Beyond Sesame street-based naming schemes: Camembert vs CharacterBERT, a study on the performance robustness of large monolingual language models and their character-based counterparts

CAMEMBERT MUST DIE! (JK, LOL)

Louis Martin, **Benjamin Muller,** **Pedro Javier Ortiz Suárez,**
Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah* and Benoît Sagot.
Ganesh Jawahar, Arij Riabi, Wissam Antoun

Djamé Seddah: Who am I?

- Tenured Associate Professor at Sorbonne University since 2006 (ex Paris IV)
- On teaching leave at the Inria Paris since 2018, focusing on morphologically-rich and low-resource languages with an emphasis on extremely-scarce resource scenarios (noisy user-generated content for dialects, various hate-speech and radicalization domains, etc.)
- Of course, trying to understand how to best take advantages of large neural language models in context of high language variability
- Participated in the first large scale replication of Bert (CamemBERT), first largest GPT model for French (PagnolXL) and first version of a character-based language model on very noisy UGC of dialectal Arabic (CharacterBert for UGC)
- Many data sets created (SPMRL, Sequoia, French Social Media Bank, French QuestionBank, Deep Syntax treebanks, etc.)

Djamé Seddah: main projects

- Anr Sequoia (2009-2012, co-PI): Statistical parsing of French. First free and libre syntactically annotated data set for French, morphological-clustering for parsing..
- Anr SoSweet (2015-2021, co-PI): Twitter Sociolinguistics Variability. Capturing contextual diachronic change, co-funded CamemBert, Bert-based normalisation, ..
- Anr Parsiti (2016-2022, PI) Tackling noisy user-generated content for Parsing and Machine Translation. Parallel French social media bank, cross-lingual transfer, etc.
- PHC Maimonide (2018-2020, co-PI with Bar Illan): community detection via shared semantic drift, word usage change detection,..
- H2020 Counter (2021-2024, co-PI) multilingual multi domain (almost) zero-shot online radicalisation detection. CharacterBert for UGC, hateful multimodal meme detection, data augmentation via target domain data generation,…

NLP: How does it work?

- **Using linguistics knowledge. One principle, two schools:**

  (i) Building grammars, extraction rules and associated software.

  ⇒ Old-school approach, costly. Precise but very application-dependant.

  (ii) Building annotated data set and build learning models that wil do the same as (1) (but better, certainly faster)

  ⇒Data-driven approach, we try to generalize the data. Flexible & domain sensitive

- **No (or much fewer) linguistics knowledge.**

  (i) Building « nothing » and counting on massive amount of data to detect regularities, bring out information

  ⇒**Non-supervised approaches** (=no prior explicit linguistics knowledge)

  (ii) Using (I) via language models and directly transfer knowledge to tasks => this is the current NLP revolution

The NLP first Revolution: the word embeddings

The problem : words as discrete symbols

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

chowder was nasty
pudding was terrible
cake was bad
hamburger was lousy

service was poor
atmosphere was shoddy
hammer was heavy

- To the computer, each word is just a symbol, so these are all the same.
- But to us, some are more similar than others.
- We'd like a word representation that can capture that.

(borrowed from Goldberg (2015))

The NLP first Revolution: the word embeddings

Path to the solution : distributional hypothesis

Dr. Baroni saw a hairy little wampinuck sleeping behind a tree

The Distributional Hypothesis - Harris 1954

Word in similar contexts tend to have similar meanings

Firth, 1957

« You should know a word by the company it keeps »

(borrowed from Goldberg (2015))

The NLP first Revolution: the word embeddings

Representing words as Vectors

Collecting contexts from co-occurences

he curtains open and the moon shining in on the barely
ars and the cold , close moon " . And neither of the w
rough the night with the moon shining so brightly , it
made in the light of the moon . It all boils down , wr
surely under a crescent moon , thrilled by ice-white
sun , the seasons of the moon ? Home , alone , Jay pla
m is dazzling snow , the moon has risen full and cold
un and the temple of the moon , driving out of the hug
in the dark and now the moon rises , full and amber a
bird on the shape of the moon over the trees in front
But I could n't see the moon or the stars , only the
rning , with a sliver of moon hanging among the stars
they love the sun , the moon and the stars . None of
the light of an enormous moon . The plash of flowing w
man 's first step on the moon ; various exhibits , aer
the inevitable piece of moon rock . Housing The Airsh
oud obscured part of the moon . The Allied guns behind

Word as vectors (embeddings)

Represent each word as a sparse, high dimensional vector of the words that co-occur with it.

moon = (the:324, shining:4, cold:1, brightly:2, stars:12, elephant:0, ...)

Words are similar if their vectors are similar.

