Apprentissage Profond

Génération de proverbes

Groupe L34

Élèves :

THEVENET Louis

LEBOBE Timothé

TENE Zacharie

SABLAYROLLES Guillaume

31 Mars 2025

Table des matières

1. Génération de proverbes en anglais

Nous avons choisi de créer un modèle de génération de proverbes. La base de donnée est trouvable dans le dossier `raw_data/` à la racine de ce dépôt GitHub

Voici un exemple de proverbes de notre base d'entraînement :

- He that brings good news, knocks hard.
- Anger and haste hinder good counsel.
- Big thunder, little rain.
- Romeo must die in order to save the love.
- The point is plain as a pike staff.

2. Consitution de la base de données

2.1. Acquisition des données

Puisqu'il est plus simple de trouver des données en langue anglaise, nous avons choisi de nous limiter à cette langue et avons utilisé des scripts Python de scrapping pour récolter des données sur différents sites internet.

Nous avons 3200 proverbes originaux anglais et 35000 proverbes en incluant des proverbes traduits d'autres langues et prévoyons de tester le modèle sur ces deux bases.

2.2. Partionnement des données

Nous avons choisi de partionner les données de la manière suivante qui est un standart dans l'apprentissage profond :

- 80% des données pour l'apprentissage
- 10% des données pour le test
- 10% des données pour la validation

Nous pourrons augmenter la part d'apprentissage si le manque de donnée a un impact trop important.

2.3. Script de chargement des données

Nous avons réalisé un script de téléchargement et traitement des données. Un exemple d'utilisation est donné dans le fichier `main.ipynb`, il suffit d'appeler le fonction `make_dataset.load_data()` qui renvoie les proverbes classés par sources.

```
proverbs_db.txt: 34142 proverbs
proverbs_db_only_english.txt: 2208 proverbs
proverbs_digest.txt: 1000 proverbs
Total: 37350 proverbs
Total length: 1853527 characters


Examples of proverbs:
Money's for buying and a horse is for riding.
A set of white teeth does not indicate a pure heart.
Time discloses the truth.
The earth has ears, the wind has a voice.
The fox will catch you with cunning, and the wolf with courage.
```

3. Création et entraînement du modèle

3.1. Création du modèle

Nous allons faire du fine-tuning à partir du modèle `facebook/opt-125` qui un modèle type GPT-3 basé sur l'architecture Transformers.

On charge le modèle et son Tokenizer à l'aide des fonctions `AutoModelForCausalLM.from_pretrained()` et `AutoTokenizer.from_pretrained()` de la librarie `transformers`.

On décide des sources de proverbes que l'on va utiliser, puis on les fusionne en une seule liste:

```
selected_proverbs_groups = [
    "proverbs_db.txt",
    "proverbs_digest.txt"
]

proverbs = []
for group in selected_proverbs_groups:
    proverbs.extend(all_proverbs[group])
```

On utilise ensuite la librarie `datasets` pour préparer ces données à l'entraînement. La tokenisation du dataset consiste à calculer la taille du proverbe le plus long et ajouter du padding aux autres pour uniformiser les tailles.

On utilise ensuite la librarie `peft` afin d'utiliser la technique LoRA (Low-Rank Adaptation). Ainsi, on ajoute un petit nombre de nouveaux paramètres entraînables au modèle afin de l'adapter à la nouvelle tâche.

Paramètres de la configuration LoRA:

```
LoraConfig(
    r=8,
    lora_alpha=16,
    target_modules=["q_proj", "v_proj"],
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM"
)
```

La méthode `get_peft_model` permet d'obtenir un nouveau modèle à partir de cette configuration de notre modèle initial.

3.2. Entraînement

A l'aide de la librarie `transformers`, on définit les paramètres d'entraînement:

```
TrainingArguments(
    output_dir="./results",
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=1,
    logging_dir='./logs',
    logging_steps=10,
    eval_strategy="no",
    save_strategy="epoch",
    report_to="none"
```

```
)
```

Finalement, on met en commun notre modèle, nos paramètres d'entraînement, notre dataset tokenisé et notre tokeniser via la classe `Trainer` et on peut lancer l'entraînement avec la méthode `train()`.

3.3. Génération

Après entraînement du modèle, on crée un pipeline de génération:

```
generator = pipeline("text-generation", model=model, tokenizer=tokenizer)
```

On peut maintenant utiliser le modèle pour terminer un début de proverbes:

```
prompt = "A"
results = generator(prompt, max_length=max_length, num_return_sequences=3,
do_sample=True, temperature=0.7)

for i, result in enumerate(results):
    print(result['generated_text'])
```

Quelques proverbes obtenus:

- A nice dog is a dog.
- A man who keeps his family safe can never be found.
- A child that has no teeth is a coward.

4. Analyse des résultats

5. Conclusion