FlauBERT: Unsupervised Language Model Pre-training for French

Hang Le[1] Loïc Vial[1] Jibril Frej[1] Vincent Segonne[2] Maximin Coavoux[1] Benjamin Lecouteux[1] Alexandre Allauzen[3] Benoît Crabbé[2] Laurent Besacier[1] Didier Schwab[1]

[1]Univ. Grenoble Alpes, CNRS, LIG [2]Université Paris Diderot
[3]E.S.P.C.I, CNRS LAMSADE, PSL Research University

Outline

1. Unsupervised pre-trained language models
2. FlauBERT
   - Models and architectures
   - Pre-training
   - Training data
   - How to use FlauBERT
3. FLUE benchmark
4. FlauBERT performance on FLUE

Unsupervised pre-trained language models

Principles

Unlabeled text data is used as a supervision signal
$\rightarrow$ also called *self-supervised* learning.

- **Word embeddings**: learn a feature vector for each word.
    - Feed-forward network (Bengio et al. 2003), convolutional network (Collobert and Weston 2008).
    - word2vec (Mikolov et al. 2010), GloVe (Pennington et al. 2014).
- **Contextual embeddings**: output representation is a function of entire input sequence.
    - Recurrent neural network: (Dai and Le 2015), ELMo (Peters et al. 2018), ULMFiT (Howard and Ruder 2018), MultiFiT (Eisenschlos et al. 2019).
    - Transformer-based: GPT (Radford et al. 2018), BERT (Devlin et al. 2019), XLNet (Yang et al. 2019), XLM (Lample and Conneau 2019), RoBERTa (Liu et al. 2019), ALBERT (Lan et al. 2019), T5 (Raffel et al. 2019).

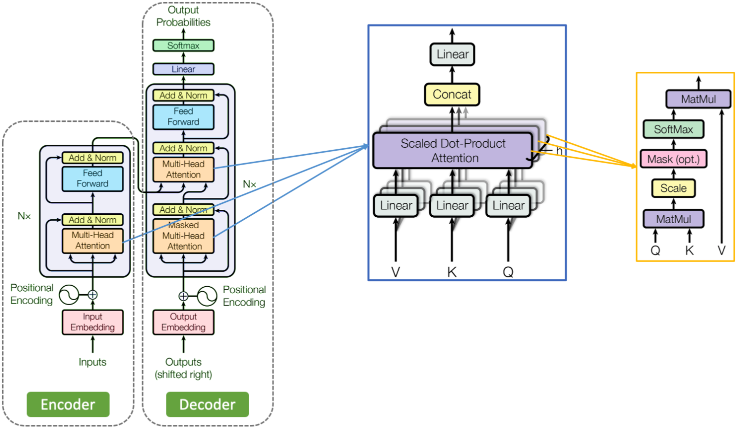
Unsupervised pre-trained language models

Advantages

- Leverage large amount of freely available *unlabeled* text.
- Facilitate *transfer learning* in NLP.
- Yield *state-of-the-art* results on a wide range of NLP tasks.
- Save time and computational *resources*.

FlauBERT: models and architectures

Based on the **Transformer** (Vaswani et al. 2017):



FlauBERT follows **BERT** (Devlin et al. 2019): only the encoder is used.

FlauBERT: models and architectures

Two different FlauBERT models:

- **FlauBERT$_{\textbf{BASE}}$**: $L = 12, H = 768, A = 12$.
- **FlauBERT$_{\textbf{LARGE}}$**: $L = 24, H = 1024, A = 16$.

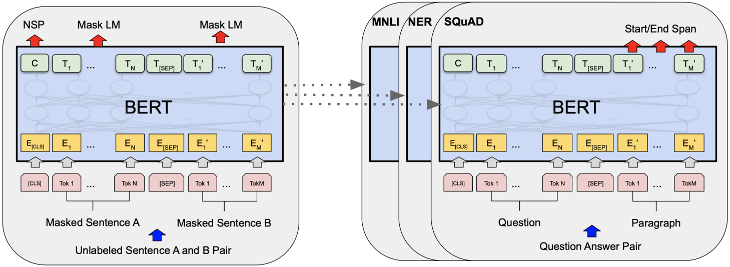
where:

- H: hidden size,
- L: number of Transformer blocks,
- A: number of self-attention heads.