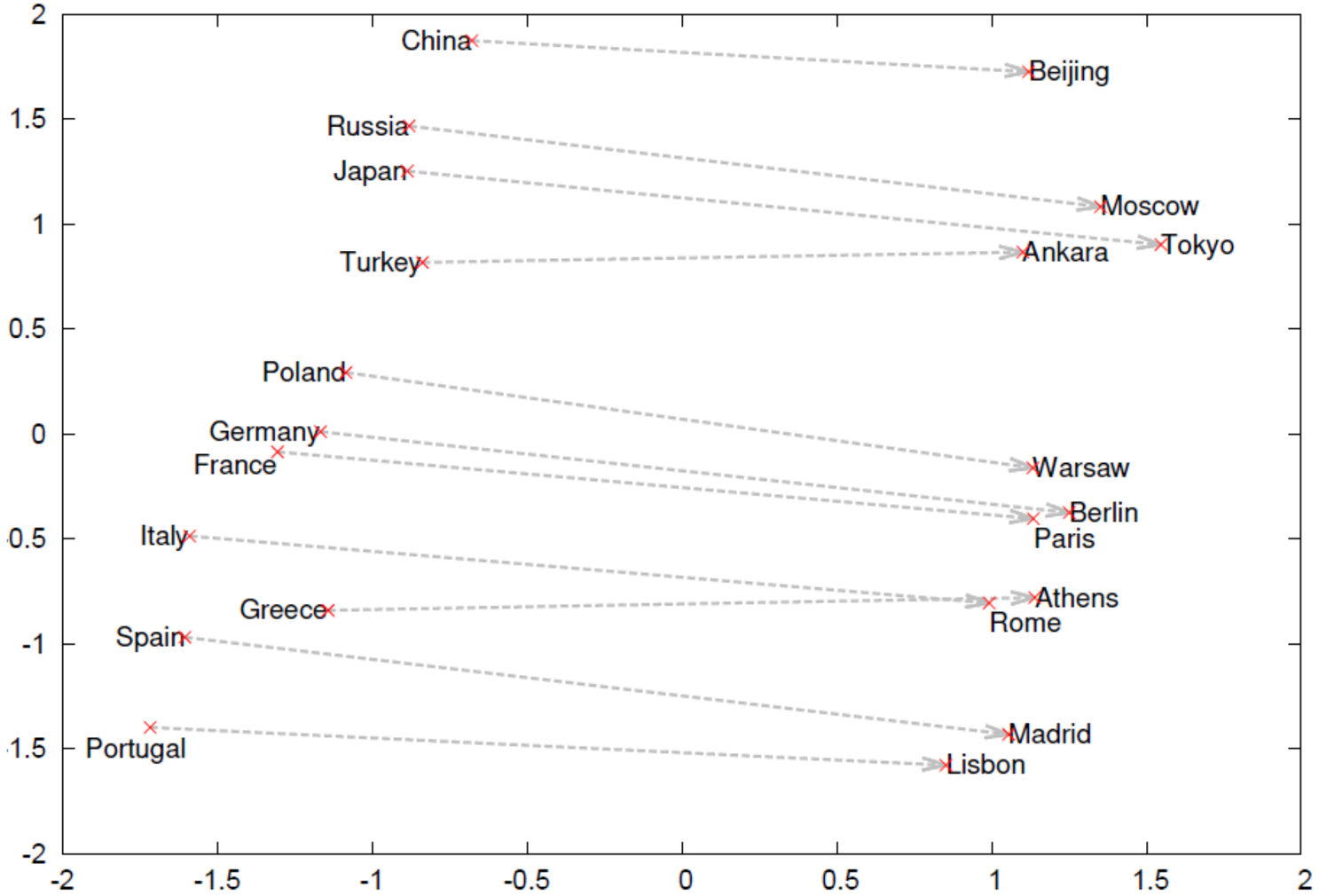
We measure similarity using geometric measures, for example *cosine distance*.

But more intuitively, **words are similar if they share many similar contexts.**

(borrowed from Goldberg (2015))

The NLP first Revolution: the word embeddings

Word2Vec (Mikolov et al., 2013) almost enabled magic



$b \quad a \quad a^* \quad b^*$

king − man + woman = queen

$b \quad a \quad a^* \quad b^*$

Tokyo − Japan + France = Paris

$b \quad a \quad a^* \quad b^*$

best − good + strong = strongest

vectors in $\mathbb{R}^n$

(borrowed from Goldberg (2015))

The NLP Second Revolution: Contextualization

- **Word embeddings are not that magic**
  - One huge drawback : **only one vector per word** (static vector)
  - **What about polysemy?** Think of the French word « réserver » in its *booking a flight* sense and its *cooking one*. **What changes?** Its **context of occurence.**
- **Solution : contextualized word embeddings**
  - Idea: relying on a **neural language model** to provide a different vector depending on the context (neighbors) of the word
  - many models appeared on a very short time span, less than a year (Elmo, Flair, GPT, **BERT**, GPT)…

***Language model** : a model that can predict the next word given a sequence of words

BERT: Bidirectional Encoder Representations from Transformers (Devlin et al., 2018, Naacl'2019)

- Contextual word embeddings model trained with
- **Masked word** prediction

my dog is hairy => my dog is [MASK] => Predict the word 'hairy'

- **Next sentence** prediction

the man went to a store [SEP] he bought a [MASK] milk => IsNext

- Transformer architecture (Vaswani et al., NeurIPS'17)
- Trained on BooksCorpus and Wikipedia (**English**)

BERT: Bidirectional Encoder Representations from Transformers (Devlin et al., 2018, Naacl'2019)

- **State-of-the-art results:**

GLUE score	80.5%	(**+7.7**)
MultiNLI accuracy	86.7%	(**+4.6**)
SQuAD v1.1 Q&A	93.2%	(**+1.5**)
SQuAD v2.0	83.1%	(**+5.1**)

So it turned out, BERT could actually "learn" something better than previous approaches about the English language.

The question of what can it learn instantly became a bubbling subfield, « the BERTology » see (Rogers et al, 2020)

BERTology Everyone wants to understand why it works so well and what it captures in terms of syntax

- Tenney et al. ICLR'19.
- Goldberg. arXiv'19.
- Hewitt et al. NAACL'19.
- Liu et al. NAACL'19.
- Jawahar et al. ACL'19
- Tenney et al. ACL'19.
- Wang et al. ACL'19.
- Lin et al. BlackboxNLP ACL'19.
- Clark et al. BlackboxNLP ACL'19
- Coenen et al. arXiv'19.
- Michel et al. arXiv'19.
- …

All of this, within basically six months !

So what does it capture?

