasets is the additional filtering step of CCNet that favors Wikipedia-Like texts. The model pretrained on OSCAR gets slightly better results

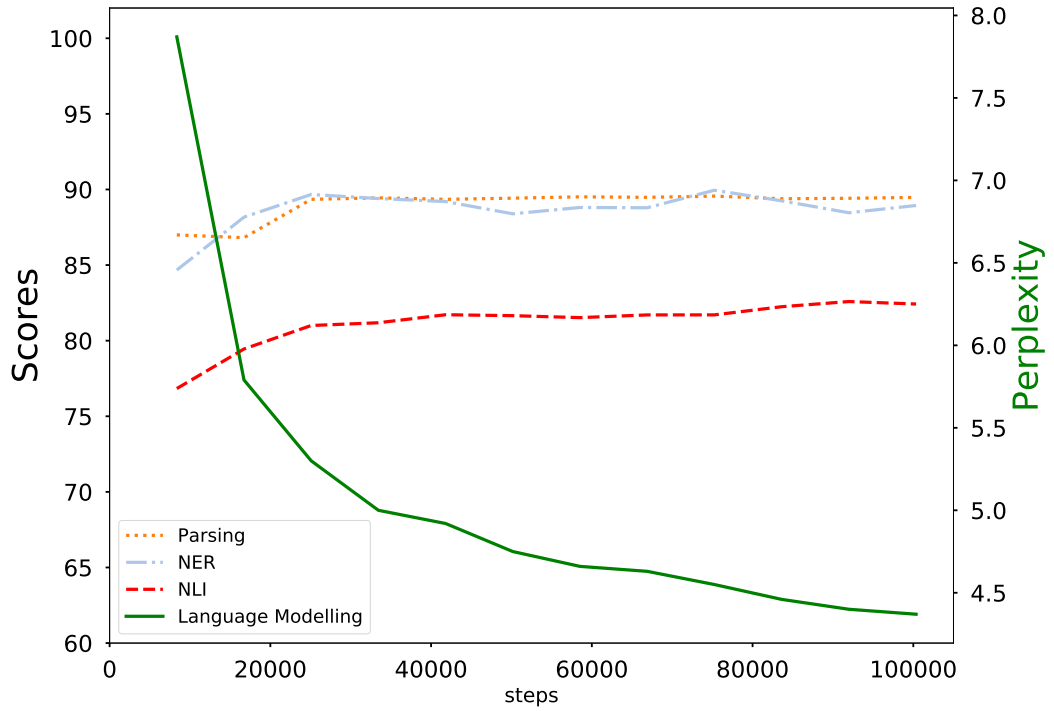


Figure 9.1: Impact of number of pretraining steps on downstream performance for CamemBERT.

.

on POS tagging and dependency parsing, but gets a larger +1.31 improvement on NER. The CCNet model gets better performance on NLI (+0.67).

9.12 Impact of number of steps

Figure 9.1 displays the evolution of downstream task performance with respect to the number of steps. All scores in this section are averages from at least 4 runs with different random seeds. For POS tagging and dependency parsing, we also average the scores on the 4 treebanks.

We evaluate our model at every epoch (1 epoch equals 8360 steps). We report the masked language modelling perplexity along with downstream performances. Figure 9.1, suggests that the more complex the task the more impactful the number of steps is. We observe an early plateau for dependency parsing and NER at around 22k steps, while for NLI, even if the marginal improvement with regard to pretraining steps becomes smaller, the performance is still slowly increasing at 100k steps.

In Table 9.8, we compare two models trained on CCNet, one for 100k steps and the other for 500k steps to evaluate the influence of the total number of steps. The model

trained for 500k steps does not increase the scores much from just training for 100k steps in POS tagging and parsing. The increase is slightly higher for XNLI (+0.84).

Those results suggest that low level syntactic representation are captured early in the language model training process while it needs more steps to extract complex semantic information as needed for NLI.

10 FrELMo

10.1 Introduction

Named entity recognition (NER) is the widely studied task consisting in identifying text spans that denote *named entities* such as person, location and organization names, to name the most important types. Such text spans are called named entity *mentions*. In NER, mentions are generally not only identified, but also classified according to a more or less fine-grained ontology, thereby allowing for instance to distinguish between the telecommunication company *Orange* and the town *Orange* in southern France (amongst others). Importantly, it has long been recognised that the type of named entities can be defined in two ways, which underlies the notion of metonymy: the intrinsic type (*France* is always a location) and the contextual type (in *la France a signé un traité* 'France signed a treaty', *France* denotes an organization).

NER has been an important task in natural language processing for quite some time. It was already the focus of the MUC conferences and associated shared tasks (Marsh and Perzanowski, 1998), and later that of the CoNLL 2003 and ACE shared tasks (Tjong Kim Sang and De Meulder, 2003; Doddington et al., 2004). Traditionally, as for instance was the case for the MUC shared tasks, only person names, location names, organization names, and sometimes "other proper names" are considered. However, the notion of named entity mention is sometimes extended to cover any text span that does not follow the general grammar of the language at hand, but a type- and often culture-specific grammar, thereby including entities ranging from product and brand names to dates and from URLs to monetary amounts and other types of numbers.

As for many other tasks, NER was first addressed using rule-based approaches, followed by statistical and now neural machine learning techniques (see Section 10.3.1 for a brief discussion on NER approaches). Of course, evaluating NER systems as well as training machine-learning-based NER systems, statistical or neural, require named-entity-annotated corpora. Unfortunately, most named entity annotated French corpora are oral transcripts, and they are not always freely available. The ESTER and ESTER2 corpora (60 plus 150 hours of NER-annotated broadcast transcripts) (Galliano et al., 2005, 2009), as well as the Quaero (Grouin et al., 2011) corpus are based on oral transcripts (radio broadcasts). Interestingly, the Quaero corpus relies on an original, very rich and structured definition of the notion of named entity (Rosset et al., 2011). It contains both the intrinsic and the contextual types of each mention, whereas the ESTER and ESTER2 corpora only provide the contextual type.

Sagot et al. (2012) describe the addition to the French Treebank (FTB) (Abeillé et al., 2003) in its FTB-UC version (Candito et al., 2010) of a new, freely available annotation layer providing named entity information in terms of span and type (NER) as well as reference (NE linking), using the Wikipedia-based Aleda (Sagot and Stern, 2012) as a reference entity database. This was the first freely available French corpus annotated with referential named entity information and the first freely available such corpus for the written journalistic genre. However, this annotation is provided in the form of

an XML-annotated text with sentence boundaries but no tokenization. This corpus will be referred to as FTB-NE in the rest of the article.

Since the publication of that named entity FTB annotation layer, the field has evolved in many ways. Firstly, most treebanks are now available as part of the *Universal Dependencies* (UD)[1] treebank collection. Secondly, neural approaches have considerably improved the state of the art in natural language processing in general and in NER in particular. In this regard, the emergence of contextual language models has played a major role. However, surprisingly few neural French NER systems have been published.[2] This might be because large contextual language models for French have only been made available very recently (Martin et al., 2020). But it is also the result of the fact that getting access to the FTB with its named entity layer as well as using this corpus were not straightforward tasks.

For a number of technical reasons, re-aligning the XML-format named entity FTB annotation layer created by Sagot et al. (2012) with the "official" version of the FTB or, later, with the version of the FTB provided in the Universal Dependency (UD) framework was not a straightforward task.[3] Moreover, due to the intellectual property status of the source text in the FTB, the named entity annotations could only be provided to people having signed the FTB license, which prevented them from being made freely downloadable online.

The goal of this paper is to establish a new state of the art for French NER by (i) providing a new, easy-to-use UD-aligned version of the named entity annotation layer in the FTB and (ii) using this corpus as a training and evaluation dataset for carrying out NER experiments using state-of-the-art architectures, thereby improving over the previous state of the art in French NER. In particular, by using both FastText embeddings (Bojanowski et al., 2017) and one of the versions of the CamemBERT French neural contextual language model (Martin et al., 2020) within an LSTM-CRF architecture, we can reach an F1-score of 90.25, a 6.5-point improvement over the previously state-of-the-art system SEM (Dupont, 2017).

10.2 A named entity annotation layer for the UD version of the French TreeBank

In this section, we describe the process whereby we re-aligned the named entity FTB annotations by Sagot et al. (2012) with the UD version of the FTB. This makes it possible to share these annotations in the form of a set of additional columns that can easily be pasted to the UD FTB file. This new version of the named entity FTB

[1] https://universaldependencies.org

[2] We are only aware of the *entity-fishing* NER (and NE linking) system developed by Patrice Lopez, a freely available yet unpublished system.

[3] Note that the UD version of the FTB is freely downloadable, but does not include the original tokens or lemmas. Only people with access to the original FTB can restore this information, as required by the intellectual property status of the source text.

layer is much more readily usable than the original XML version, and will serve as a basis for our experiments in the next sections. Yet information about the named entity annotation guidelines, process and results can only be found in Sagot et al. (2012), which is written in French. We therefore begin with a brief summary of this publication before describing the alignment process.

10.2.1 The original named entity FTB layer

Sagot et al. (2012) annotated the FTB with the span, absolute type[4], sometimes subtype and Aleda unique identifier of each named entity mention.[5] Annotations are restricted to person, location, organization and company names, as well as a few product names.[6] There are no nested entities. Non capitalized entity mentions (e.g. *banque mondiale* 'World Bank') are annotated only if they can be disambiguated independently of their context. Entity mentions that require the context to be disambiguated (e.g. *Banque centrale*) are only annotated if they are capitalized. [7] For person names, grammatical or contextual words around the mention are not included in the mention (e.g. in *M. Jacques Chirac* or *le Président Jacques Chirac*, only *Jacques Chirac* is included in the mention).

Tags used for the annotation have the following information:

- the identifier of the NE in the Aleda database (`eid` attribute); when a named entity is not present in the database, the identifier is `null`,[8]
- the normalized named of the named entity as given in Aleda; for locations it is their name as given in GeoNames and for the others, it is the title of the corresponding French Wikipedia article,
- the type and, when relevant, the subtype of the entity.

Here are two annotation examples:

```
<ENAMEX type="Organization" eid="1000000000016778" name="Confédération française
```

[4]Every mention of *France* is annotated as a `Location` with subtype `Country`, as given in Aleda database, even if in context the mentioned entity is a political organization, the French people, a sports team, etc.

[5]Only proper nouns are considered as named entity mentions, thereby excluding other types of referential expressions.

[6]More precisely, we used a tagset of 7 base NE types: `Person`, `Location`, `Organization`, `Company`, `Product`, `POI` (Point of Interest) and `FictionChar`.

[7]So for instance, in *université de Nantes* 'Nantes university', only *Nantes* is annotated, as a city, as *université* is written in lowercase letters. However, *Université de Nantes* 'Nantes University' is wholly annotated as an organization. It is non-ambiguous because *Université* is capitalized. *Université de Montpellier* 'Montpellier University' being ambiguous when the text of the FTB was written and when the named entity annotations were produced, only *Montpellier* is annotated, as a city.

[8]Specific conventions for entities that have merged, changed name, ceased to exist as such (e.g. *Tchequoslovaquie*) or evolved in other ways are described in Sagot et al. (2012).

```
démocratique du travail">CFDT</ENAMEX>
<ENAMEX type="Location" sub_type="Country" eid="2000000001861060" name="Japan">Japon</ENAMEX>
```

Sagot et al. (2012) annotated the 2007 version of the FTB treebank (with the exception of sentences that did not receive any functional annotation), i.e. 12,351 sentences comprising 350,931 tokens. The annotation process consisted in a manual correction and validation of the output of a rule- and heuristics-based named entity recognition and linking tool in an XML editor. Only a single person annotated the corpus, despite the limitations of such a protocol, as acknowledged by Sagot et al. (2012).