$\rightarrow$ Number of parameters: FlauBERT$_{\text{BASE}}$ has $138M$, FlauBERT$_{\text{LARGE}}$ has $373M$.

FlauBERT: pre-training

Following the **BERT** (Devlin et al. 2019) paradigm.



Pre-training Fine-Tuning

- **Masked Language Model**: learn to predict randomly masked input tokens.
- **Next Sentence Prediction**: learn to predict if B is the next sentence to A, given an input pair (A, B). (Not used in FlauBERT.)

FlauBERT: pre-training

Training details

	BERT	RoBERTa	FlauBERT
Language	English	English	French
Training objectives	NSP and MLM	MLM	MLM
Tokenizer	WordPiece 30K	BPE 50K	BPE 50K

- Training configurations: following **RoBERTa** (Liu et al. 2019).
- Implementation: based on **XLM** library (Conneau et al. 2019).
- FlauBERT was trained using the new CNRS **Jean Zay** supercomputer:
    - $\text{FlauBERT}_{\text{BASE}}$: trained on 8 nodes (32 GPUs).
    - $\text{FlauBERT}_{\text{LARGE}}$: trained on 32 nodes (128 GPUs).

FlauBERT: pre-training

Training data

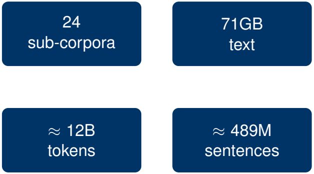


Figure 1: Training data.

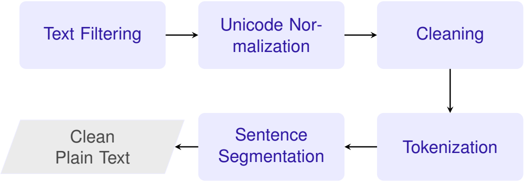


Figure 2: Text preprocessing pipeline.

- CommonCrawl: 43.4 GB
- NewsCrawl: 9.2 GB
- Wikipedia: 4.2 GB
- Wikisource: 2.4 GB
- EU Bookshop: 2.3 GB
- MultiUN: 2.3 GB
- GIGA: 2.0 GB
- PCT: 1.2 GB
- Project Gutenberg: 1.1 GB
- OpenSubtitles: 1.1 GB
- And others: 1.8 GB.

How to use FlauBERT

Feature extraction

FlauBERT can be used to extract embedding vectors for input tokens or sentences.

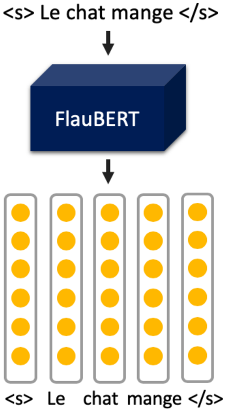
```python
import torch
from transformers import FlaubertModel,
     FlaubertTokenizer

# Load pretrained model and tokenizer
modelname = 'flaubert-base-cased'
flaubert, log = FlaubertModel.
     from_pretrained(modelname,
     output_loading_info=True)
flaubert_tokenizer = FlaubertTokenizer.
     from_pretrained(modelname, do_lowercase
     =False)

# Sample sentence
sentence = "Le chat mange une pomme."
token_ids = torch.tensor([flaubert_tokenizer
     .encode(sentence)])

# Get embeddings for all tokens
last_layer = flaubert(token_ids)[0]

# Get sentence embeddings
cls_embedding = last_layer[:, 0, :]
```



How to use FlauBERT

Fine-tuning on downstream tasks

Append the task-specific layers to FlauBERT $\rightarrow$ Train on target tasks.
Pre-trained weights can be freezed or not freezed.