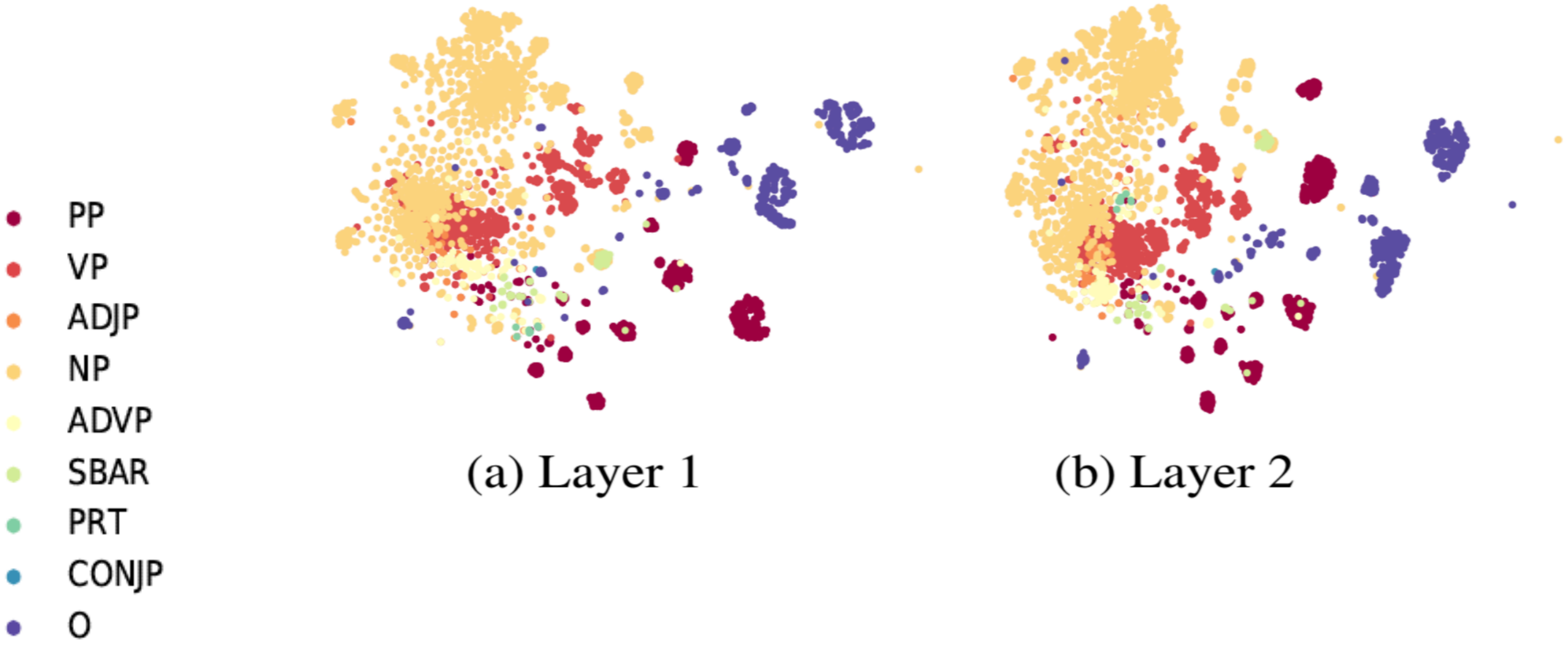
Long story short : SYNTAX

Using clustering as well as probing tasks (Conneau et al , 2019),
we showed that **lower layers capture phrasal information** while
upper layers captures relations between semantic heads
(Jawasahar et al, 2019)

- *Probe using the CoNLL 2000 Chunking dataset (Sang et al., 2000):*
*[NP **He**] [VP **reckons**] [NP **the current account deficit**] [VP **will narrow**]*
- *Compute **phrase representation** from representation of first and last token of the chunk.*
- *Plot t-SNE (Maaten and Hinton, JMLR'08) and perform clustering.*

So what does it capture? (2)

Phrasal Syntax – t-SNE Result



(a) Layer 1

(b) Layer 2

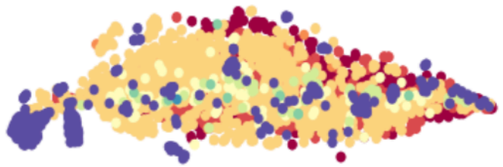
So what does it capture? (3)

Phrasal Syntax – t-SNE Result

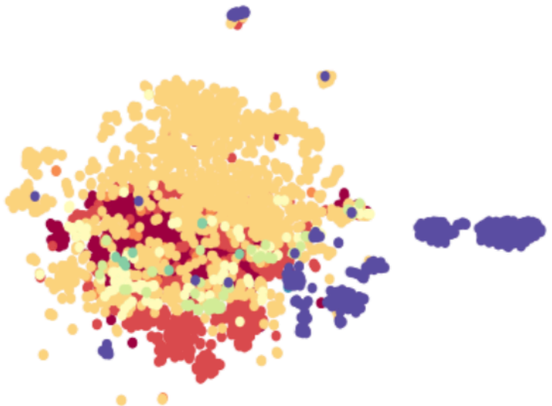
(a) Layer 1

(b) Layer 2

- PP
- VP
- ADJP
- NP
- ADVP
- SBAR
- PRT
- CONJP
- O



(c) Layer 11



(d) Layer 12

So what does it capture? (4)

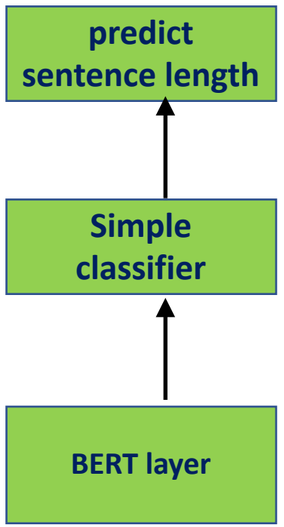
A bit more on that…

•Conneau et al., ACL'18 - Build diagnostic classifier to predict if a linguistic property is encoded in the given sentence representation.

•Features:

Surface – Sentence Length, Word Content
Syntactic – Bigram shift, Tree depth, Top constituent
Semantic – Tense, Subject Number, Object Number, Coordination Inversion and Semantic Odd Man Out.

If the prediction accuracy is good, then the model might be capturing the sentence length feature



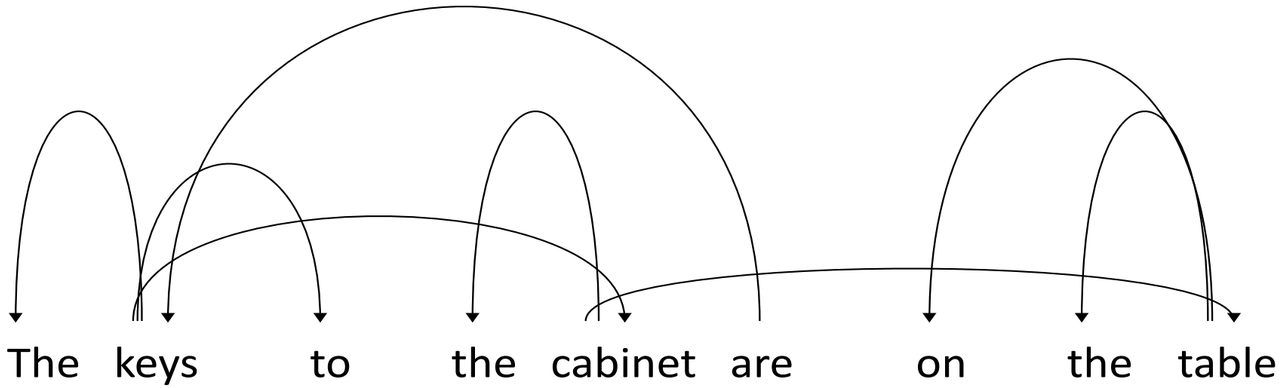
So what does it capture? (5)