In total, 5,890 of the 12,351 sentences contain at least a named entity mention. 11,636 mentions were annotated, which are distributed as follows: 3,761 location names, 3,357 company names, 2,381 organization names, 2,025 person names, 67 product names, 29 fiction character names and 15 points of interest.

10.2.2 Alignment to the UD version of the FTB

The named entity (NE) annotation layer for the FTB was developed using an XML editor on the raw text of the FTB. Annotations are provided as inline XML elements within the sentence-segmented but non tokenized text. For creating our NER models, we first had to align these XML annotations with the already tokenized UD version of FTB.

Sentences were provided in the same order for both corpora, so we did not have to align them. For each sentence, we created a mapping M between the raw text of the NE-annotated FTB (i.e. after having removed all XML annotations) and tokens in the UD version of the FTB corpus. More precisely, character offsets in the FTB-NE raw text were mapped to token offsets in the tokenized FTB-UD. This alignment was done using case insensitive character-based comparison and were a mapping of a span in the raw text to a span in the tokenized corpus. We used the inlined XML annotations to create offline, character-level NE annotations for each sentence, and reported the NE annotations at the token level in the FTB-UD using the mapping M obtained.

We logged each error (i.e. an unaligned NE or token) and then manually corrected the corpora, as those cases were always errors in either corpora and not alignment errors. We found 70 errors in FTB-NE and 3 errors in FTB-UD. Errors in FTB-NE were mainly XML entity problems (unhandled "&", for instance) or slightly altered text (for example, a missing comma). Errors in FTB-UD were probably some XML artifacts.

10.3 Benchmarking NER Models

10.3.1 Brief state of the art of NER

As mentioned above, NER was first addressed using rule-based approaches, followed by statistical and now neural machine learning techniques. In addition, many systems use a lexicon of named entity mentions, usually called a "gazetteer" in this context.

Most of the advances in NER have been achieved on English, in particular with the CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003) and Ontonotes v5 (Pradhan et al., 2012, 2013) corpora. In recent years, NER was traditionally tackled using Conditional Random Fields (CRF) (Lafferty et al., 2001) which are quite suited for NER; CRFs were later used as decoding layers for Bi-LSTM architectures (Huang et al., 2015; Lample et al., 2016) showing considerable improvements over CRFs alone. These Bi-LSTM-CRF architectures were later enhanced with contextualized word embeddings which yet again brought major improvements to the task (Peters et al., 2018; Akbik et al., 2018). Finally, large pre-trained architectures settled the current state of the art showing a small yet important improvement over previous NER-specific architectures (Devlin et al., 2019; Baevski et al., 2019).

For French, rule-based system have been developed until relatively recently, due to the lack of proper training data (Sekine and Nobata, 2004; Rosset et al., 2005; Stern and Sagot, 2010; Nouvel et al., 2014). The limited availability of a few annotated corpora (cf. Section 10.1) made it possible to apply statistical machine learning techniques (Bechet and Charton, 2010; Dupont and Tellier, 2014; Dupont, 2017) as well as hybrid techniques combining handcrafted grammars and machine learning (Béchet et al., 2011). To the best of our knowledge, the best results previously published on FTB NER are those obtained by Dupont (2017), who trained both CRF and BiLSTM-CRF architectures and improved them using heuristics and pre-trained word embeddings. We use this system as our strong baseline.

Leaving aside French and English, the CoNLL 2002 shared task included NER corpora for Spanish and Dutch corpora (Tjong Kim Sang, 2002) while the CoNLL 2003 shared task included a German corpus (Tjong Kim Sang and De Meulder, 2003). The recent efforts by Straková et al. (2019) settled the state of the art for Spanish and Dutch, while Akbik et al. (2018) did so for German.

10.3.2 Experiments

We used SEM (Dupont, 2017) as our strong baseline because, to the best of our knowledge, it was the previous state-of-the-art for named entity recognition on the FTB-NE corpus. Other French NER systems are available, such as the one given by SpaCy. However, it was trained on another corpus called WikiNER, making the results non-comparable. We can also cite the system of (Stern et al., 2012). This system was trained on another newswire (AFP) using the same annotation

Model	Precision	Recall	F1-Score
			baseline
SEM (CRF)	87.18	80.48	83.70
LSTM-seq2seq	85.10	81.87	83.45
+ FastText	86.98	83.07	84.98
+ FastText + FrELMo	89.49	87.48	88.47
+ FastText + CamemBERT$_{\text{OSCAR-BASE-WWM}}$	89.79	88.86	89.32
+ FastText + CamemBERT$_{\text{OSCAR-BASE-WWM}}$ + FrELMo	90.00	88.60	89.30
+ FastText + CamemBERT$_{\text{CCNET-BASE-WWM}}$	90.31	89.29	89.80
+ FastText + CamemBERT$_{\text{CCNET-BASE-WWM}}$ + FrELMo	90.11	88.86	89.48
+ FastText + CamemBERT$_{\text{OSCAR-BASE-SWM}}$	90.09	89.46	89.77
+ FastText + CamemBERT$_{\text{OSCAR-BASE-SWM}}$ + FrELMo	90.11	88.95	89.53
+ FastText + CamemBERT$_{\text{CCNET-BASE-SWM}}$	90.31	89.38	89.84
+ FastText + CamemBERT$_{\text{CCNET-BASE-SWM}}$ + FrELMo	90.64	89.46	<u>90.05</u>
+ FastText + CamemBERT$_{\text{CCNET-500K-WWM}}$	<u>90.68</u>	89.03	89.85
+ FastText + CamemBERT$_{\text{CCNET-500K-WWM}}$ + FrELMo	90.13	88.34	89.23
+ FastText + CamemBERT$_{\text{CCNET-LARGE-WWM}}$	90.39	88.51	89.44
+ FastText + CamemBERT$_{\text{CCNET-LARGE-WWM}}$ + FrELMo	89.72	88.17	88.94
			LSTM-CRF + embeddings
LSTM-CRF	85.87	81.35	83.55
+ FastText	88.53	84.63	86.53
+ FastText + FrELMo	88.89	88.43	88.66
+ FastText + CamemBERT$_{\text{OSCAR-BASE-WWM}}$	90.47	88.51	89.48
+ FastText + CamemBERT$_{\text{OSCAR-BASE-WWM}}$ + FrELMo	89.70	88.77	89.24
+ FastText + CamemBERT$_{\text{CCNET-BASE-WWM}}$	90.24	89.46	89.85
+ FastText + CamemBERT$_{\text{CCNET-BASE-WWM}}$ + FrELMo	89.38	88.69	89.03
+ FastText + CamemBERT$_{\text{OSCAR-BASE-SWM}}$	**90.96**	<u>89.55</u>	**90.25**
+ FastText + CamemBERT$_{\text{OSCAR-BASE-SWM}}$ + FrELMo	89.44	88.51	88.98
+ FastText + CamemBERT$_{\text{CCNET-BASE-SWM}}$	90.09	88.69	89.38
+ FastText + CamemBERT$_{\text{CCNET-BASE-SWM}}$ + FrELMo	88.18	87.65	87.92
+ FastText + CamemBERT$_{\text{CCNET-500K-WWM}}$	89.46	88.69	89.07
+ FastText + CamemBERT$_{\text{CCNET-500K-WWM}}$ + FrELMo	90.11	88.86	89.48
+ FastText + CamemBERT$_{\text{CCNET-LARGE-WWM}}$	89.19	88.34	88.76
+ FastText + CamemBERT$_{\text{CCNET-LARGE-WWM}}$ + FrELMo	89.03	88.34	88.69
			fine-tuning
mBERT	80.35	84.02	82.14
CamemBERT$_{\text{OSCAR-BASE-WWM}}$	89.36	89.18	89.27
CamemBERT$_{\text{CCNET-500K-WWM}}$	89.35	88.81	89.08
CamemBERT$_{\text{CCNET-LARGE-WWM}}$	88.76	**89.58**	89.39

Table 10.1: Results on the test set for the best development set scores.

Model	Precision	Recall	F1-Score
			shuf 1
SEM(dev)	92.96	87.84	90.33
LSTM-CRF+CamemBERT$_{\text{OSCAR-BASE-SWM}}$(dev)	93.77	94.00	93.89
SEM(test)	91.88	87.14	89.45
LSTM-CRF+CamemBERT$_{\text{OSCAR-BASE-SWM}}$(test)	**92.59**	**93.96**	**93.27**
			shuf 2
SEM(dev)	91.67	85.96	88.73
LSTM-CRF+CamemBERT$_{\text{OSCAR-BASE-SWM}}$(dev)	93.15	94.21	93.68
SEM(test)	90.57	87.76	89.14
LSTM-CRF+CamemBERT$_{\text{OSCAR-BASE-SWM}}$(test)	**92.63**	**94.31**	**93.46**
			shuf 3
SEM(dev)	92.53	88.75	90.60
LSTM-CRF+CamemBERT$_{\text{OSCAR-BASE-SWM}}$(dev)	94.85	95.82	95.34
SEM(test)	90.68	85.00	87.74
LSTM-CRF+CamemBERT$_{\text{OSCAR-BASE-SWM}}$(test)	**91.30**	**92.67**	**91.98**

Table 10.2: Results on the test set for the best development set scores.

guidelines, so the results given in this article are not directly comparable. This model was trained on FTB-NE in Stern (2013) (table C.7, page 303), but the article is written in French. The model yielded an F1-score of 0.7564, which makes it a weaker baseline than SEM. We can cite yet another NER system, namely grobid-ner.[9] It was trained on the FTB-NE and yields an F1-score of 0.8739. Two things are to be taken into consideration: the tagset was slightly modified and scores were averaged over a 10-fold cross validation. To see why this is important for FTB-NE, see section 10.3.2.

In this section, we will compare our strong baseline with a series of neural models. We will use the two current state-of-the-art neural architectures for NER, namely seq2seq and LSTM-CRFs models. We will use various pre-trained embeddings in said architectures: fastText, CamemBERT (a French BERT-like model) and FrELMo (a French ELMo model) embeddings.

SEM

SEM (Dupont, 2017) is a tool that relies on linear-chain CRFs (Lafferty et al., 2001) to perform tagging. SEM uses Wapiti (Lavergne et al., 2010) v1.5.0 as linear-chain CRFs implementation. SEM uses the following features for NER:

- token, prefix/suffix from 1 to 5 and a Boolean isDigit features in a [-2, 2] window;
- previous/next common noun in sentence;

[9] https://github.com/kermitt2/grobid-ner#corpus-lemonde-ftb-french

- 10 gazetteers (including NE lists and trigger words for NEs) applied with some priority rules in a [-2, 2] window;
- a "fill-in-the-gaps" gazetteers feature where tokens not found in any gazetteer are replaced by their POS, as described in (Raymond and Fayolle, 2010). This features used token unigrams and token bigrams in a [-2, 2] a window.
- tag unigrams and bigrams.

We trained our own SEM model by using SEM features on gold tokenization and optimized L1 and L2 penalties on the development set. The metric used to estimate convergence of the model is the error on the development set ($1 - accuracy$). Our best result on the development set was obtained using the rprop algorithm, a 0.1 L1 penalty and a 0.1 L2 penalty.

SEM also uses an NE mention broadcasting post-processing (mentions found at least once are used as a gazetteer to tag unlabeled mentions), but we did not observe any improvement using this post-processing on the best hyperparameters on the development set.

