



CamemBERT

A Tasty French Language Model

[Facebook AI Research](#)

[Inria](#)

[ALMAAnaCH](#)



CamemBERT

CamemBERT is a state-of-the-art language model for French based on the [RoBERTa architecture](#) pretrained on the French subcorpus of the newly available multilingual corpus [OSCAR](#).




We evaluate CamemBERT in four different downstream tasks for French: part-of-speech (POS) tagging, dependency parsing, named entity recognition (NER) and natural language inference (NLI); improving the state of the art for most tasks over previous monolingual and multilingual approaches, which confirms the effectiveness of large pretrained language models for French.

CamemBERT was trained and evaluated by [Louis Martin](#), [Benjamin Muller](#), [Pedro Javier Ortiz Suárez](#), [Yoann Dupont](#), [Laurent Romary](#), [Éric Villemonte de la Clergerie](#), [Djamé Seddah](#) and [Benoît Sagot](#).

Publications

CamemBERT: a Tasty French Language Model

We release CamemBERT a Tasty French Language Model. CamemBERT is trained on 138GB of French text. It establishes a new state of the art in POS tagging, Dependency Parsing and NER, and achieves strong results in NLI. Bon appétit !

[Louis Martin](#) , [Benjamin Muller](#) , [Pedro Javier Ortiz Suárez](#) , [Yoann Dupont](#), [Laurent Romary](#), [Éric Villemonte de la Clergerie](#), [Djamé Seddah](#), [Benoît Sagot](#)

[PDF](#)[Cite](#)[Dataset](#)[ACL Anthology](#)[arXiv](#)[ACL 2020](#)[HAL](#)

Les modèles de langue contextuels Camembert pour le Français : impact de la taille et de l'hétérogénéité des données d'entraînement

We explore the impact of the training data size and heterogeneity on French language modeling.

[Louis Martin](#), [Benjamin Muller](#), [Pedro Javier Ortiz Suárez](#), [Yoann Dupont](#), [Laurent Romary](#), [Éric de la Clergerie](#), [Benoît Sagot](#), [Djamé Seddah](#)

- PDF
- Cite
- Dataset
- TALN 2020
- HAL
- Website



Download

Pre-trained models

CamemBERT is available in github.com/huggingface/transformers and <https://github.com/pytorch/fairseq/>

fairseq

Model	#params	Download	Arch.	Training data
camembert / camembert-base	110M	camembert-base.tar.gz	Base	OSCAR (138 GB of text)
camembert-large	335M	camembert-large.tar.gz	Large	CCNet (135 GB of text)
camembert-base-ccnet	110M	camembert-base-ccnet.tar.gz	Base	CCNet (135 GB of text)
camembert-base-wikipedia-4gb	110M	camembert-base-wikipedia-4gb.tar.gz	Base	Wikipedia (4 GB of text)
camembert-base-oscar-4gb	110M	camembert-base-oscar-4gb.tar.gz	Base	Subsample of OSCAR (4 GB of text)
camembert-base-ccnet-4gb	110M	camembert-base-ccnet-4gb.tar.gz	Base	Subsample of CCNet (4 GB of text)

Load CamemBERT from torch.hub (PyTorch >= 1.1):

```
import torch
camembert = torch.hub.load('pytorch/fairseq', 'camembert')
camembert.eval() # disable dropout (or leave in train mode to finetune)
```

Load CamemBERT (for PyTorch 1.0 or custom models):

Download camembert model

```
wget https://dl.fbaipublicfiles.com/fairseq/models/camembert-base.tar.gz
tar -xzf camembert-base.tar.gz
```

Load the model in fairseq

```
from fairseq.models.roberta import CamembertModel
camembert = CamembertModel.from_pretrained('./camembert-base/')
camembert.eval() # disable dropout (or leave in train mode to finetune)
```

Filling masks:

```
masked_line = 'Le camembert est <mask> :)'
camembert.fill_mask(masked_line, topk=3)
# [('Le camembert est délicieux :)', 0.4909118115901947, ' délicieux'),
# ('Le camembert est excellent :)', 0.10556942224502563, ' excellent'),
# ('Le camembert est succulent :)', 0.03453322499990463, ' succulent')]
```

Extract features from Camembert:

```
# Extract the last layer's features
line = "J'aime le camembert !"
tokens = camembert.encode(line)
last_layer_features = camembert.extract_features(tokens)
assert last_layer_features.size() == torch.Size([1, 10, 768])

# Extract all layer's features (layer 0 is the embedding layer)
all_layers = camembert.extract_features(tokens, return_all_hiddens=True)
assert len(all_layers) == 13
assert torch.all(all_layers[-1] == last_layer_features)
```

Transformers

```
import torch

from transformers.modeling_camembert import CamembertForMaskedLM
from transformers.tokenization_camembert import CamembertTokenizer

def fill_mask(masked_input, model, tokenizer, topk=5):
    # Adapted from https://github.com/pytorch/fairseq/blob/master/fairseq/models/roberta/hub_interface.py
    assert masked_input.count("<mask>") == 1
    input_ids = torch.tensor(tokenizer.encode(masked_input, add_special_tokens=True)).unsqueeze(0) # Batch size 1
    logits = model(input_ids)[0] # The last hidden-state is the first element of the output tuple
    masked_index = (input_ids.squeeze() == tokenizer.mask_token_id).nonzero().item()
    logits = logits[0, masked_index, :]
    prob = logits.softmax(dim=0)
    values, indices = prob.topk(k=topk, dim=0)
    topk_predicted_token_bpe = " ".join(
        [tokenizer.convert_ids_to_tokens(indices[i].item()) for i in range(len(indices))]
    )
    masked_token = tokenizer.mask_token
    topk_filled_outputs = []
    for index, predicted_token_bpe in enumerate(topk_predicted_token_bpe.split(" ")):
        predicted_token = predicted_token_bpe.replace("\u2581", " ")
        if "{0}".format(masked_token) in masked_input:
            topk_filled_outputs.append(
                (
                    masked_input.replace("{0}".format(masked_token), predicted_token),
                    values[index].item(),
                    predicted_token,
                )
            )
        else:
            topk_filled_outputs.append(
                (masked_input.replace(masked_token, predicted_token), values[index].item(), predicted_token,)
            )
    return topk_filled_outputs

tokenizer = CamembertTokenizer.from_pretrained("camembert-base")
model = CamembertForMaskedLM.from_pretrained("camembert-base")
model.eval()

masked_input = "Le camembert est <mask> :)"
print(fill_mask(masked_input, model, tokenizer, topk=3))
```

Citation

If you use our work, please cite:

```
@inproceedings{martin2020camembert,
  title={CamemBERT: a Tasty French Language Model},
  author={Martin, Louis and Muller, Benjamin and Ortiz Su{'a}rez, Pedro Javier and Dupont, Yoann and Romary, Laurent and de la Clergerie, {'\
booktitle={Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics},
  year={2020}
}
```

License

MIT License

Copyright (c) 2020 - Inria and Facebook, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.


Contact

Name


Email

Message

Send



[\(+33\) 01 80 49 40 00](tel:+33180494000)



[DM Me](#)