Surface features on lower layers, more semantic on higher ones

Layer	SentLen (Surface)	WC (Surface)	TreeDepth (Syntactic)	TopConst (Syntactic)	BShift (Syntactic)	Tense (Semantic)	SubjNum (Semantic)	ObjNum (Semantic)	SOMO (Semantic)	CoordInv (Semantic)
1	93.9 (2.0)	24.9 (24.8)	35.9 (6.1)	63.6 (9.0)	50.3 (0.3)	82.2 (18.4)	77.6 (10.2)	76.7 (26.3)	49.9 (-0.1)	53.9 (3.9)
2	95.9 (3.4)	65.0 (64.8)	40.6 (11.3)	71.3 (16.1)	55.8 (5.8)	85.9 (23.5)	82.5 (15.3)	80.6 (17.1)	53.8 (4.4)	58.5 (8.5)
3	**96.2 (3.9)**	66.5 (66.0)	39.7 (10.4)	71.5 (18.5)	64.9 (14.9)	86.6 (23.8)	82.0 (14.6)	80.3 (16.6)	55.8 (5.9)	59.3 (9.3)
4	94.2 (2.3)	**69.8 (69.6)**	39.4 (10.8)	71.3 (18.3)	74.4 (24.5)	87.6 (25.2)	81.9 (15.0)	81.4 (19.1)	59.0 (8.5)	58.1 (8.1)
5	92.0 (0.5)	69.2 (69.0)	40.6 (11.8)	81.3 (30.8)	81.4 (31.4)	89.5 (26.7)	85.8 (19.4)	81.2 (18.6)	60.2 (10.3)	64.1 (14.1)
6	88.4 (-3.0)	63.5 (63.4)	**41.3 (13.0)**	83.3 (36.6)	82.9 (32.9)	89.8 (27.6)	**88.1 (21.9)**	82.0 (20.1)	60.7 (10.2)	71.1 (21.2)
7	83.7 (-7.7)	56.9 (56.7)	40.1 (12.0)	**84.1 (39.5)**	83.0 (32.9)	89.9 (27.5)	87.4 (22.2)	**82.2 (21.1)**	61.6 (11.7)	74.8 (24.9)
8	82.9 (-8.1)	51.1 (51.0)	39.2 (10.3)	84.0 (39.5)	83.9 (33.9)	89.9 (27.6)	87.5 (22.2)	81.2 (19.7)	62.1 (12.2)	76.4 (26.4)
9	80.1 (-11.1)	47.9 (47.8)	38.5 (10.8)	83.1 (39.8)	**87.0 (37.1)**	**90.0 (28.0)**	87.6 (22.9)	81.8 (20.5)	63.4 (13.4)	**78.7 (28.9)**
10	77.0 (-14.0)	43.4 (43.2)	38.1 (9.9)	81.7 (39.8)	86.7 (36.7)	89.7 (27.6)	87.1 (22.6)	80.5 (19.9)	63.3 (12.7)	78.4 (28.1)
11	73.9 (-17.0)	42.8 (42.7)	36.3 (7.9)	80.3 (39.1)	86.8 (36.8)	89.9 (27.8)	85.7 (21.9)	78.9 (18.6)	64.4 (14.5)	77.6 (27.9)
12	69.5 (-21.4)	49.1 (49.0)	34.7 (6.9)	76.5 (37.2)	86.4 (36.4)	89.5 (27.7)	84.0 (20.2)	78.7 (18.4)	**65.2 (15.3)**	74.9 (25.4)

So what does it capture? (6)

Cherry on the cake



Dependency parse tree induced from attention head #11 in layer #2 using gold root (‘are’) as starting node for the maximum spanning tree algorithm.

So everything solved ?

At that moment, all results were done on English

a very specific language (configurational language: word functions can be deduced from word order, poor morphology, etc..)

Then came Multilingual BERT (Delivn et al, 2019, Pires et al, 2019)

again it's not been a year already !

- Basically BERT trained on the concatenation of 104 languages (including French)
- Initial results showed **an almost magic ability to transfer** information across different languages, even different scripts
- Still, a picture began to emerged : **monolingual improvement were not as high as those experienced on English.**

Questions : Is it because of the **training data size** ? The **lack of text variability?** (mostly wikipedia-based)

Enters CamemBERT...

Some facts

- Prior to CamemBERT's release, no large monolingual transformer-based models comparable to BERT available (German and Chinese training data being much smaller in size)
- Since then, many came out (FlauBERT for French, BERTje for Dutch, FinBERT for Finnish)

Some technical facts about CamemBERT (base)

- 12 layers, 768 hidden dimensions, 12 attention heads, 110M parameters, 32Kwords (sentence piece)
- Trained on the Oscar corpora (138gb of raw texts from Common Crawl)
- Adapted from RoBERTA (Liu et al, 2019) that improves over Devlin et al's (2018) implementation (meaning only Mask Language Modeling as pretraining objective)

CamemBERT impact on downstream tasks: Setup

Two usages, 4 tasks

- (i) Fine-tuning and as (ii) Feature-based Embeddings
- POS tagging, Dependency Parsing, Named-Entity Recognition and Natural Language Inference

CamemBERT Embeddings

- Usually shown to perform slightly than lower fine-tuning, depends of the tasks
- Compute the average over of each subword representations in the last four layers and then average the resulting sub-word vectors.

Fine-Tuning

- No special tricks, fine-tuned for each individual task. Best model selected out of the 30 first epoch on the validation test.
- POS-tagging, Dep. parsing and NER are run within the HugginFace Transformer Library, NLI with Fairseq's implementation of Roberta

CamemBERT impact on downstream tasks: Setup

Baselines

- mBERT, (Pires et al, 2019) Multilingual BERT trained on 104 languages
- XLM(MLM-TLM) (multilingual pretrained language model with cross-lingual objectives)
- UDify (Kondratyuk, 2019), multitask and multilingual model that basically fine-tuned mBERT on the 124 UD treebanks *(brutal but genius if you ask me)*
- UDpipe Future+mBERT+Flair (Straka et al, 2019), a bi-LSTM-based parser that uses both mBERT and Flair as features-based contextualized embeddings

Data Set

- POS tagging/parsing: 4 French treebanks (UD-GSD, Sequoia, Spoken and ParTUT)
- NER : NER annotated version of the FTB (Sagot et al, 2012)
- NLI : XNLI (Conneau et al, 2018) French test and dev: manual, train: MT

Parsing and POS tagging

Sota Results on almost all data sets, except Spoken.

Possible reasons: speech data set with no capitalisation and no punctuations.