Neural models

In order to study the relative impact of different word vector representations and different architectures, we trained a number of NER neural models that differ in multiple ways. They use zero to three of the following vector representations: FastText non-contextual embeddings (Bojanowski et al., 2017), the FrELMo contextual language model obtained by training the ELMo architecture on the OSCAR large-coverage Common-Crawl-based corpus developed by Ortiz Suárez et al. (2019), and one of multiple CamemBERT language models (Martin et al., 2020). CamemBERT models are transformer-based models based on an architecture similar to that of RoBERTa (Liu et al., 2019), an improvement over the widely used and successful BERT model (Devlin et al., 2019). The CamemBERT models we use in our experiments differ in multiple ways:

- Training corpus: OSCAR (cited above) or CCNet, another Common-Crawl-based corpus (Wenzek et al., 2020) classified by language, of an almost identical size (~32 billion tokens); although extracted using similar pipelines from Common Crawl, they differ slightly in so far that OSCAR better reflects the variety of genre and style found in Common Crawl, whereas CCNet was designed to better match the style of Wikipedia; moreover, OSCAR is freely available, whereas only the scripts necessary to rebuild CCNet can be downloaded freely. For comparison purposes, we also display the results of an experiment using the mBERT multilingual BERT model trained on the Wikpiedias for over 100 languages.

- Model size: following Devlin et al. (2019), we use both "BASE" and "LARGE" models; these models differ by their number of layers (12 vs. 24), hidden dimensions (768 vs. 1024), attention heads (12 vs. 16) and, as a result, their number of parameters (110M vs. 340M).

- Masking strategy: the objective function used to train a CamemBERT model is a masked language model objective. However, BERT-like architectures like CamemBERT rely on a fixed vocabulary of explicitly predefined size obtained by an algorithm that splits rarer words into subwords, which are part of the vocabulary together with more frequent words. As a result, it is possible to use a whole-word masked language objective (the model is trained to guess missing words, which might be made of more than one subword) or a subword masked language objective (the model is trained to guess missing subwords). Our models use the acronyms WWM and SWM respectively to indicate the type of masking they used.

We use these word vector representations in three types of architectures:

- Fine-tuning architectures: in this case, we add a dedicated linear layer to the first subword token of each word, and the whole architecture is then fine-tuned to the NER task on the training data.

- Embedding architectures: word vectors produced by language models are used as word embeddings. We use such embeddings in two types of LSTM-based architectures: an LSTM fed to a seq2seq layer and an LSTM fed to a CRF layer. In such configurations, the use of several word representations at the same time is possible, using concatenation as a combination operator. For instance, in Table 10.1, the model FastText + CamemBERT$_{\text{OSCAR-BASE-WWM}}$ under the header "*LSTM-CRF + embeddings* corresponds to a model using the LSTM-CRF architecture and, as embeddings, the concatenation of FastText embeddings, the output of the CamemBERT "BASE" model trained on OSCAR with a whole-word masking objective, and the output of the FrELMo language model.

For our neural models, we optimized hyperparameters using F1-score on development set as our convergence metric.

We train each model three times with three different seeds, select the best seed on the development set, and report the results of this seed on the test set in Table 10.1.

Results

Word Embeddings: Results obtained by SEM and by our neural models are shown in table 10.1. First important result that should be noted is that LSTM+CRF and LSTM+seq2seq models have similar performances to that of the SEM (CRF) baseline

when they are not augmented with any kind of embeddings. Just adding classical fastText word embeddings dramatically increases the performance of the model.

ELMo Embeddings: Adding contextualized ELMo embeddings increases again the performance for both architectures. However we note that the difference is not as big as in the case of the pair with/without fastText word embeddings for the LSTM-CRF. For the seq2seq model, it is the contrary: adding ELMo gives a good improvement while fastText does not improve the results as much.

CamemBERT Embeddings: Adding the CamemBERT embeddings always increases the performance of the model LSTM based models. However, as opposed to adding ELMo, the difference with/without CamemBERT is equally considerable for both the LSTM-seq2seq and LSTM-CRF. In fact adding CamemBERT embeddings increases the original scores far more than ELMo embeddings does, so much so that the state-of-the-art model is the LSTM + CRF + FastText + $\text{CamemBERT}_{\text{OSCAR-BASE-SWM}}$.

CamemBERT + FrELMo: Contrary to the results given in Straková et al. (2019), adding ELMo to CamemBERT did not have a positive impact on the performances of the models. Our hypothesis for these results is that, contrary to Straková et al. (2019), we trained ELMo and CamemBERT on the same corpus. We think that, in our case, ELMo either does not bring any new information or even interfere with CamemBERT.

Base vs large: an interesting observation is that using large model negatively impacts the performances of the models. One possible reason could be that, because the models are larger, the information is more sparsely distributed and that training on the FTB-NE, a relatively small corpus, is harder.

Impact of shuffling the data

One important thing about the FTB is that the underlying text is made of articles from the newspaper Le Monde that are chronologically ordered. Moreover, the standard development and test sets are at the end of the corpus, which means that they are made of articles that are more recent than those found in the training set. This means that a lot of entities in the development and test sets may be new and therefore unseen in the training set. To estimate the impact of this distribution, we shuffled the data, created a new training/development/test split of the same lengths than in the standard split, and retrained and reevaluated our models. We repeated this process 3 times to avoid unexpected biases. The raw results of this experiment are given in table 10.2. We can see that the shuffled splits result in improvements on all metrics, the improvement in F1-score on the test set ranging from 4.04 to 5.75 (or

25% to 35% error reduction) for our SEM baseline, and from 1.73 to 3.21 (or 18% to 30% error reduction) for our LSTM-CRF architectures, reaching scores comparable to the English state-of-the-art. This highlights a specific difficulty of the FTB-NE corpus where the development and test sets seem to contain non-negligible amounts of unknown entities. This specificity, however, allows to have a quality estimation which is more in line with real use cases, where unknown NEs are frequent. This is especially the case when processing newly produced texts with models trained on FTB-NE, as the text annotated in the FTB is made of articles around 20 years old.

10.4 Conclusion

In this article, we introduce a new, more usable version of the named entity annotation layer of the French TreeBank. We aligned the named entity annotation to reference segmentation, which will allow to better integrate NER into the UD version of the FTB.

We establish a new state-of-the-art for French NER using state-of-the-art neural techniques and recently produced neural language models for French. Our best neural model reaches an F1-score which is 6.55 points higher (a 40% error reduction) than the strong baseline provided by the SEM system.

We also highlight how the FTB-NE is a good approximation of a real use case. Its chronological partition increases the number of unseen entities allows to have a better estimation of the generalisation capacities of machine learning models than if it were randomised.

Integration of the NER annotations in the UD version of FTB would allow to train more refined model, either by using more information or through multitask learning by learning POS and NER at the same time. We could also use dependency relationships to provide additional information to a NE linking algorithm.

One interesting point to investigate is that using Large embeddings overall has a negative impact on the models performances. It could be because larger models store information relevant to NER more sparingly, making it harder for trained models to capitalize them. We would like to investigate this hypothesis in future research.

Acknowledgments

This work was partly funded by the French national ANR grant BASNUM (ANR-18-CE38-0003), as well as by the last author's chair in the PRAIRIE institute,[10] funded by the French national ANR as part of the "Investissements d'avenir" programme under the reference ANR-19-P3IA-0001. The authors are grateful to Inria Sophia Antipolis - Méditerranée "Nef" [11] computation cluster for providing resources and support.

[10] http://prairie-institute.fr/
[11] https://wiki.inria.fr/wikis/ClustersSophia

11 D'Alembert

12 BERTrade

Part IV

Evaluation

13 Language Modeling

13.1 Introduction

One of the key elements that has pushed the state of the art considerably in neural NLP in recent years has been the introduction and spread of transfer learning methods to the field. These methods can normally be classified in two categories according to how they are used:

- *Feature-based* methods, which involve pre-training real-valued vectors ("embeddings") at the word, sentence, or paragraph level; and using them in conjunction with a specific architecture for each individual downstream task.
- *Fine-tuning* methods, which introduce a minimal number of task-specific parameters, and instead copy the weights from a pre-trained network and then tune them to a particular downstream task.

Embeddings or language models can be divided into *fixed*, meaning that they generate a single representation for each word in the vocabulary; and *contextualized*, meaning that a representation is generated based on both the word and its surrounding context, so that a single word can have multiple representations, each one depending on how it is used.

In practice, most fixed embeddings are used as feature-based models. The most notable examples are *word2vec* (Mikolov et al., 2013), *GloVe* (Pennington et al., 2014) and *fastText* (Mikolov et al., 2018). All of them are extensively used in a variety of applications nowadays. On the other hand, contextualized word representations and language models have been developed using both feature-based architectures, the most notable examples being ELMo and Flair (Peters et al., 2018; Akbik et al., 2018), and transformer based architectures, that are commonly used in a fine-tune setting, as is the case of GPT-1, GPT-2 (Radford et al., 2018, 2019), BERT and its derivatives (Devlin et al., 2019; Liu et al., 2019; Lan et al., 2020) and more recently T5 (Raffel et al., 2020). All of them have repeatedly improved the state-of-the art in many downstream NLP tasks over the last year.

In general, the main advantage of using language models is that they are mostly built in an *unsupervised* manner and they can be trained with raw, unannotated plain text. Their main drawback is that enormous quantities of data seem to be required to properly train them especially in the case of contextualized models, for which larger corpora are thought to be needed to properly address polysemy and cover the wide range of uses that commonly exist within languages.

For gathering data in a wide range of languages, Wikipedia is a commonly used option. It has been used to train fixed embeddings (Al-Rfou' et al., 2013; Bojanowski et al., 2017) and more recently the multilingual BERT (Devlin et al., 2019), hereafter mBERT. However, for some languages, Wikipedia might not be large enough to train good quality contextualized word embeddings. Moreover, Wikipedia data all belong to the same specific genre and style. To address this problem, one can

resort to crawled text from the internet; the largest and most widespread dataset of crawled text being Common Crawl.[1] Such an approach generally solves the quantity and genre/style coverage problems but might introduce noise in the data, an issue which has earned the corpus some criticism, most notably by Trinh and Le (2018) and Radford et al. (2019). Using Common Crawl also leads to data management challenges as the corpus is distributed in the form of a large set of plain text each containing a large quantity of unclassified multilingual documents from different websites.

In this paper we study the trade-off between quantity and quality of data for training contextualized representations. To this end, we use the OSCAR corpus (Ortiz Suárez et al., 2019), a freely available[2] multilingual dataset obtained by performing language classification, filtering and cleaning of the whole Common Crawl corpus.[3] OSCAR was created following the approach of Grave et al. (2018) but proposing a simple improvement on their filtering method. We then train OSCAR-based and Wikipedia-based ELMo contextualized word embeddings (Peters et al., 2018) for 5 languages: Bulgarian, Catalan, Danish, Finnish and Indonesian. We evaluate the models by attaching them to the to UDPipe 2.0 architecture (Straka, 2018; Straka et al., 2019) for dependency parsing and part-of-speech (POS) tagging. We show that the models using the OSCAR-based ELMo embeddings consistently outperform the Wikipedia-based ones, suggesting that big high-coverage noisy corpora might be better than small high-quality narrow-coverage corpora for training contextualized language representations[4]. We also establish a new state of the art for both POS tagging and dependency parsing in 6 different treebanks covering all 5 languages.

The structure of the paper is as follows. In Section 2 we describe the recent related work. In Section 3 we present, compare and analyze the corpora used to train our contextualized embeddings, and the treebanks used to train our POS tagging and parsing models. In Section 4 we examine and describe in detail the model used for our contextualized word representations, as well as the parser and the tagger we chose to evaluate the impact of corpora in the embeddings' performance in downstream tasks. Finally we provide an analysis of our results in Section 5 and in Section 6 we present our conclusions.

