Tugas Course NLP

Chapter 4

Nama : muhammad makhlufi makbullah

Kelas : TK 45 01

NIM : 1103210171

Chapter 4 sharing models and tokenizers

Hugging Face Hub adalah platform pusat yang memungkinkan siapa saja untuk menemukan, menggunakan, dan berkontribusi pada model dan dataset terbaru. Platform ini memiliki lebih dari 10.000 model yang tersedia secara publik. Pada bab ini, fokus utamanya adalah pada model, sedangkan dataset akan dibahas di Bab 5.

Model-model yang ada di Hub tidak terbatas pada 🤗 Transformers atau bahkan NLP saja. Ada juga model untuk Flair dan AllenNLP (untuk NLP), Asteroid dan pyannote (untuk suara), serta timm (untuk visi), dan lainnya.

Setiap model di-hosting sebagai repositori Git, yang memungkinkan versi dan reproduksibilitas. Berbagi model di Hub berarti membuka akses kepada komunitas dan memungkinkan orang lain untuk menggunakannya tanpa perlu melatih model mereka sendiri, serta mempermudah proses berbagi dan penggunaan.

Selain itu, berbagi model di Hub secara otomatis menyediakan API Inferensi yang di-hosting untuk model tersebut. Komunitas dapat langsung mengujinya di halaman model dengan input kustom dan widget yang sesuai.

using pretrained models

1. Menginstal Paket yang Dibutuhkan

!pip install datasets evaluate transformers[sentencepiece]

!pip install adalah perintah untuk menginstal paket-paket Python.

- datasets: Pustaka ini digunakan untuk mengakses dan memanipulasi dataset.
- evaluate: Digunakan untuk evaluasi model.
- transformers[sentencepiece]: Pustaka Hugging Face Transformers dengan dukungan tambahan untuk SentencePiece, yang digunakan untuk tokenisasi teks. SentencePiece sering digunakan untuk model-model berbasis byte pair encoding (BPE) atau model-model lain seperti BERT, T5, dan Camembert.

2. Mengimpor dan Menggunakan pipeline untuk Masked Language Model

from transformers import pipeline

- from transformers import pipeline: Mengimpor fungsi pipeline dari pustaka Hugging Face Transformers. Fungsi pipeline memudahkan kita untuk melakukan tugas-tugas tertentu (seperti klasifikasi teks, pemrosesan teks, dan pengisian teks) tanpa menulis kode kompleks.

camembert_fill_mask = pipeline("fill-mask", model="camembert-base")

- pipeline("fill-mask", model="camembert-base"): Membuat pipeline untuk tugas masked language modeling (MLM) menggunakan model Camembert dengan arsitektur berbasis RoBERTa, yang dilatih untuk memahami konteks dalam kalimat.
- "fill-mask" adalah tugas di mana model mengisi bagian teks yang hilang (mask) dengan prediksi yang sesuai.
- model="camembert-base": Menentukan model yang digunakan adalah Camembert Base (pretrained model Camembert dari Hugging Face).

results = camembert_fill_mask("Le camembert est <mask> :)")

- camembert_fill_mask("Le camembert est <mask> :)"): Memberikan input ke pipeline dengan kalimat yang memiliki <mask> di tengahnya, yang berarti bagian teks tersebut akan diisi oleh model.
- Kalimat ini adalah contoh teks berbahasa Prancis yang meminta model untuk menebak kata yang hilang (diwakili dengan <mask>).

3. Memuat Tokenizer dan Model Secara Manual

from transformers import CamembertTokenizer, CamembertForMaskedLM

- from transformers import CamembertTokenizer, CamembertForMaskedLM: Mengimpor CamembertTokenizer dan CamembertForMaskedLM.
- CamembertTokenizer digunakan untuk tokenisasi teks (mengubah teks menjadi token yang dimengerti oleh model).
- CamembertForMaskedLM adalah model Camembert yang dilatih untuk tugas Masked Language Modeling.

tokenizer = CamembertTokenizer.from_pretrained("camembert-base")

- CamembertTokenizer.from_pretrained("camembert-base"): Memuat tokenizer Camembert dari model camembert-base yang sudah dilatih sebelumnya. Tokenizer ini akan mengubah teks menjadi token yang bisa diproses oleh model.

model = CamembertForMaskedLM.from_pretrained("camembert-base")

- CamembertForMaskedLM.from_pretrained("camembert-base"): Memuat model Camembert untuk tugas Masked Language Modeling dari model yang sudah dilatih sebelumnya (camembert-base).