CamemBERT embeddings still improve though.

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, 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

Named-Entity Recognition and Natural Language Inference

Sota results on both tasks. Embeddings more impactful on NER

Note that we trained a Large version for a fair comparison with XLM-R$_{large}$

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

Table 2: **NER** scores on the FTB (best model selected on validation out of 4). Best scores in bold, second best underlined.

Model	Acc.	#Params
mBERT (Devlin et al., 2019)	76.9	175M
XLM$_{\text{MLM-TLM}}$ (Lample and Conneau, 2019)	<u>80.2</u>	250M
XLM-R$_{\text{BASE}}$ (Conneau et al., 2019)	80.1	270M
CamemBERT (fine-tuned)	**82.5**	110M
Supplement: LARGE models		
XLM-R$_{\text{LARGE}}$ (Conneau et al., 2019)	<u>85.2</u>	550M
CamemBERT$_{\text{LARGE}}$ (fine-tuned)	**85.7**	335M

Table 3: **NLI** accuracy on the French XNLI test set (best model selected on validation out of 10). Best scores in bold, second best underlined.

Striking questions: Impact of training data origin and size

Oscar vs CCNet: CCNet is a Common crawl filtered by a language model trained on wikipedia while Oscar just filtered CC based on a langage id classifier.

4GB vs 138GB :Varying the size and the origin of data, shows actually how little impact the pretraining set size actually has on model performances while uniformity (wikipedia) is detrimental in all cases.

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 4: 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).

Striking questions: Impact of Design Choices

Masking strategy, architecture, model size

Performance are comparable between CCNet and Oscar-based Camembert.

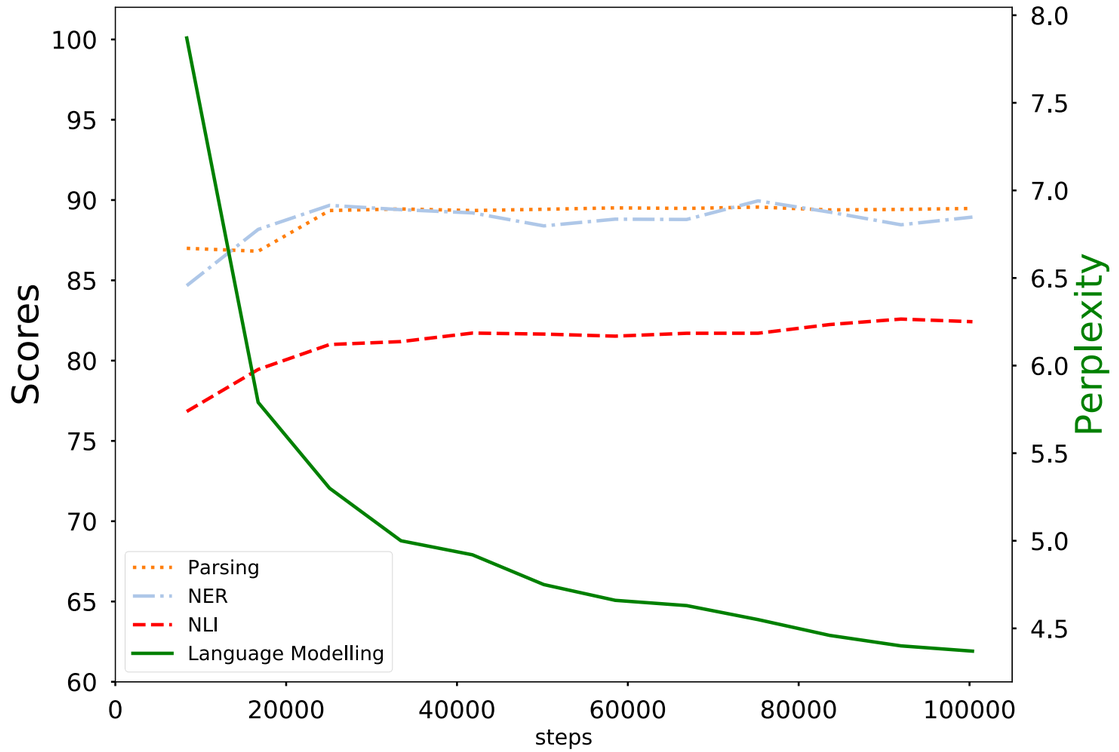
Positive Impact of large models of course.

Dataset	Masking	Arch.	#Param.	#Steps	UPOS	LAS	NER	XNLI
Masking Strategy								
CCNet	Subword	Base	110M	100K	97.78	89.80	**91.55**	81.04
CCNet	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**

Striking questions: Impact of Design Choices

Number of steps:

Varying the number of steps shows an early plateau for low level tasks (dep parsing and NER) while there's still an improvement for NLI and no performance ceiling in sight.



This suggests that low-level syntactic representation are captured early in the LM training process while it needs more steps to extract complex semantic information as needed for NLI.

Striking questions: How about Low Ressource and High Variability Languages Scenarios ?

Case Study: North-African Dialectal Arabic written in Latin Script

Linguistics Facts about Narabizi

- Arabic dialect spoken in North-Africa and among the diaspora.
- Mostly User-Generated Content
- Semitic language, rich inflexion system
- High degree of variability among speakers (spelling, transliteration, phonology)
- High degree of code switching with French (36%)

Gloss	Attested forms	Lang
why	wa3lach w3alh 3alach 3lache	NArabizi
all	ekl kal kolach koulli kol	NArabizi
many	beaucoup boucoup bcp	French

Data Set: the Narabizi Treebank (Seddah et al., 2020)

- 1500 sent. with morphology, dep. trees, French translation, etc.
- 99k web-crawled unlabeled sentences (46k of higher quality available)

ABSOLUTELY NOT ENOUGH DATA TO TRAIN A BERT MODEL !

Character-based language models to the rescue?