13.2 Related work

Since the introduction of *word2vec* (Mikolov et al., 2013), many attempts have been made to create multilingual language representations; for fixed word embeddings the most remarkable works are those of (Al-Rfou' et al., 2013) and (Bojanowski

[1] https://commoncrawl.org
[2] https://oscar-corpus.com
[3] Snapshot from November 2018
[4] Both the Wikipedia- and the OSCAR-based embeddings for these 5 languages are available at: https://oscar-corpus.com/#models.

et al., 2017) who created word embeddings for a large quantity of languages using Wikipedia, and later (Grave et al., 2018) who trained the fastText word embeddings for 157 languages using Common Crawl and who in fact showed that using crawled data significantly increased the performance of the embeddings especially for mid- to low-resource languages.

Regarding contextualized models, the most notable non-English contribution has been that of the mBERT (Devlin et al., 2019), which is distributed as (i) a single multilingual model for 100 different languages trained on Wikipedia data, and as (ii) a single multilingual model for both Simplified and Traditional Chinese. Four monolingual fully trained ELMo models have been distributed for Japanese, Portuguese, German and Basque[5]; 44 monolingual ELMo models[6] where also released by the *HIT-SCIR* team (Che et al., 2018) during the *CoNLL 2018 Shared Task* (Zeman et al., 2018), but their training sets where capped at 20 million words. A German BERT (Chan et al., 2019) as well as a French BERT model (called CamemBERT) (Martin et al., 2020) have also been released. In general no particular effort in creating a set of high-quality monolingual contextualized representations has been shown yet, or at least not on a scale that is comparable with what was done for fixed word embeddings.

For dependency parsing and POS tagging the most notable non-English specific contribution is that of the *CoNLL 2018 Shared Task* (Zeman et al., 2018), where the 1st place (LAS Ranking) was awarded to the *HIT-SCIR* team (Che et al., 2018) who used Dozat and Manning (2017)'s *Deep Bi-affine parser* and its extension described in (Dozat et al., 2017), coupled with deep contextualized ELMo embeddings (Peters et al., 2018) (capping the training set at 20 million words). The 1st place in universal POS tagging was awarded to Smith et al. (2018) who used two separate instances of Bohnet et al. (2018)'s tagger.

More recent developments in POS tagging and parsing include those of Straka et al. (2019) which couples another CoNLL 2018 shared task participant, UDPipe 2.0 (Straka, 2018), with mBERT greatly improving the scores of the original model, and UDify (Kondratyuk and Straka, 2019), which adds an extra attention layer on top of mBERT plus a Deep Bi-affine attention layer for dependency parsing and a Softmax layer for POS tagging. UDify is actually trained by concatenating the training sets of 124 different UD treebanks, creating a single POS tagging and dependency parsing model that works across 75 different languages.

13.3 Corpora

We train ELMo contextualized word embeddings for 5 languages: Bulgarian, Catalan, Danish, Finnish and Indonesian. We train one set of embeddings using only

[5] https://allennlp.org/elmo
[6] https://github.com/HIT-SCIR/ELMoForManyLangs

Language	Size	#Ktokens	#Kwords	#Ksentences
Bulgarian	609M	64,190	54,748	3,685
Catalan	1.1G	211,627	179,108	8,293
Danish	338M	60,644	52,538	3,226
Finnish	669M	89,580	76,035	6,847
Indonesian	488M	80,809	68,955	4,298

Table 13.1: Size of Wikipedia corpora, measured in bytes, thousands of tokens, words and sentences.

Wikipedia data, and another set using only Common-Crawl-based OSCAR data. We chose these languages primarily because they are morphologically and typologically different from one another, but also because all of the OSCAR datasets for these languages were of a sufficiently manageable size such that the ELMo pre-training was doable in less than one month. Contrary to *HIT-SCIR* team (Che et al., 2018), we do not impose any cap on the amount of data, and instead use the entirety of Wikipedia or OSCAR for each of our 5 chosen languages.

13.3.1 Wikipedia

Wikipedia is the biggest online multilingual open encyclopedia, comprising more than 40 million articles in 301 different languages. Because articles are curated by language and written in an open collaboration model, its text tends to be of very high-quality in comparison to other free online resources. This is why Wikipedia has been extensively used in various NLP applications (Wu and Weld, 2010; Mihalcea, 2007; Al-Rfou' et al., 2013; Bojanowski et al., 2017). We downloaded the XML Wikipedia dumps[7] and extracted the plain-text from them using the `wikiextractor.py` script[8] from Giuseppe Attardi. We present the number of words and tokens available for each of our 5 languages in Table 13.1. We decided against deduplicating the Wikipedia data as the corpora are already quite small. We tokenize the 5 corpora using *UDPipe* (Straka and Straková, 2017).

13.3.2 OSCAR

Common Crawl is a non-profit organization that produces and maintains an open, freely available repository of crawled data from the web. Common Crawl's complete archive consists of petabytes of monthly snapshots collected since 2011. Common Crawl snapshots are not classified by language, and contain a certain level of noise (e.g. one-word "sentences" such as "OK" and "Cancel" are unsurprisingly very frequent).

[7] XML dumps from April 4, 2019.
[8] Available here.

Language	Size	#Ktokens	#Kwords	#Ksentences
Bulgarian	14G	1,466,051	1,268,115	82,532
Catalan	4.3G	831,039	729,333	31,732
Danish	9.7G	1,828,881	1,620,091	99,766
Finnish	14G	1,854,440	1,597,856	142,215
Indonesian	16G	2,701,627	2,394,958	140,138

Table 13.2: Size of OSCAR subcorpora, measured in bytes, thousands of tokens, words and sentences.

This is what motivated the creation of the freely available multilingual OSCAR corpus (Ortiz Suárez et al., 2019), extracted from the November 2018 snapshot, which amounts to more than 20 terabytes of plain-text. In order to create OSCAR from this Common Crawl snapshot, Ortiz Suárez et al. (2019) reproduced the pipeline proposed by (Grave et al., 2018) to process, filter and classify Common Crawl. More precisely, language classification was performed using the *fastText* linear classifier (Joulin et al., 2016; Joulin et al., 2017), which was trained by Grave et al. (2018) to recognize 176 languages and was shown to have an extremely good accuracy to processing time trade-off. The filtering step as performed by Grave et al. (2018) consisted in only keeping the lines exceeding 100 bytes in length.[9] However, considering that Common Crawl is a mutilingual UTF-8 encoded corpus, this 100-byte threshold creates a huge disparity between ASCII and non-ASCII encoded languages. The filtering step used to create OSCAR therefore consisted in only keeping the lines containing at least 100 UTF-8-encoded characters. Finally, as in (Grave et al., 2018), the OSCAR corpus is deduplicated, i.e. for each language, only one occurrence of a given line is included.

As we did for Wikipedia, we tokenize OSCAR corpora for the 5 languages we chose for our study using UDPipe. Table 13.2 provides quantitative information about the 5 resulting tokenized corpora.

We note that the original Common-Crawl-based corpus created by Grave et al. (2018) to train fastText is not freely available. Since running the experiments described in this paper, a new architecture for creating a Common-Crawl-based corpus named CCNet (Wenzek et al., 2020) has been published, although it includes specialized filtering which might result in a cleaner corpus compared to OSCAR, the resulting CCNet corpus itself was not published. Thus we chose to keep OSCAR as it remains the only very large scale, Common-Crawl-based corpus currently available and easily downloadable.

[9] Script available here.

Language	OOV Wikipedia	OOV OSCAR
Bulgarian	60,879	66,558
Catalan	34,919	79,678
Danish	134,677	123,299
Finnish	266,450	267,525
Indonesian	116,714	124,607

Table 13.3: Number of out-of-vocabulary words in random samples of 1M words for OSCAR and Wikipedia.

13.3.3 Noisiness

We wanted to address (Trinh and Le, 2018) and (Radford et al., 2019)'s criticisms of Common Crawl, so we devised a simple method to measure how noisy the OSCAR corpora were for our 5 languages. We randomly extract a number of lines from each corpus, such that the resulting random sample contains one million words.[10] We test if the words are in the corresponding *GNU Aspell*[11] dictionary. We repeat this task for each of the 5 languages, for both the OSCAR and the Wikipedia corpora. We compile in Table 13.3 the number of out-of-vocabulary tokens for each corpora.

As expected, this simple metric shows that in general the OSCAR samples contain more out-of-vocabulary words than the Wikipedia ones. However the difference in magnitude between the two is strikingly lower that one would have expected in view of the criticisms by Trinh and Le (2018) and Radford et al. (2019), thereby validating the usability of Common Crawl data when it is properly filtered, as was achieved by the OSCAR creators. We even observe that, for Danish, the number of out-of-vocabulary words in OSCAR is lower than that in Wikipedia.

13.4 Experimental Setting

The main goal of this paper is to show the impact of training data on contextualized word representations when applied in particular downstream tasks. To this end, we train different versions of the *Embeddings from Language Models* (ELMo) (Peters et al., 2018) for both the Wikipedia and OSCAR corpora, for each of our selected 5 languages. We save the models' weights at different number of epochs for each language, in order to test how corpus size affect the embeddings and to see whether and when overfitting happens when training elmo on smaller corpora.

We take each of the trained ELMo models and use them in conjunction with the UDPipe 2.0 (Straka, 2018; Straka et al., 2019) architecture for dependency parsing

[10] We remove tokens that are capitalized or contain less than 4 UTF-8 encoded characters, allowing us to remove bias against Wikipedia, which traditionally contains a large quantity of proper nouns and acronyms.

[11] http://aspell.net/

and POS-tagging to test our models. We train UDPipe 2.0 using gold tokenization and segmentation for each of our ELMo models, the only thing that changes from training to training is the ELMo model as hyperparameters always remain at the default values (except for number of training tokens) (Peters et al., 2018).

13.4.1 Contextualized word embeddings

Embeddings from Language Models (ELMo) (Peters et al., 2018) is an LSTM-based language model. More precisely, it uses a bidirectional language model, which combines a forward and a backward LSTM-based language model. ELMo also computes a context-independent token representation via a CNN over characters.

We train ELMo models for Bulgarian, Catalan, Danish, Finnish and Indonesian using the OSCAR corpora on the one hand and the Wikipedia corpora on the other. We train each model for 10 epochs, as was done for the original English ELMo (Peters et al., 2018). We save checkpoints at 1st, 3rd and 5th epoch in order to investigate some concerns about possible overfitting for smaller corpora (Wikipedia in this case) raised by the original ELMo authors.[12]

13.4.2 UDPipe 2.0

For our POS tagging and dependency parsing evaluation, we use UDPipe 2.0, which has a freely available and ready to use implementation.[13] This architecture was submitted as a participant to the *2018 CoNLL Shared Task* (Zeman et al., 2018), obtaining the 3rd place in LAS ranking. UDPipe 2.0 is a multi-task model that predicts POS tags, lemmas and dependency trees jointly.

The original UDPipe 2.0 implementation calculates 3 different embeddings, namely:

- *Pre-trained word embeddings*: In the original implementation, the Wikipedia version of fastText embeddings is used (Bojanowski et al., 2017); we replace them in favor of the newer Common-Crawl-based fastText embeddings trained by Grave et al. (2018).
- *Trained word embeddings*: Randomly initialized word representations that are trained with the rest of the network.
- *Character-level word embeddings*: Computed using bi-directional GRUs of dimension 256. They represent every UTF-8 encoded character with two 256 dimensional vectors, one for the forward and one for the backward layer. This two vector representations are concatenated and are trained along the whole network.