4. Menggunakan AutoTokenizer dan AutoModelForMaskedLM

from transformers import AutoTokenizer, AutoModelForMaskedLM

- from transformers import AutoTokenizer, AutoModelForMaskedLM: Mengimpor AutoTokenizer dan AutoModelForMaskedLM.
- AutoTokenizer adalah kelas yang secara otomatis memilih tokenizer yang sesuai dengan model.
- AutoModelForMaskedLM adalah kelas yang secara otomatis memilih model yang sesuai untuk tugas Masked Language Modeling.

tokenizer = AutoTokenizer.from_pretrained("camembert-base")

- AutoTokenizer.from_pretrained("camembert-base"): Memuat tokenizer yang sesuai untuk camembert-base secara otomatis. Ini memungkinkan kita untuk menggunakan berbagai model tanpa harus mengetahui secara eksplisit tokenizer yang diperlukan.

model = AutoModelForMaskedLM.from_pretrained("camembert-base")

- AutoModelForMaskedLM.from_pretrained("camembert-base"): Memuat model camembert-base untuk Masked Language Modeling secara otomatis.

sharing pretrained models

1. Menginstal Paket yang Dibutuhkan

!pip install datasets evaluate transformers[sentencepiece]

!pip install adalah perintah untuk menginstal pustaka-pustaka Python yang diperlukan.

- datasets: Digunakan untuk mengelola dan memproses dataset.
- evaluate: Digunakan untuk mengevaluasi kinerja model.
- transformers[sentencepiece]: Menginstal pustaka transformers dengan dukungan untuk SentencePiece (metode tokenisasi).

2. Menginstal Git Large File Storage (LFS)

!apt install git-lfs

- git-lfs adalah alat untuk menangani file besar di repositori Git. Model-model besar dan file terkait seperti weights model dapat berukuran besar, dan Git LFS digunakan untuk mengelola file besar tersebut dalam repositori Git.

3. Mengatur Konfigurasi Git

!git config --global user.email "you@example.com"

!git config --global user.name "Your Name"

- Mengatur konfigurasi Git global untuk menetapkan email dan nama pengguna yang digunakan untuk identifikasi saat mengunggah ke repositori GitHub atau Hugging Face Hub.

4. Login ke Hugging Face Hub

```
from huggingface_hub import notebook_login

notebook_login()
```

- from huggingface_hub import notebook_login: Mengimpor fungsi notebook_login yang digunakan untuk login ke Hugging Face Hub.
- notebook_login(): Fungsi ini meminta pengguna untuk memasukkan token otentikasi (biasanya tersedia di akun Hugging Face) untuk mengakses dan mengunggah model ke platform.

5. Menyiapkan Argumen Pelatihan

```
from transformers import TrainingArguments

training_args = TrainingArguments(
    "bert-finetuned-mrpc", save_strategy="epoch", push_to_hub=True
)
```

- TrainingArguments: Mengatur argumen untuk pelatihan model.
- "bert-finetuned-mrpc": Nama direktori model yang telah disesuaikan (fine-tuned).
- save_strategy="epoch": Menentukan kapan model disimpan selama pelatihan (dalam hal ini, setelah setiap epoch).
- push_to_hub=True: Menentukan bahwa model ini harus diunggah ke Hugging Face Hub setelah pelatihan.

6. Memuat Model dan Tokenizer

from transformers import AutoModelForMaskedLM, AutoTokenizer

checkpoint = "camembert-base"

model = AutoModelForMaskedLM.from_pretrained(checkpoint)

tokenizer = AutoTokenizer.from_pretrained(checkpoint)

- AutoModelForMaskedLM dan AutoTokenizer: Kelas ini digunakan untuk memuat model dan tokenizer dari model yang sudah dilatih sebelumnya.
- checkpoint = "camembert-base": Menentukan model yang akan digunakan (dalam hal ini, Camembert).
- from_pretrained(checkpoint): Memuat model dan tokenizer Camembert yang sudah dilatih sebelumnya dari Hugging Face Hub.

7. Mengunggah Model dan Tokenizer ke Hugging Face Hub

model.push_to_hub("dummy-model")

tokenizer.push_to_hub("dummy-model")

- push_to_hub("dummy-model"): Mengunggah model dan tokenizer ke Hugging Face Hub dengan nama dummy-model. Model ini kemudian bisa digunakan oleh orang lain.

8. Mengunggah ke Organisasi Hugging Face

tokenizer.push_to_hub("dummy-model", organization="huggingface")

- organization="huggingface": Mengunggah model ke organisasi Hugging Face daripada ke akun pribadi.

tokenizer.push_to_hub("dummy-model",

organization="huggingface", use_auth_token="<TOKEN>")

- use_auth_token="<TOKEN>": Menyertakan token autentikasi jika perlu untuk mengakses repositori yang dilindungi.

9. Menggunakan Hugging Face Hub untuk Manajemen Repositori

from huggingface_hub import (

 # User management

login,

logout,

whoami,

 # Repository creation and management

create_repo,

delete_repo,

update_repo_visibility,

 # And some methods to retrieve/change information about the content

list_models,

list_datasets,

list_metrics,

list_repo_files,

upload_file,

delete_file,

)

- Mengimpor berbagai fungsi dari huggingface_hub untuk mengelola repositori di Hugging Face Hub, termasuk:
- create_repo: Membuat repositori baru.
- delete_repo: Menghapus repositori.
- upload_file: Mengunggah file ke repositori.
- Fungsi lainnya untuk mengelola repositori, pengguna, dan file.