Character-BERT Model (El Boukkouri et al., 2020)

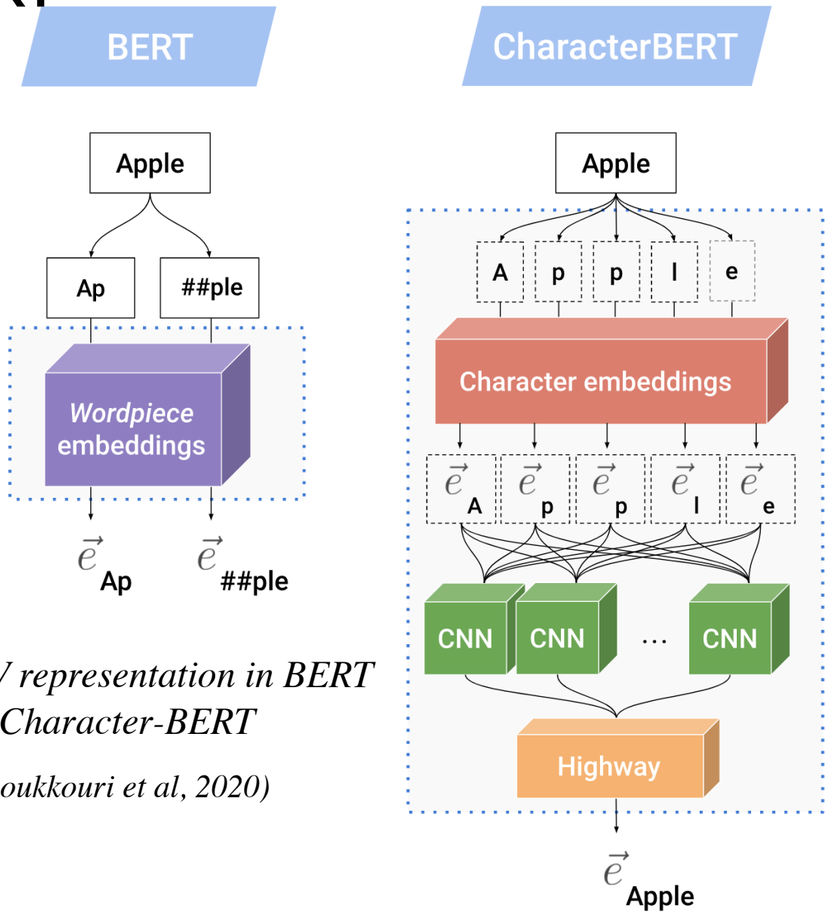
BERT is great but has some shortcomings:

- its vocabulary is fixed
- use of word-pieces to handle OOVs
- More generally domain-bias perceptible in the sub-words vocabulary
- Needs a lot of data (4gb is a lot in Narabizi)

Reference	Medical Vocabulary	General Vocabulary
paracetamol	[paracetamol	[para, ce, tam, ol]
choledocholithiasis	[choledoch, olithiasis]	[cho, led, och, oli, thi, asi, s]
borborygmi	[bor, bor, yg, mi]	[bo, rb, ory, gm, i]

Comparison of the tokenization of specific medical terms by vocabularies from different domains. (El Boukkouri et al, 2020)

Idea: Pushing Elmo's word representation model (Char-CNN) to BERT



OOV representation in BERT and Character-BERT

(El Boukkouri et al, 2020)

Character-based language models to the rescue? (2)

Character-BERT Model: an ideal model for noisy dialectal UGC?

- Robustsness to noise
- Needs less data to train
- Better performance than BERT in 3 out of 4 biomedical tasks (sequence labeling, NLI, Similarity detection)

Experiments along 2 axis

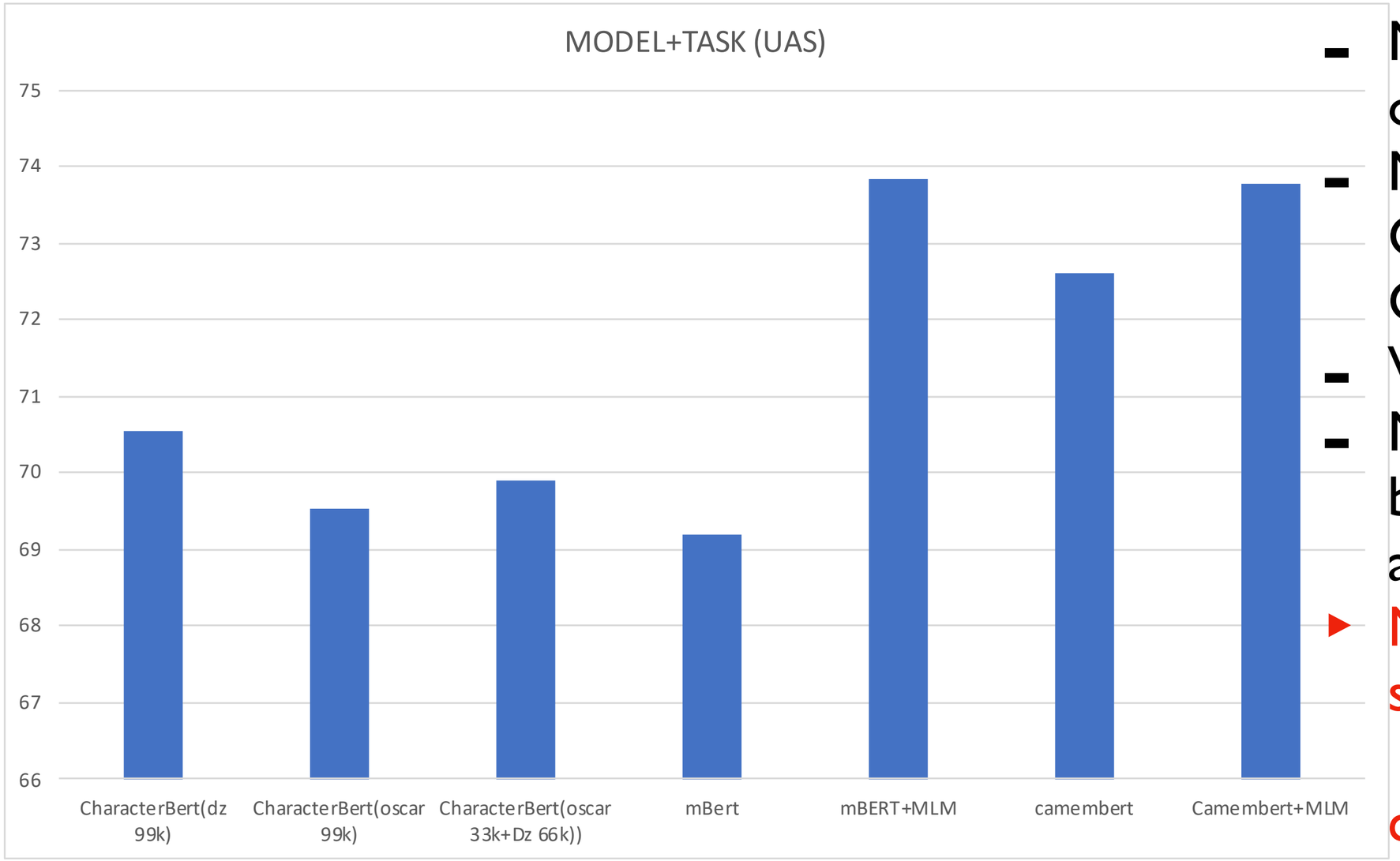
- **Pretraining data :**
  - Narabizi (99k),
  - Oscar (99k, **0.01%**)
  - Oscar+Narabizi (33k+66k)
- **Architecture (last layer ft)**
  - Model+Task:
    Pretraining + Fine-Tuning on Task
  - Model+MLM+Task:
    Pretraining+Fine-Tuning on MLM (Narabizi)+Fine-Tuning on Task