[12] https://github.com/allenai/bilm-tf/issues/135
[13] https://github.com/CoNLL-UD-2018/UDPipe-Future

Treebank	#Ktokens	#Ksentences
Bulgarian-BTB	156	11
Catalan-AnCora	530	17
Danish-DDT	100	6
Finnish-FTB	159	19
Finnish-TDT	202	15
Indonesian-GSD	121	6

Table 13.4: Size of treebanks, measured in thousands of tokens and sentences.

After the CoNLL 2018 Shared Task, the UDPipe 2.0 authors added the option to concatenate contextualized representations to the embedding section of the network (Straka et al., 2019), we use this new implementation and we concatenate our pre-trained deep contextualized ELMo embeddings to the three embeddings mentioned above.

Once the embedding step is completed, the concatenation of all vector representations for a word are fed to two shared bidirectional LSTM (Hochreiter and Schmidhuber, 1997) layers. The output of these two BiLSTMS is then fed to two separate specific LSTMs:

- The tagger- and lemmatizer-specific bidirectional LSTMs, with Softmax classifiers on top, which process its output and generate UPOS, XPOS, UFeats and Lemmas. The lemma classifier also takes the character-level word embeddings as input.
- The parser-specific bidirectional LSTM layer, whose output is then passed to a bi-affine attention layer (Dozat and Manning, 2017) producing labeled dependency trees.

13.4.3 Treebanks

To train the selected parser and tagger (cf. Section 13.4.2) and evaluate the pre-trained language models in our 5 languages, we run our experiments using the Universal Dependencies (UD)[14] paradigm and its corresponding UD POS tag set (Petrov et al., 2012). We use all the treebanks available for our five languages in the UD treebank collection version 2.2 (Nivre et al., 2018), which was used for the CoNLL 2018 shared task, thus we perform our evaluation tasks in 6 different treebanks (see Table 13.4 for treebank size information).

- *Bulgarian BTB*: Created at the Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, it consists of legal documents, news articles and fiction pieces.

[14] https://universaldependencies.org

- *Catalan-AnCora*: Built on top of the Spanish-Catalan *AnCora corpus* (Taulé et al., 2008), it contains mainly news articles.
- *Danish-DDT*: Converted from the *Danish Dependency Treebank* (Buch-Kromann, 2003). It includes news articles, fiction and non fiction texts and oral transcriptions.
- *Finnish-FTB*: Consists of manually annotated grammatical examples from VISK[15] (The Web Version of the Large Grammar of Finnish).
- *Finnish-TDT*: Based on the Turku Dependency Treebank (TDT). Contains texts from Wikipedia, Wikinews, news articles, blog entries, magazine articles, grammar examples, Europarl speeches, legal texts and fiction.
- *Indonesian-GSD*: Includes mainly blog entries and news articles.

13.5 Results & Discussion

13.5.1 Parsing and POS tagging results

We use UDPipe 2.0 without contextualized embeddings as our baseline for POS tagging and dependency parsing. However, we did not train the model without contextualized word embedding ourselves. We instead take the scores as they are reported in (Kondratyuk and Straka, 2019). We also compare our UDPipe 2.0 + ELMo models against the state-of-the-art results (assuming gold tokenization) for these languages, which are either UDify (Kondratyuk and Straka, 2019) or UDPipe 2.0 + mBERT (Straka et al., 2019).

Results for UPOS, UAS and LAS are shown in Table 13.5. We obtain the state of the art for the three metrics in each of the languages with the UDPipe 2.0 + $\text{ELMo}_{\text{OSCAR}}$ models. We also see that in every single case the UDPipe 2.0 + $\text{ELMo}_{\text{OSCAR}}$ result surpasses the UDPipe 2.0 + $\text{ELMo}_{\text{Wikipedia}}$ one, suggesting that the size of the pre-training data plays an important role in downstream task results. This is also supports our hypothesis that the OSCAR corpora, being multi-domain, exhibits a better coverage of the different styles, genres and uses present at least in these 5 languages.

Taking a closer look at the results for Danish, we see that $\text{ELMo}_{\text{Wikipedia}}$, which was trained with a mere 300MB corpus, does not show any sign of overfitting, as the UDPipe 2.0 + $\text{ELMo}_{\text{Wikipedia}}$ results considerably improve the UDPipe 2.0 baseline. This is the case for all of our $\text{ELMo}_{\text{Wikipedia}}$ models as we never see any evidence of a negative impact when we add them to the baseline model. In fact, the results of UDPipe 2.0 + $\text{ELMo}_{\text{Wikipedia}}$ give better than previous state-of-the-art results in all metrics for the Finnish-FTB and in UPOS for the Finnish-TDT. The results for Finnish are actually quite interesting, as mBERT was pre-trained on Wikipedia and here we

[15] http://scripta.kotus.fi/visk

Treebank	Model	UPOS	UAS	LAS
Bulgarian BTB	UDify	98.89	95.54	92.40
	UDPipe 2.0	98.98	93.38	90.35
	+mBERT	99.20	95.34	92.62
	+ELMo$_{\text{Wikipedia}}$	99.17	94.93	92.05
	+ELMo$_{\text{OSCAR}}$	**99.40**	**96.01**	**93.56**
Catalan-AnCora	UDify	98.89	94.25	92.33
	UDPipe 2.0	98.88	93.22	91.06
	+mBERT	**99.06**	**94.49**	92.74
	+ELMo$_{\text{Wikipedia}}$	99.05	93.99	92.24
	+ELMo$_{\text{OSCAR}}$	**99.06**	**94.49**	**92.88**
Danish-DDT	UDify	97.50	87.76	84.50
	UDPipe 2.0	97.78	86.88	84.31
	+mBERT	98.21	89.32	87.24
	+ELMo$_{\text{Wikipedia}}$	98.45	89.05	86.92
	+ELMo$_{\text{OSCAR}}$	**98.62**	**89.84**	**87.95**
Finnish-FTB	UDify	93.80	86.37	81.40
	UDPipe 2.0	96.65	90.68	87.89
	+mBERT	96.97	91.68	89.02
	+ELMo$_{\text{Wikipedia}}$	97.27	92.05	89.62
	+ELMo$_{\text{OSCAR}}$	**98.13**	**93.81**	**92.02**
Finnish-TDT	UDify	94.43	86.42	82.03
	UDPipe 2.0	97.45	89.88	87.46
	+mBERT	97.57	91.66	89.49
	+ELMo$_{\text{Wikipedia}}$	97.65	91.60	89.34
	+ELMo$_{\text{OSCAR}}$	**98.36**	**93.54**	**91.77**
Indonesian-GSD	UDify	93.36	86.45	80.10
	UDPipe 2.0	93.69	85.31	78.99
	+mBERT	94.09	86.47	80.40
	+ELMo$_{\text{Wikipedia}}$	93.94	86.16	80.10
	+ELMo$_{\text{OSCAR}}$	**94.12**	**86.49**	**80.59**

Table 13.5: Scores from UDPipe 2.0 (from Kondratyuk and Straka, 2019), the previous state-of-the-art models UDPipe 2.0+mBERT (Straka et al., 2019) and UDify (Kondratyuk and Straka, 2019), and our ELMo-enhanced UDPipe 2.0 models. Test scores are given for UPOS, UAS and LAS in all five languages. Best scores are shown in bold, second best scores are underlined.

see that the multilingual setting in which UDify was fine-tuned exhibits sub-baseline results for all metrics, and that the UDPipe + mBERT scores are often lower than those of our UDPipe 2.0 + ELMo$_{\text{Wikipedia}}$. This actually suggests that even though the multilingual approach of mBERT (in pre-training) or UDify (in pre-training and fine-tuning) leads to better performance for high-resource languages or languages that are closely related to high-resource languages, it might also significantly degrade the representations for more isolated or even simply more morphologically rich

Treebank	Model	UPOS	UAS	LAS
Bulgarian BTB	UDPipe 2.0	98.98	93.38	90.35
	+ELMo$_{\text{Wikipedia}(1)}$	98.81	93.60	90.21
	+ELMo$_{\text{Wikipedia}(3)}$	99.01	94.32	91.36
	+ELMo$_{\text{Wikipedia}(5)}$	99.03	94.32	91.38
	+ELMo$_{\text{Wikipedia}(10)}$	99.17	94.93	92.05
	+ELMo$_{\text{OSCAR}(1)}$	99.28	95.45	92.98
	+ELMo$_{\text{OSCAR}(3)}$	99.34	95.58	93.12
	+ELMo$_{\text{OSCAR}(5)}$	99.34	95.63	93.25
	+ELMo$_{\text{OSCAR}(10)}$	**99.40**	**96.01**	**93.56**
Catalan-AnCora	UDPipe 2.0	98.88	93.22	91.06
	+ELMo$_{\text{Wikipedia}(1)}$	98.93	93.24	91.21
	+ELMo$_{\text{Wikipedia}(3)}$	99.02	93.75	91.93
	+ELMo$_{\text{Wikipedia}(5)}$	99.04	93.86	92.05
	+ELMo$_{\text{Wikipedia}(10)}$	99.05	93.99	92.24
	+ELMo$_{\text{OSCAR}(1)}$	99.07	93.92	92.29
	+ELMo$_{\text{OSCAR}(3)}$	**99.10**	94.29	92.69
	+ELMo$_{\text{OSCAR}(5)}$	99.07	94.38	92.75
	+ELMo$_{\text{OSCAR}(10)}$	99.06	**94.49**	**92.88**
Danish-DDT	UDPipe 2.0	97.78	86.88	84.31
	+ELMo$_{\text{Wikipedia}(1)}$	97.47	86.98	84.15
	+ELMo$_{\text{Wikipedia}(3)}$	98.03	88.16	85.81
	+ELMo$_{\text{Wikipedia}(5)}$	98.15	88.24	85.96
	+ELMo$_{\text{Wikipedia}(10)}$	98.45	89.05	86.92
	+ELMo$_{\text{OSCAR}(1)}$	98.50	89.47	87.43
	+ELMo$_{\text{OSCAR}(3)}$	98.59	89.68	87.77
	+ELMo$_{\text{OSCAR}(5)}$	98.59	89.46	87.64
	+ELMo$_{\text{OSCAR}(10)}$	**98.62**	**89.84**	**87.95**

Treebank	Model	UPOS	UAS	LAS
Finnish-FTB	UDPipe 2.0	96.65	90.68	87.89
	+ELMo$_{\text{Wikipedia}(1)}$	95.86	89.63	86.39
	+ELMo$_{\text{Wikipedia}(3)}$	96.76	91.02	88.27
	+ELMo$_{\text{Wikipedia}(5)}$	96.97	91.66	89.04
	+ELMo$_{\text{Wikipedia}(10)}$	97.27	92.05	89.62
	+ELMo$_{\text{OSCAR}(1)}$	97.91	93.41	91.43
	+ELMo$_{\text{OSCAR}(3)}$	98.00	**93.99**	91.98
	+ELMo$_{\text{OSCAR}(5)}$	**98.15**	93.98	**92.24**
	+ELMo$_{\text{OSCAR}(10)}$	98.13	93.81	92.02
Finnish-TDT	UDPipe 2.0	97.45	89.88	87.46
	+ELMo$_{\text{Wikipedia}(1)}$	96.73	89.11	86.33
	+ELMo$_{\text{Wikipedia}(3)}$	97.55	90.84	88.50
	+ELMo$_{\text{Wikipedia}(5)}$	97.55	91.11	88.88
	+ELMo$_{\text{Wikipedia}(10)}$	97.65	91.60	89.34
	+ELMo$_{\text{OSCAR}(1)}$	98.27	93.03	91.29
	+ELMo$_{\text{OSCAR}(3)}$	98.38	**93.60**	**91.83**
	+ELMo$_{\text{OSCAR}(5)}$	**98.39**	93.57	91.80
	+ELMo$_{\text{OSCAR}(10)}$	98.36	93.54	91.77
Indonesian-GSD	UDPipe 2.0	93.69	85.31	78.99
	+ELMo$_{\text{Wikipedia}(1)}$	93.70	85.81	79.46
	+ELMo$_{\text{Wikipedia}(3)}$	93.90	86.04	79.72
	+ELMo$_{\text{Wikipedia}(5)}$	94.04	85.93	79.97
	+ELMo$_{\text{Wikipedia}(10)}$	93.94	86.16	80.10
	+ELMo$_{\text{OSCAR}(1)}$	93.95	86.25	80.23
	+ELMo$_{\text{OSCAR}(3)}$	94.00	86.21	80.14
	+ELMo$_{\text{OSCAR}(5)}$	**94.23**	86.37	80.40
	+ELMo$_{\text{OSCAR}(10)}$	94.12	**86.49**	**80.59**

Table 13.6: UPOS, UAS and LAS scores for the UDPipe 2.0 baseline reported by (Kondratyuk and Straka, 2019), plus the scores for checkpoints at 1, 3, 5 and 10 epochs for all the ELMo$_{\text{OSCAR}}$ and ELMo$_{\text{Wikipedia}}$. All scores are test scores. Best ELMo$_{\text{OSCAR}}$ scores are shown in bold while best ELMo$_{\text{Wikipedia}}$ scores are underlined.

languages like Finnish. In contrast, our monolingual approach with UDPipe 2.0 + $\text{ELMo}_{\text{OSCAR}}$ improves the previous SOTA considerably, by more than 2 points for some metrics. Note however that Indonesian, which might also be seen as a relatively isolated language, does not behave in the same way as Finnish.