10. Membuat Repositori di Hugging Face Hub

from huggingface_hub import create_repo

create_repo("dummy-model")

- create_repo("dummy-model"): Membuat repositori baru di Hugging Face Hub dengan nama dummy-model.

create_repo("dummy-model", organization="huggingface")

- Membuat repositori di bawah organisasi Hugging Face.

11. Mengunggah File ke Repositori

```
from huggingface_hub import upload_file

upload_file(
    "<path_to_file>/config.json",
    path_in_repo="config.json",
    repo_id="<namespace>/dummy-model",
)
```

- upload_file(): Mengunggah file ke repositori di Hugging Face Hub. File yang diunggah di sini adalah config.json ke dalam repositori dummy-model.

12. Mengelola Repositori dengan Git

```
from huggingface_hub import Repository

repo = Repository("<path_to_dummy_folder>", clone_from="<namespace>/dummy-model")

repo.git_pull()

repo.git_add()

repo.git_commit()

repo.git_push()

repo.git_tag()

repo.git_pull()
```

- Repository: Mengelola repositori lokal dan memfasilitasi interaksi dengan repositori di Hugging Face Hub.
- repo.git_pull(): Mengambil perubahan terbaru dari repositori.
- repo.git_add(): Menambahkan file yang diubah untuk komit.
- repo.git_commit(): Menyimpan perubahan ke dalam repositori lokal.
- repo.git_push(): Mengirimkan perubahan ke repositori di Hugging Face Hub.
- repo.git_tag(): Memberikan tag pada repositori (misalnya, versi model).

13. Menyimpan Model dan Tokenizer Secara Lokal

model.save_pretrained("<path_to_dummy_folder>")

tokenizer.save_pretrained("<path_to_dummy_folder>")

- save_pretrained(): Menyimpan model dan tokenizer ke dalam folder lokal, memungkinkan pengguna untuk memuatnya kembali atau mengunggahnya ke platform lain.

14. Mengkomit dan Mendorong Perubahan ke Repositori

repo.git_add()

repo.git_commit("Add model and tokenizer files")

repo.git_push()

- Setelah menyimpan model dan tokenizer, perubahan tersebut dikomit dan didorong ke repositori Hugging Face Hub.

15. Pelatihan dan Penyimpanan Model

checkpoint = "camembert-base"

model = AutoModelForMaskedLM.from_pretrained(checkpoint)

tokenizer = AutoTokenizer.from_pretrained(checkpoint)

 # Do whatever with the model, train it, fine-tune it...

model.save_pretrained("<path_to_dummy_folder>")

tokenizer.save_pretrained("<path_to_dummy_folder>")

- Melatih dan Menyimpan Model: Setelah model dimuat dan dilatih (fine-tuned), model dan tokenizer disimpan kembali untuk diunggah atau digunakan di masa depan.

Membangun Model Card

Model card adalah file yang sangat penting dalam repositori model, setara dengan file model dan tokenizer. Model card menjelaskan model secara rinci untuk memastikan model dapat digunakan kembali oleh komunitas dan hasilnya dapat direproduksi. Ini juga menyediakan informasi tentang data, pra-pemrosesan, pasca-pemrosesan, serta keterbatasan dan bias model.

Model card dibuat dalam file README.md (format Markdown), dan biasanya mencakup bagian-bagian berikut:

1. Deskripsi Model

Menyediakan informasi dasar tentang model, seperti arsitektur, versi, penulis, implementasi asli, serta pelatihan dan parameter penting.

2. Penggunaan yang Dimaksud & Keterbatasan

Menjelaskan penggunaan model, termasuk bahasa, bidang, dan domain yang cocok, serta area yang di luar cakupan atau di mana model mungkin kurang optimal.

3. Cara Menggunakan

Memberikan contoh penggunaan model, seperti dengan fungsi pipeline() dan cara menggunakan kelas model dan tokenizer.

4. Data Pelatihan

Menyebutkan dataset yang digunakan untuk melatih model dan memberikan deskripsi singkat tentang dataset tersebut.

5. Prosedur Pelatihan

Menguraikan proses pelatihan yang relevan, termasuk pra-pemrosesan, pasca-pemrosesan, jumlah epoch, ukuran batch, dan tingkat pembelajaran.

6. Metode dan Metrik

Menyebutkan metrik yang digunakan untuk evaluasi model, serta dataset dan split yang digunakan untuk pengujian.

7. Hasil Evaluasi

Memberikan hasil performa model pada dataset evaluasi, termasuk informasi tentang ambang keputusan atau evaluasi pada berbagai ambang untuk penggunaan yang dimaksud.

Membangun model card yang jelas sangat penting agar model dapat digunakan kembali, diuji ulang, dan dipahami oleh komunitas, serta untuk memastikan transparansi, keadilan, dan reprodusibilitas.