Tasks: POS Tagging, Dependency parsing
Parser: Neat version of the Biaffine parser by Grobold & Crabbé (2021)

Models: CamemBERT (Full) & mBERT
*** no pretraining from scratch ***

Character-based language models to the rescue? (3)

Character-BERT Model performance: okay-ish (0.01% of Oscar size)



- Model+MLM+Task overperform
- Model+Task: CamemBERT > CharBERT > mBERT
- Very small test set
- Many variations between configurations and tasks
- ▶ Need to know the full story with more test data and comparable training

Character-based language models to the rescue? (4)

Tackling Extreme UGC with the French Social Media Bank

- Very noisy UGC data set: Twitter, Facebook+forums (Seddah et al, 2012)
- Much larger test sets, splits: 2k/1k/1k
- Many in-domain treebanks (Sequoia: 2k training)
- Model+Task only. 2 scenarios: In-domain and Out-domain

CharacterBERT vs CamemBERT4GB

	UPOS	UAS	LAS	%Oscar
	FSMB fine-tuned (in-domain)			
CamemBERT	*95.48*	*87.47*	*82.66*	*100*
CamemBERT$_{4gb}$	95.13	85.73	80.72	2.38
Character-BERT	**95.19**	**86.26**	**81.26**	1
	Sequoia fine-tuned (out domain)			
CamemBERT	*90.10*	*82.68*	*75.85*	*100*
CamemBERT$_{4gb}$	**90.69**	82.29	**75.83**	2.38
Character-BERT	**90.68**	**82.39**	75.39	1

- CharacterBert: very strong performance in both scenarios but surprisingly so does CamemBERT4G !
- How does it work with even less pretraining data ?

Character-based language models to the rescue? (5)

Tackling Extreme UGC with the French Social Media Bank

- Very noisy UGC data set: Twitter, Facebook+forums (Seddah et al, 2012)
- Much larger test sets, splits: 2k/1k/1k
- Many in-domain treebanks (Sequoia: 2k training)
- Model+Task only. 2 scenarios: In-domain and Out-domain

CharacterBERT vs CamemBERT4GB

	UPOS	UAS	LAS	%Oscar
	FSMB fine-tuned (in-domain)			
No pre-training	*81.62*	*69.19*	*59.17*	*0*
CamemBERT	**95.25**	**86.91**	**81.87**	*100*
CamemBERT$_{4gb}$	95.2	86.39	81.21	2.38
Character-BERT	95.12	86.29	81.07	1
Character-BERT	93.78	83.13	77.49	0.1
Character-BERT	91.85	79.64	73.01	0.01
	Sequoia fine-tuned (out domain)			
No pre-training	*72.79*	*59.92*	*48.81*	*0*
CamemBERT	90.35	81.91	75.1	*100*
CamemBERT$_{4gb}$	**90.52**	**82.25**	**75.47**	2.38
Character-BERT	90.69	81.82	74.96	1
Character-BERT	88.33	77.28	69.89	0.1
Character-BERT	85.81	73.29	65.07	0.01

	# Tokens	# Sentences	Language
99k Narabizi	2.527k	99k	ar-dz
0.01% Oscar	3388k	99k	fr
0.1% Oscar	33.881k	993k	fr
1% Oscar	318.715k	9.342k	fr
10% Oscar	1.885.351k	55.261k	fr
100% Oscar	23.209.872k	558.092k	fr

- Lesser performance overall but not so bad. Much better than w/o any pretraining.

Character-based language models to the rescue? (6)

Tackling Extreme UGC with the French Social Media Bank

- Very noisy UGC data set: Twitter, Facebook+forums (Seddah et al, 2012)
- Much larger test sets, splits: 2k/1k/1k
- Many in-domain treebanks (Sequoia: 2k training)
- Model+Task only. 2 scenarios: In-domain and Out-domain

CharacterBERT vs CamemBERT4GB

	UPOS	UAS	LAS	%Oscar
	FSMB fine-tuned (in-domain)			
CamemBERT	*95.48*	*87.47*	*82.66*	*100*
CamemBERT_{4gb}	95.13	85.73	80.72	2.38
Character-BERT	**95.19**	**86.26**	**81.26**	1
	Sequoia fine-tuned (out domain)			
CamemBERT	*90.10*	*82.68*	*75.85*	*100*
CamemBERT_{4gb}	**90.69**	82.29	**75.83**	2.38
Character-BERT	**90.68**	**82.39**	75.39	1

- CharacterBert: very strong performance in both scenarios but surprisingly so does CamemBERT4G !
- **WHY ?**

The unreasonable effectiveness of subwords-based language models

Why are Bert-based models so robust?

- One part of the answer may come from an Omer Levy's group recent paper **Models In a Spelling Bee: Language Models Implicitly Learn the Character Composition of Tokens** (Itzhak and Levy, 2021)
- Using a smart Spelling Task probe they show that (i) BPE-based models are able to model character-level composition and (ii) more interestingly that pretraining the model on this spelling task embeddings doesn't improve the model MLM performance. Meaning that this character modeling ability is inherent to the model itself.

In conclusion, pretrained models are here to last

Especially since we showed that « less is beautiful » and training Bert-based models on corpora as small as 4GB with a relatively small number of steps is likely to boost NLP for under-resourced languages and domain-specific tasks.

Do we need Character-based models though? Big question, I'd say that if the target language/domain contains a lot of lexical variabilities, is under-resourced and is not included in the pre-training set, probably yes.
note:These conditions include most of the minority languages, specialized dialects and « niche » domains

But that's ongoing research under heavy scrutiny :)

CamemBERTa: adding Electra/DeBERTa to the mix

Goal: finding a more efficient model

DeBERTA v3 (He et al, 2021; current sota in Bert-class models)

Disantangled attention: The proposed disentangled attention mechanism differs from all existing approaches in that each input word is represented using two separate vectors that encode a word's content and position, respectively.