13.5.2 Impact of the number of training epochs

An important topic we wanted to address with our experiments was that of *overfitting* and the number of epochs one should train the contextualized embeddings for. The ELMo authors have expressed that increasing the number of training epochs is generally better, as they argue that training the ELMo model for longer reduces held-out perplexity and further improves downstream task performance.[16] This is why we intentionally fully pre-trained the $\text{ELMo}_{\text{Wikipedia}}$ to the 10 epochs of the original ELMo paper, as its authors also expressed concern over the possibility of overfitting for smaller corpora. We thus save checkpoints for each of our ELMo model at the 1, 3, 5 and 10 epoch marks so that we can properly probe for overfitting. The scores of all checkpoints are reported in Table 13.6. Here again we do not train the UDPipe 2.0 baselines without embedding, we just report the scores published in Kondratyuk and Straka (2019).

The first striking finding is that even though all our Wikipedia data sets are smaller than 1GB in size (except for Catalan), none of the $\text{ELMo}_{\text{Wikipedia}}$ models show any sign of overfitting, as the results continue to improve for all metrics the more we train the ELMo models, with the best results consistently being those of the fully trained 10 epoch ELMos. For all of our Wikipedia models, but those of Catalan and Indonesian, we see sub-baseline results at 1 epoch; training the model for longer is better, even if the corpora are small in size.

$\text{ELMo}_{\text{OSCAR}}$ models exhibit exactly the same behavior as $\text{ELMo}_{\text{Wikipedia}}$ models where the scores continue to improve the longer they are pre-trained, except for the case of Finnish. Here we actually see an unexpected behavior where the model performance caps around the 3^{rd} to 5^{th} epoch. This is surprising because the Finnish OSCAR corpus is more than 20 times bigger than our smallest Wikipedia corpus, the Danish Wikipedia, that did not exhibit this behavior. As previously mentioned Finnish is morphologically richer than the other languages in which we trained ELMo, we hypothesize that the representation space given by the ELMo embeddings might not be sufficiently big to extract more features from the Finnish OSCAR corpus beyond the 5^{th} epoch mark, however in order to test this we would need to train a larger language model like BERT which is sadly beyond our computing infrastructure limits (cf. Subsection 13.5.3). However we do note that pre-training our current language model architectures in a morphologically rich language like Finnish might actually better expose the limits of our existing approaches to language modeling.

[16] Their comments on the matter can be found here.

Language	Power	Hours	Days	KWh·PUE	CO_2e
OSCAR-Based ELMos					
Bulgarian	1183	515.00	21.45	962.61	49.09
Catalan	1118	199.98	8.33	353.25	18.02
Danish	1183	200.89	8.58	375.49	19.15
Finnish	1118	591.25	24.63	1044.40	53.26
Indonesian	1183	694.26	28.93	1297.67	66.18
Wikipedia-Based ELMos					
Bulgarian	1118	15.45	0.64	27.29	1.39
Catalan	1118	51.08	2.13	90.22	4.60
Danish	1118	14.56	0.61	25,72	1.31
Finnish	1118	21.79	0.91	38.49	1.96
Indonesian	1118	20.28	0.84	35.82	1.82
Total emissions					216.78

Table 13.7: Average power draw (Watts), training times (in both hours and days), mean power consumption (KWh) and CO_2 emissions (kg) for each ELMo model trained.

One last thing that it is important to note with respect to the number of training epochs is that even though we fully pre-trained our $\text{ELMo}_{\text{Wikipedia}}$'s and $\text{ELMo}_{\text{OSCAR}}$'s to the recommended 10 epoch mark, and then compared them against one another, the number of training steps between both pre-trained models differs drastically due to the big difference in corpus size (for Indonesian, for instance, 10 epochs correspond to 78K steps for $\text{ELMo}_{\text{Wikipedia}}$ and to 2.6M steps for OSCAR; the complete picture is provided in the Appendix, in Table 13.8). In fact, we can see in Table 13.6 that all the UDPipe 2.0 + $\text{ELMo}_{\text{OSCAR(1)}}$ perform better than the UDPipe 2.0 + $\text{ELMo}_{\text{Wikipedia(1)}}$ models across all metrics. Thus we believe that talking in terms of training steps as opposed to training epochs might be a more transparent way of comparing two pre-trained models.

13.5.3 Computational cost and carbon footprint

Considering the discussion above, we believe an interesting follow-up to our experiments would be training the ELMo models for more of the languages included in the OSCAR corpus. However training ELMo is computationally costly, and one way to estimate this cost, as pointed out by Strubell et al. (2019), is by using the training times of each model to compute both power consumption and CO_2 emissions.

In our set-up we used two different machines, each one having 4 NVIDIA GeForce GTX 1080 Ti graphic cards and 128GB of RAM, the difference between the machines being that one uses a single Intel Xeon Gold 5118 processor, while the other uses two Intel Xeon E5-2630 v4 processors. One GeForce GTX 1080 Ti card is rated at

around 250 W,[17] the Xeon Gold 5118 processor is rated at 105 W,[18] while one Xeon E5-2630 v4 is rated at 85 W.[19] For the DRAM we can use the work of Desrochers et al. (2016) to estimate the total power draw of 128GB of RAM at around 13W. Having this information, we can now use the formula proposed by Strubell et al. (2019) in order to compute the total power required to train one ELMo model:

$$p_t = \frac{1.58t(cp_c + p_r + gp_g)}{1000}$$

Where c and g are the number of CPUs and GPUs respectively, p_c is the average power draw (in Watts) from all CPU sockets, p_r the average power draw from all DRAM sockets, and p_g the average power draw of a single GPU. We estimate the total power consumption by adding GPU, CPU and DRAM consumptions, and then multiplying by the *Power Usage Effectiveness* (PUE), which accounts for the additional energy required to support the compute infrastructure. We use a PUE coefficient of 1.58, the 2018 global average for data centers (Strubell et al., 2019). In table 13.7 we report the training times in both hours and days, as well as the total power draw (in Watts) of the system used to train each individual ELMo model. We use this information to compute the total power consumption of each ELMo, also reported in table 13.7.

We can further estimate the CO_2 emissions in kilograms of each single model by multiplying the total power consumption by the average CO_2 emissions per kWh in France (where the models were trained). According to the RTE (Réseau de transport d'électricité / Electricity Transmission Network) the average emission per kWh were around 51g/kWh in November 2019,[20] when the models were trained. Thus the total CO_2 emissions in kg for one single model can be computed as:

$$\mathrm{CO_2e} = 0.051p_t$$

All emissions for the ELMo models are also reported in table 13.7.

We do not report the power consumption or the carbon footprint of training the UDPipe 2.0 architecture, as each model took less than 4 hours to train on a machine using a single NVIDIA Tesla V100 card. Also, this machine was shared during training time, so it would be extremely difficult to accurately estimate the power consumption of these models.

Even though it would have been interesting to replicate all our experiments and computational cost estimations with state-of-the-art fine-tuning models such as BERT,

[17] https://www.geforce.com/hardware/desktop-gpus/geforce-gtx-1080-ti/specifications
[18] https://ark.intel.com/content/www/us/en/ark/products/120473/intel-xeon-gold-5118-processor-16-5m-cache-2-30-ghz.html
[19] https://ark.intel.com/content/www/us/en/ark/products/92981/intel-xeon-processor-e5-2630-v4-25m-cache-2-20-ghz.html
[20] https://www.rte-france.com/fr/eco2mix/eco2mix-co2

XLNet, RoBERTa or ALBERT, we recall that these transformer-based architectures are extremely costly to train, as noted by the BERT authors on the official BERT GitHub repository,[21] and are currently beyond the scope of our computational infrastructure. However we believe that ELMo contextualized word embeddings remain a useful model that still provide an extremely good trade-off between performance to training cost, even setting new state-of-the-art scores in parsing and POS tagging for our five chosen languages, performing even better than the multilingual mBERT model.

13.6 Conclusions

In this paper, we have explored the use of the Common-Crawl-based OSCAR corpora to train ELMo contextualized embeddings for five typologically diverse mid-resource languages. We have compared them with Wikipedia-based ELMo embeddings on two classical NLP tasks, POS tagging and parsing, using state-of-the-art neural architectures. Our goal was to explore whether the noisiness level of Common Crawl data, often invoked to criticize the use of such data, could be compensated by its larger size; for some languages, the OSCAR corpus is several orders of magnitude larger than the corresponding Wikipedia. Firstly, we found that when properly filtered, Common Crawl data is not massively noisier than Wikipedia. Secondly, we show that embeddings trained using OSCAR data consistently outperform Wikipedia-based embeddings, to the extent that they allow us to improve the state of the art in POS tagging and dependency parsing for all the 6 chosen treebanks. Thirdly, we observe that more training epochs generally results in better embeddings even when the training data is relatively small, as is the case for Wikipedia.

Our experiments show that Common-Crawl-based data such as the OSCAR corpus can be used to train high-quality contextualized embeddings, even for languages for which more standard textual resources lack volume or genre variety. This could result in better performances in a number of NLP tasks for many non highly resourced languages.

Acknowledgments

We want to thank Ganesh Jawahar for his insightful comments and suggestions during the early stages of this project. This work was partly funded by the French national ANR grant BASNUM (ANR-18-CE38-0003), as well as by the last author's chair in the PRAIRIE institute,[22] funded by the French national ANR as part of the "Investissements d'avenir" programme under the reference ANR-19-P3IA-0001. The authors are grateful to Inria Sophia Antipolis - Méditerranée "Nef"[23] computation cluster for providing resources and support.

[21] https://github.com/google-research/bert
[22] http://prairie-institute.fr/
[23] https://wiki.inria.fr/wikis/ClustersSophia

13.7 Appendix

13.7.1 Number of training steps for each checkpoint and each corpus

Language	1 Epoch	3 Epochs	5 Epochs	10 Epochs
Wikipedia-Based ELMos				
Bulgarian	6,268	18,804	31,340	62,680
Catalan	20,666	61,998	103,330	206,660
Danish	5,922	17,766	29,610	59,220
Finnish	8,763	26,289	43,815	87,630
Indonesian	7,891	23,673	39,455	78,910
OSCAR-Based ELMos				
Bulgarian	143,169	429,507	715,845	1,431,690
Catalan	81,156	243,468	405,780	811,560
Danish	81,156	243,468	405,780	811,560
Finnish	181,230	543,690	906,150	1,812,300
Indonesian	263,830	791,490	1,319,150	2,638,300

Table 13.8: Number of training steps for each checkpoint, for the $\text{ELMo}_{\text{Wikipedia}}$ and $\text{ELMo}_{\text{OSCAR}}$ of each language.