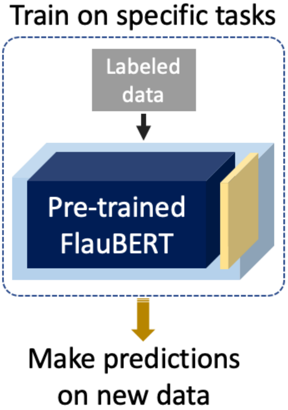
```
import numpy as np
import torch
from transformers import FlaubertTokenizer,
     FlaubertForSequenceClassification

# Load pretrained model and tokenizer
tokenizer = FlaubertTokenizer.from_pretrained('
     flaubert-base-cased')
model = FlaubertForSequenceClassification.
     from_pretrained('flaubert-base-cased')

# 1. Train model on target labeled data
train(model)
# For more details, please refer to
# https://github.com/huggingface/transformers/tree
     /master/examples/text-classification

# 2. Make predictions
# Sample data
input_ids = torch.tensor(tokenizer.encode("Le chat
      mange une pomme.", \
            add_special_tokens=True)).unsqueeze(0)
labels = torch.tensor([1]).unsqueeze(0)

# Get prediction label
outputs = model(input_ids, labels=labels)
loss, logits = outputs[:2]
preds = logits.detach().cpu().numpy()
preds = np.argmax(preds, axis=-1)
```



FLUE benchmark

- Text Classification
- Paraphrasing
- Natural Language Inference (NLI)
- Parsing and Part-of-Speech Tagging
- Word Sense Disambiguation Tasks

Dataset		Domain	Train	Dev	Test
CLS-FR	Books	Product reviews	2 000	-	2 000
	DVD		1 999	-	2 000
	Music		1 998	-	2 000
PAWS-X-FR		General domain	49 401	1 992	1 985
XNLI-FR		Diverse genres	392 702	2 490	5 010
French Treebank		Daily newspaper	14 759	1 235	2 541
Verb WSD		Diverse genres	55 206	-	3 199
Noun Sense Disambiguation		Diverse genres	818 262	-	1 445

Table 1: Descriptions of the datasets included in our FLUE benchmark.

FlauBERT performance on FLUE

Task	Classification			Paraphrasing	NLI	Constituency		Dependency		Disambiguation	
Section	Books	DVD	Music							Nouns	Verbs
Measure	Acc.	Acc.	Acc.	Acc.	Acc.	F_1	POS	UAS	LAS	F_1	F_1
State-of-the-art	91.25[c]	89.55[c]	93.40[c]	66.20[d]	80.1/**85.2**[e]	87.4[a]		89.19[b]	85.86[b]	-	43.0[h]
Without pre-training	-	-	-			83.9	97.5	88.92	85.11	50.03	-
FastText	-	-	-			83.6	97.7	86.32	82.04	49.41	34.90
mBERT	86.15[c]	86.9[c]	86.65[c]	89.30[d]	76.9[f]	87.5	98.1	89.50	85.86	56.47	49.83
CamemBERT	92.30	93.00	94.85	**90.14**	81.2	88.4	**98.2**	91.37	88.13	56.06	50.02
FlauBERT$_{\text{BASE}}$	93.10	92.45	94.10	89.49	80.6	**89.1**	98.1	91.56	88.35	54.74	43.92
FlauBERT$_{\text{LARGE}}$	**95.00**	**94.10**	**95.85**	89.34	83.4	88.6	**98.2**	**91.61**	**88.47**	**57.85**	**50.48**

Table 2: Final results on FLUE. [a](Kitaev et al. 2019). [b](Constant et al. 2013). [c](Eisenschlos et al. 2019). [d](Chen et al. 2017). [e](Conneau et al. 2019). [f](Martin et al. 2019). [h](Segonne et al. 2019).

Thank you for your attention!

Code and data are available:

- FlauBERT: `https://github.com/getalp/Flaubert`.
- FLUE: `https://github.com/getalp/Flaubert/tree/master/flue`.

Acknowledgements:

- GENCI (Grand Equipment National de Calcul Intensif) for computing resources.
- Our code is based on Facebook's XLM and HuggingFace's Transformers libraries.