Key difference: instead of adding them, the two vectors are treated separately throughout the network.

Remplacement Token Detection objective: RTD corrupts the input by replacing some input tokens with incorrect—but somewhat plausible—fakes. The goal is therefore to predict whether the token has been replaced or not. Instead of only the [mask] tokens, all tokens are seen here.

CamemBERTa: Very cool results

CamemBERTA trained on 33% of CamemBERT, better results.

Same architecture as CamemBERT, same training data, same experiment settings

Model	CLS	PAWS-X	XNLI
CamemBERT$_{30\%}$	93.28	88.94	79.89
CAMEMBERTA	**94.92**	**91.67**	**82.00**
CamemBERT$_{CCNet}$	94.62	91.36	81.95

More performant than other monolingual models

More or equally performant as multilingual models

	XNLI	Steps	# tokens†	Size (in tokens)
mDeBERTa	84.4	500k	2T	2.5T
CAMEMBERTA	82.0	33k*	.139T	.319T
XLM-R**	81.4	1.5M	6T	2.5T
CamemBERT$_{CCNet}$	81.95	100k	.419T	.319T

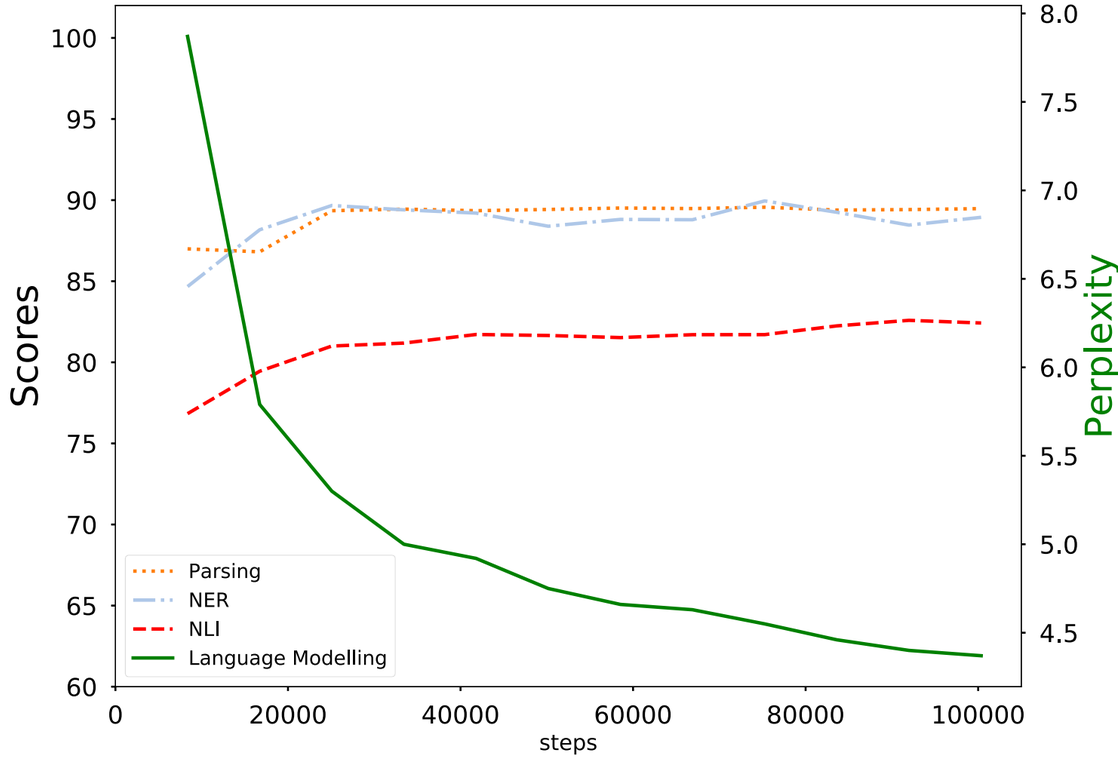
CamemBERTa: Very cool results

Things are lest clear on lower level tasks somehow:

> CamemBERT30%

~CamemBERTccnet

Model	GSD		Rhapsodie		Sequoia		FSMB		NER
	UPOS	LAS	UPOS	LAS	UPOS	LAS	UPOS	LAS	F1
CamemBERT$_{30\%}$	98.55	94.26	97.61	83.19	99.32	94.09	94.63	80.13	**91.04**
CamemBERTa	98.55	**94.38**	97.52	84.23	**99.44**	**94.85**	**94.80**	80.74	90.33
CamemBERT$_{CCNet}$	**98.57**	94.35	**97.62**	**84.29**	99.35	94.78	**94.80**	**81.34**	89.97



CamemBERT brought to you by the public service of research

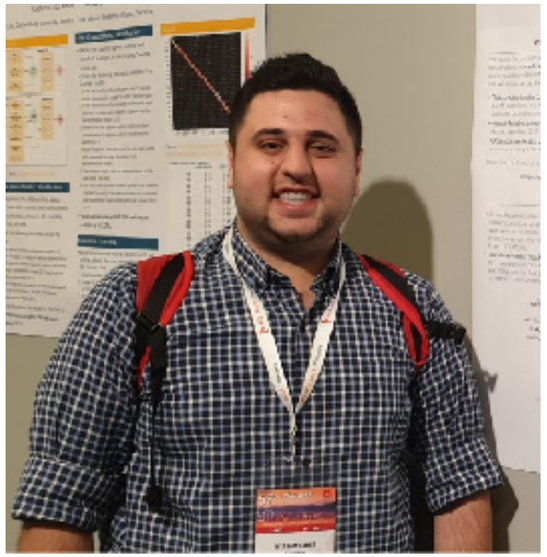
(with the support of FAIR)

https://camembert-model.fr

Thank you :)

- **Arij Riabi**
  CharacterBert for UGC : http://pauillac.inria.fr/~seddah/WNUT2021_CharacterBert4UGC_Riabi_Seddah.pdf

- Code and data set: https://gitlab.inria.fr/ariabi/character-bert-ugc



- **Wissam Antoun**
  CameBERTa : A French language model based on DeBERTa V3

- Code: https://gitlab.inria.fr/almanach/CamemBERTa