14 Parsing

15 POS Tagging

16 Named-Entity Recognition

Part V

Real World Application

17 BASNUM

18 Named-Entity Recognition corpora

Bibliography

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. *Building a Treebank for French*, pages 165–187. Springer Netherlands, Dordrecht.

Željko Agić and Ivan Vulić. 2019. JW300: A wide-coverage parallel corpus for low-resource languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3204–3210, Florence, Italy. Association for Computational Linguistics.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Rami Al-Rfou', Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria. Association for Computational Linguistics.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(61):1817–1853.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. 2019. Massively Multilingual Neural Machine Translation in the Wild: Findings and Challenges. *arXiv e-prints*, page arXiv:1907.05019.

Mikel Artetxe and Holger Schwenk. 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th*

International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 5360–5369, Hong Kong, China. Association for Computational Linguistics.

Marta Bañón, Pinzhen Chen, Barry Haddow, Kenneth Heafield, Hieu Hoang, Miquel Esplà-Gomis, Mikel L. Forcada, Amir Kamran, Faheem Kirefu, Philipp Koehn, Sergio Ortiz Rojas, Leopoldo Pla Sempere, Gema Ramírez-Sánchez, Elsa Sarrías, Marek Strelec, Brian Thompson, William Waites, Dion Wiggins, and Jaume Zaragoza. 2020. ParaCrawl: Web-scale acquisition of parallel corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4555–4567, Online. Association for Computational Linguistics.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Rachel Bawden, Marie-Amélie Botalla, Kim Gerdes, and Sylvain Kahane. 2014. Correcting and validating syntactic dependency in the spoken French treebank rhapsodie. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2320–2325, Reykjavik, Iceland. European Language Resources Association (ELRA).

Frederic Bechet and Eric Charton. 2010. Unsupervised knowledge acquisition for extracting named entities from speech. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5338–5341.

Frédéric Béchet, Benoît Sagot, and Rosa Stern. 2011. Coopération de méthodes statistiques et symboliques pour l'adaptation non-supervisée d'un système d'étiquetage en entités nommées (statistical and symbolic methods cooperation for the unsupervised adaptation of a named entity recognition system). In *Actes de la 18e conférence sur le Traitement Automatique des Langues Naturelles. Articles courts*, pages 19–24, Montpellier, France. ATALA.

Emily M. Bender and Batya Friedman. 2018. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? 🦜. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Michaël Benesty. Ner algo benchmark: spacy, flair, m-bert and camembert on anonymizing french commercial legal cases [online]. 2019.

Shohini Bhattasali, Murielle Fabre, and John Hale. 2018. Processing MWEs: Neurocognitive bases of verbal MWEs and lexical cohesiveness within MWEs. In

Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018), pages 6–17, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Douglas Biber. 1993. Representativeness in Corpus Design. *Literary and Linguistic Computing*, 8(4):243–257.

Stella Biderman and Walter J. Scheirer. 2020. Pitfalls in Machine Learning Research: Reexamining the Development Cycle. *arXiv e-prints*, page arXiv:2011.02832.

Abeba Birhane and Vinay Uday Prabhu. 2021. Large image datasets: A pyrrhic win for computer vision? In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1536–1546.

Robert D. Blumofe and Charles E. Leiserson. 1999. Scheduling multithreaded computations by work stealing. *J. ACM*, 46(5):720–748.

Bernd Bohnet, Ryan McDonald, Gonçalo Simões, Daniel Andor, Emily Pitler, and Joshua Maynez. 2018. Morphosyntactic tagging with a meta-BiLSTM model over context sensitive token encodings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2642–2652, Melbourne, Australia. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Olivier Bonami and Sacha Beniamine. 2015. Implicative structure and joint predictiveness. In *Proceedings of the NetWordS Final Conference on Word Knowledge and Word Usage: Representations and Processes in the Mental Lexicon, Pisa, Italy, March 30 - April 1, 2015*, volume 1347 of *CEUR Workshop Proceedings*, pages 4–9. CEUR-WS.org.

Jan A. Botha, Emily Pitler, Ji Ma, Anton Bakalov, Alex Salcianu, David Weiss, Ryan McDonald, and Slav Petrov. 2017. Natural language processing with small feed-forward networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2879–2885, Copenhagen, Denmark. Association for Computational Linguistics.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based *n*-gram models of natural language. *Computational Linguistics*, 18(4):467–480.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Matthias Buch-Kromann. 2003. The danish dependency treebank and the dtag treebank tool. In *2nd Workshop on Treebanks and Linguistic Theories (TLT), Sweden*, pages 217–220.

Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. 2009. Clueweb09 data set.

Marie Candito and Benoît Crabbé. 2009. Improving generative statistical parsing with semi-supervised word clustering. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 138–141, Paris, France. Association for Computational Linguistics.

Marie Candito, Benoît Crabbé, and Pascal Denis. 2010. Statistical French dependency parsing: Treebank conversion and first results. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Marie Candito, Guy Perrier, Bruno Guillaume, Corentin Ribeyre, Karën Fort, Djamé Seddah, and Éric de la Clergerie. 2014. Deep syntax annotation of the sequoia French treebank. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2298–2305, Reykjavik, Iceland. European Language Resources Association (ELRA).

Marie Candito and Djamé Seddah. 2012. Le corpus sequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical (the sequoia corpus : Syntactic annotation and use for a parser lexical domain adaptation method) [in French]. In *Proceedings of the Joint Conference JEP-TALN-RECITAL 2012, volume 2: TALN*, pages 321–334, Grenoble, France. ATALA/AFCP.

Isaac Caswell, Theresa Breiner, Daan van Esch, and Ankur Bapna. 2020. Language ID in the wild: Unexpected challenges on the path to a thousand-language web text corpus. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6588–6608, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Isaac Caswell, Julia Kreutzer, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, Monang Setyawan, Supheakmungkol Sarin, Sokhar Samb, Benoît Sagot,

Clara Rivera, Annette Rios, Isabel Papadimitriou, Salomey Osei, Pedro Javier Ortiz Suárez, Iroro Orife, Kelechi Ogueji, Rubungo Andre Niyongabo, Toan Q. Nguyen, Mathias Müller, André Müller, Shamsuddeen Hassan Muhammad, Nanda Muhammad, Ayanda Mnyakeni, Jamshidbek Mirzakhalov, Tapiwanashe Matangira, Colin Leong, Nze Lawson, Sneha Kudugunta, Yacine Jernite, Mathias Jenny, Orhan Firat, Bonaventure F. P. Dossou, Sakhile Dlamini, Nisansa de Silva, Sakine Çabuk Ballı, Stella Biderman, Alessia Battisti, Ahmed Baruwa, Ankur Bapna, Pallavi Baljekar, Israel Abebe Azime, Ayodele Awokoya, Duygu Ataman, Orevaoghene Ahia, Oghenefego Ahia, Sweta Agrawal, and Mofetoluwa Adeyemi. 2021. Quality at a Glance: An Audit of Web-Crawled Multilingual Datasets. *arXiv e-prints*, page arXiv:2103.12028.

Branden Chan, Timo Möller, Malte Pietsch, Tanay Soni, and Chin Man Yeung. 2019. German bert. `https://deepset.ai/german-bert`.

Thierry Chanier, Céline Poudat, Benoît Sagot, Georges Antoniadis, Ciara R. Wigham, Linda Hriba, Julien Longhi, and Djamé Seddah. 2014. The CoMeRe corpus for French: structuring and annotating heterogeneous CMC genres. *Journal for language technology and computational linguistics*, 29(2):1–30. Final version to Special Issue of JLCL (Journal of Language Technology and Computational Linguistics (JLCL, http://jlcl.org/): BUILDING AND ANNOTATING CORPORA OF COMPUTER-MEDIATED DISCOURSE: Issues and Challenges at the Interface of Corpus and Computational Linguistics (ed. by Michael Beißwenger, Nelleke Oostdijk, Angelika Storrer & Henk van den Heuvel).

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual

sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

BNC Consortium et al. 2007. 520 million words, 1990-present. In *The British National Corpus, version 3 - BNC XML Edition*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Mark Davies. 2009. The 385+ million word corpus of contemporary american english (1990–2008+): Design, architecture, and linguistic insights. *International Journal of Corpus Linguistics*, 14(2):159–190.

Mark Davies. 2010. The Corpus of Contemporary American English as the first reliable monitor corpus of English. *Literary and Linguistic Computing*, 25(4):447–464.

Pieter Delobelle, Thomas Winters, and Bettina Berendt. 2020. RobBERT: a Dutch RoBERTa-based Language Model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3255–3265, Online. Association for Computational Linguistics.

Spencer Desrochers, Chad Paradis, and Vincent M. Weaver. 2016. A validation of dram rapl power measurements. In *Proceedings of the Second International Symposium on Memory Systems*, MEMSYS '16, page 455–470, New York, NY, USA. Association for Computing Machinery.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Martin d'Hoffschmidt, Wacim Belblidia, Quentin Heinrich, Tom Brendlé, and Maxime Vidal. 2020. FQuAD: French question answering dataset. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1193–1208, Online. Association for Computational Linguistics.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Yoann Dupont. 2017. Exploration de traits pour la reconnaissance d'entités nommées du français par apprentissage automatique (feature exploration for French named entity recognition with machine learning). In *Actes des 24ème Conférence sur le Traitement Automatique des Langues Naturelles. 19es REncontres jeunes Chercheurs en Informatique pour le TAL (RECITAL 2017)*, pages 42–55, Orléans, France. ATALA.

Yoann Dupont and Isabelle Tellier. 2014. A named entity recognizer for French (un reconnaisseur d'entités nommées du français) [in French]. In *Proceedings of TALN 2014 (Volume 3: System Demonstrations)*, pages 40–41, Marseille, France. Association pour le Traitement Automatique des Langues.

Ahmed El-Kishky, Vishrav Chaudhary, Francisco Guzmán, and Philipp Koehn. 2020. CCAligned: A massive collection of cross-lingual web-document pairs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5960–5969, Online. Association for Computational Linguistics.

Jacob Emerick. 2018. List of dirty naughty obscene and otherwise bad words.

Miquel Esplà, Mikel Forcada, Gema Ramírez-Sánchez, and Hieu Hoang. 2019. ParaCrawl: Web-scale parallel corpora for the languages of the EU. In *Proceedings of Machine Translation Summit XVII: Translator, Project and User Tracks*, pages 118–119, Dublin, Ireland. European Association for Machine Translation.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2020. Beyond English-Centric Multilingual Machine Translation. *arXiv e-prints*, page arXiv:2010.11125.

Sylvain Galliano, Edouard Geoffrois, Djamel Mostefa, Khalid Choukri, Jean-François Bonastre, and Guillaume Gravier. 2005. The ESTER phase II evaluation campaign

for the rich transcription of French broadcast news. In *Proc. Interspeech 2005*, pages 1149–1152.

Sylvain Galliano, Guillaume Gravier, and Laura Chaubard. 2009. The ester 2 evaluation campaign for the rich transcription of French radio broadcasts. In *Proc. Interspeech 2009*, pages 2583–2586.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *arXiv e-prints*, page arXiv:2101.00027.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, III Daumé, Hal, and Kate Crawford. 2018. Datasheets for Datasets. *arXiv e-prints*, page arXiv:1803.09010.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Cyril Grouin, Sophie Rosset, Pierre Zweigenbaum, Karën Fort, Olivier Galibert, and Ludovic Quintard. 2011. Proposal for an extension of traditional named entities: From guidelines to evaluation, an overview. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 92–100, Portland, Oregon, USA. Association for Computational Linguistics.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Sarah Holland, Ahmed Hosny, Sarah Newman, Joshua Joseph, and Kasia Chmielinski. 2018. The Dataset Nutrition Label: A Framework To Drive Higher Data Quality Standards. *arXiv e-prints*, page arXiv:1805.03677.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4411–4421. PMLR.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv e-prints*, page arXiv:1508.01991.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. FastText.zip: Compressing text classification models. *arXiv e-prints*, page arXiv:1612.03651.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. 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. Association for Computational Linguistics.

Marcin Junczys-Dowmunt. 2018. Dual conditional cross-entropy filtering of noisy parallel corpora. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 888–895, Belgium, Brussels. Association for Computational Linguistics.

Marcin Junczys-Dowmunt. 2019. Microsoft translator at WMT 2019: Towards large-scale document-level neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 225–233, Florence, Italy. Association for Computational Linguistics.

David Kamholz, Jonathan Pool, and Susan Colowick. 2014. PanLex: Building a resource for panlingual lexical translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3145–3150, Reykjavik, Iceland. European Language Resources Association (ELRA).

Vincentius Kevin, Birte Högden, Claudia Schwenger, Ali Şahan, Neelu Madan, Piush Aggarwal, Anusha Bangaru, Farid Muradov, and Ahmet Aker. 2018. Information nutrition labels: A plugin for online news evaluation. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 28–33, Brussels, Belgium. Association for Computational Linguistics.

Huda Khayrallah and Philipp Koehn. 2018. On the impact of various types of noise on neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 74–83, Melbourne, Australia. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X: Papers*, pages 79–86, Phuket, Thailand.

Philipp Koehn, Vishrav Chaudhary, Ahmed El-Kishky, Naman Goyal, Peng-Jen Chen, and Francisco Guzmán. 2020. Findings of the WMT 2020 shared task on parallel corpus filtering and alignment. In *Proceedings of the Fifth Conference on Machine Translation*, pages 726–742, Online. Association for Computational Linguistics.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing Universal Dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2779–2795, Hong Kong, China. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Anne Lacheret, Sylvain Kahane, Julie Beliao, Anne Dister, Kim Gerdes, Jean-Philippe Goldman, Nicolas Obin, Paola Pietrandrea, and Atanas Tchobanov. 2014. Rhapsodie: a prosodic-syntactic treebank for spoken French. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 295–301, Reykjavik, Iceland. European Language Resources Association (ELRA).

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 504–513, Uppsala, Sweden. Association for Computational Linguistics.

Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoit Crabbé, Laurent Besacier, and Didier Schwab. 2020. FlauBERT: Unsupervised language model pre-training for French. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2479–2490, Marseille, France. European Language Resources Association.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv e-prints*, page arXiv:1907.11692.

Elaine Marsh and Dennis Perzanowski. 1998. MUC-7 evaluation of IE technology: Overview of results. In *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal Dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria. Association for Computational Linguistics.

Rada Mihalcea. 2007. Using Wikipedia for automatic word sense disambiguation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 196–203, Rochester, New York. Association for Computational Linguistics.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In

Advances in Neural Information Processing Systems, volume 26. Curran Associates, Inc.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224, Uppsala, Sweden. Association for Computational Linguistics.

Wilhelmina Nekoto, Vukosi Marivate, Tshinondiwa Matsila, Timi Fasubaa, Taiwo Fagbohungbe, Solomon Oluwole Akinola, Shamsuddeen Muhammad, Salomon Kabongo Kabenamualu, Salomey Osei, Freshia Sackey, Rubungo Andre Niyongabo, Ricky Macharm, Perez Ogayo, Orevaoghene Ahia, Musie Meressa Berhe, Mofetoluwa Adeyemi, Masabata Mokgesi-Selinga, Lawrence Okegbemi, Laura Martinus, Kolawole Tajudeen, Kevin Degila, Kelechi Ogueji, Kathleen Siminyu, Julia Kreutzer, Jason Webster, Jamiil Toure Ali, Jade Abbott, Iroro Orife, Ignatius Ezeani, Idris Abdulkadir Dangana, Herman Kamper, Hady Elsahar, Goodness Duru, Ghollah Kioko, Murhabazi Espoir, Elan van Biljon, Daniel Whitenack, Christopher Onyefuluchi, Chris Chinenye Emezue, Bonaventure F. P. Dossou, Blessing Sibanda, Blessing Bassey, Ayodele Olabiyi, Arshath Ramkilowan, Alp Öktem, Adewale Akinfaderin, and Abdallah Bashir. 2020. Participatory research for low-resourced machine translation: A case study in African languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2144–2160, Online. Association for Computational Linguistics.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Sandra Bellato, Kepa Bengoetxea, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Rogier Blokland, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva,

Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phương Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roșca, Olga Rudina, Shoval Sadde, Shadi Saleh, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Yuta Takahashi, Takaaki Tanaka, Isabelle Tellier, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Veronika Vincze, Lars Wallin, Jonathan North Washington, Seyi Williams, Mats Wirén, Tsegay Woldemariam, Tak-sum Wong, Chunxiao Yan, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, Manying Zhang, and Hanzhi Zhu. 2018. Universal dependencies 2.2. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Damien Nouvel, Jean-Yves Antoine, and Nathalie Friburger. 2014. Pattern mining for named entity recognition. In *Human Language Technology Challenges for Computer Science and Linguistics*, pages 226–237, Cham. Springer International Publishing.

Debora Nozza, Federico Bianchi, and Dirk Hovy. 2020. What the [MASK]? Making Sense of Language-Specific BERT Models. *arXiv e-prints*, page arXiv:2003.02912.

Pedro Javier Ortiz Suárez, Laurent Romary, and Benoît Sagot. 2020. A monolingual approach to contextualized word embeddings for mid-resource languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1703–1714, Online. Association for Computational Linguistics.

Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. 2019. Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures. In *Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7) 2019. Cardiff, 22nd July 2019*, pages 9 – 16, Mannheim. Leibniz-Institut für Deutsche Sprache.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition, linguistic data consortium. *Technical report, Technical Report. Linguistic Data Consortium*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. 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. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. 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. Association for Computational Linguistics.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2089–2096, Istanbul, Turkey. European Language Resources Association (ELRA).

Addison Phillips and Mark Davis. 2005. Tags for Identifying Languages. Internet-Draft draft-phillips-langtags-10, Internet Engineering Task Force. Work in Progress.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational*

Linguistics, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea. Association for Computational Linguistics.

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 529–535, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI Blog*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Christian Raymond and Julien Fayolle. 2010. Reconnaissance robuste d'entités nommées sur de la parole transcrite automatiquement. In *Actes de la 17e conférence sur le Traitement Automatique des Langues Naturelles. Articles longs*, pages 191–200, Montréal, Canada. ATALA.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Sophie Rosset, Cyril Grouin, and Pierre Zweigenbaum. 2011. Entités nommées structurées : guide d'annotation quaero. Technical report, LIMSI-CNRS. Autres.

Sophie Rosset, Gabriel Illouz, and Aurélien Max. 2005. Interaction et recherche d'information : le projet Ritel. *Traitement Automatique des Langues*, 46(3):155–179.

Benoît Sagot, Marion Richard, and Rosa Stern. 2012. Annotation référentielle du corpus arboré de Paris 7 en entités nommées (referential named entity annotation of the Paris 7 French TreeBank) [in French]. In *Proceedings of the Joint Conference JEP-TALN-RECITAL 2012, volume 2: TALN*, pages 535–542, Grenoble, France. ATALA/AFCP.

Benoît Sagot and Rosa Stern. 2012. Aleda, a free large-scale entity database for French. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 1273–1276, Istanbul, Turkey. European Language Resources Association (ELRA).

Manuela Sanguinetti and Cristina Bosco. 2015. *PartTUT: The Turin University Parallel Treebank*, pages 51–69. Springer International Publishing, Cham.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Holger Schwenk, Vishrav Chaudhary, Shuo Sun, Hongyu Gong, and Francisco Guzmán. 2021. WikiMatrix: Mining 135M parallel sentences in 1620 language pairs from Wikipedia. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1351–1361, Online. Association for Computational Linguistics.

Amit Seker, Amir More, and Reut Tsarfaty. 2018. Universal morpho-syntactic parsing and the contribution of lexica: Analyzing the ONLP lab submission to the CoNLL 2018 shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 208–215, Brussels, Belgium. Association for Computational Linguistics.

Satoshi Sekine and Chikashi Nobata. 2004. Definition, dictionaries and tagger for extended named entity hierarchy. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Linda J. Skitka, Kathleen L. Mosier, and Mark Burdick. 1999. Does automation bias decision-making? *International Journal of Human-Computer Studies*, 51(5):991–1006.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018. 82 treebanks, 34 models: Universal Dependency parsing with multi-treebank models. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 113–123, Brussels, Belgium. Association for Computational Linguistics.

Rosa Stern. 2013. *Identification automatique d'entités pour l'enrichissement de contenus textuels*. Theses, Université Paris-Diderot - Paris VII.

Rosa Stern and Benoît Sagot. 2010. Resources for Named Entity Recognition and Resolution in News Wires. In *Entity 2010 Workshop at LREC 2010*, Valletta, Malta.

Rosa Stern, Benoît Sagot, and Frédéric Béchet. 2012. A joint named entity recognition and entity linking system. In *Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data*, pages 52–60, Avignon, France. Association for Computational Linguistics.

Milan Straka. 2018. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium. Association for Computational Linguistics.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Milan Straka, Jana Straková, and Jan Hajič. 2019. Evaluating Contextualized Embeddings on 54 Languages in POS Tagging, Lemmatization and Dependency Parsing. *arXiv e-prints*, page arXiv:1908.07448.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy. Association for Computational Linguistics.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Chenkai Sun, Abolfazl Asudeh, H. V. Jagadish, Bill Howe, and Julia Stoyanovich. 2019. Mithralabel: Flexible dataset nutritional labels for responsible data science. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 2893–2896, New York, NY, USA. Association for Computing Machinery.

Mariona Taulé, M. Antònia Martí, and Marta Recasens. 2008. AnCora: Multilevel annotated corpora for Catalan and Spanish. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Wilson L. Taylor. 1953. "cloze procedure": A new tool for measuring readability. *Journalism Quarterly*, 30(4):415–433.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Trieu H. Trinh and Quoc V. Le. 2018. A Simple Method for Commonsense Reasoning. *arXiv e-prints*, page arXiv:1806.02847.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. 2019. Multilingual is not enough: BERT for Finnish. *arXiv e-prints*, page arXiv:1912.07076.

Wei Wang, Taro Watanabe, Macduff Hughes, Tetsuji Nakagawa, and Ciprian Chelba. 2018. Denoising neural machine translation training with trusted data and online data selection. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 133–143, Brussels, Belgium. Association for Computational Linguistics.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and

Alexander M. Rush. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv e-prints*, page arXiv:1910.03771.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127, Uppsala, Sweden. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 833–844, Hong Kong, China. Association for Computational Linguistics.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. 2021. Nyströmformer: A nyström-based algorithm for approximating self-attention. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14138–14148.

Hainan Xu and Philipp Koehn. 2017. Zipporah: a fast and scalable data cleaning system for noisy web-crawled parallel corpora. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2945–2950, Copenhagen, Denmark. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27.