Statistiques Appliquées, projet 19 - Rapport de Mi-Parcours

Benjamin Muller *(encadrant)* Boisseau Etienne
Dulcy Olivier Katsoulakis Christos Lavergne Eric

2019-2020

Sommaire

Introduction

Les modèles de langue sont des modèles qui portent sur le langage humain. L'objectif est de générer du texte ressemblant à ce qu'écrirait un humain. Récemment, ils ont connu d'importantes avancées avec l'utilisation de réseaux de neurones profonds. Notre projet vise à étudier ces modèles de langues dits "neuronaux" en détail. En particulier nous nous penchons sur un sous-type appelé "Transformer", dont l'un des exemples phares est le modèle "GPT-2" produit par OpenAI.

1 Etapes du projet

18 octobre 2019 : Début du projet. Suivant les attentes de Benjamin Muller, nous avons convenu d'articuler notre projet en trois étapes. La première étape consiste à mettre à l'écrit la théorie probabiliste d'un modèle de langue. Il s'agit ensuite d'implémenter un modèle "Transformer" à l'aide des deux librairies Deep Learning standard que sont PyTorch et TensorFlow, avant de finalement réaliser une étude empirique des performances intrinsèques des modèles obtenus ainsi que de leurs représentations internes.

24 octobre 2019 : Réunion à l'INRIA. Notre encadrant nous a présenté ses attendus sur le projet. Nous avons commencé à échanger sur les modèles de langue à partir d'articles présentant l'historique général des architectures utilisées jusqu'à ce jour.

21 novembre 2019 : Réunion Skype. Nous avons présenté nos premiers avancements, à savoir la description probabiliste et la programmation d'un modèle de langue n-gram. C'est un modèle historique simple très utilisé avant l'emploi de réseaux de neurones, ce qui nous a permis de nous familiariser avec les modèles de langues. Nous avions également étudié l'architecture des Transformers et particulièrement le Transformer du modèle GPT-2, ainsi que l'évaluation de la performance d'un modèle par une métrique appelée la perplexité.

17 décembre 2019 : Réunion Skype. À ce stade, nous avions mis au propre la description probabiliste de l'architecture de GPT-2 et nous avions engagé sa programmation dans les deux modules TensorFlow et PyTorch. Nous avions également lu plusieurs articles pour commencer à nous ouvrir aux perspectives possibles de la modélisation du langage, notamment les papiers de BERT et de CTRL.

10 février 2020 : Réunion Skype. Nous avons finalisé l'architecture des modèles pour les deux modules ainsi que leur procédure d'entraînement et des librairies génériques de pre-processing des données. Nous avons pu démontrer le bon fonctionnement des modèles sur de petites bases de données dans les deux cas. Par ailleurs, nous avons considéré plusieurs idées d'études futures autour des modèles de langues avec architecture de Transformer et avons commencé à parcourir l'état de l'art sur ces sujets.

4 mars 2020 : Prochaine réunion. Nous nous lançons dans la troisième étape du projet. Nous projetons d'écumer les articles scientifiques pour nous fixer rapidement une étude à effectuer. Il nous faut également lancer l'entraînement de nos modèles de façon plus poussée et étudier leurs sorties.

2 Rapport écrit

Au fur et à mesure de notre avancement, nous maintenons un rapport résumant et formalisant toutes les notions que nous étudions. Nous avons commencé par définir mathématiquement les modèles de langue à l'aide de la théorie des probabilités. Ensuite, nous avons formalisé une mesure de performance pour évaluer la pertinence d'un modèle de langue (la perplexité), avant de décrire l'architecture de modèle de langue neuronaux sur laquelle nous nous sommes concentrés : le Transformer proposé par OpenAI dans l'article "Language Models are Unsupervised Multitask Learners" (modèle GPT-2).

La première partie sur les modèles de langue est théorique. Nous posons un cadre mathématique pour définir un modèle de langue. Nous définissons l'espace des textes possibles, puis définissons un modèle de langue comme une loi de probabilité sur cet espace. A partir de ce cadre général, il découle des cas plus spécifiques. Nous formalisons d'abord les modèles n-grams, qui sont des modèles de langue supposant que la probabilité d'occurrence d'un mot ne dépend que des n mots qui le précèdent; nous exposons les avantages et les inconvénients d'un tel modèle. Dans un second temps, nous présentons les modèles de langue neuronaux qui sont un type de modèle plus efficace mais plus complexe.

Après avoir choisi un modèle, on s'intéresse à la génération d'un échantillon de textes plausibles (ayant une probabilité d'occurrence suffisamment élevée). Nous présentons alors une méthode naïve (force brute), la méthode gloutonne, puis la méthode Beam Search et la méthode d'échantillonnage itératif.

Une fois le modèle de langue ainsi qu'une stratégie de génération de texte choisis, nous définissons une métrique pour évaluer la performance de notre modèle de langue: il s'agit de la perplexité. Basé sur la notion d'entropie, elle permet de comparer plusieurs modèles de langue sur leur capacité à prédire un texte.

Enfin, nous présentons l'architecture du Transformer telle qu'elle est proposée dans l'article d'OpenAI. Nous exposons un schéma de l'architecture du Transformer, puis nous détaillons le fonctionnement de chaque composante de cette architecture, notamment l"'Attention", mécanisme central du modèle.

3 Organisation du code

Les modèles de langues neuronaux les plus performants aujourd'hui sont programmés en Python, à l'aide de l'un des deux modules dominants: PyTorch et TensorFlow.

Les deux ont leurs avantages et leurs inconvénients mais tous deux sont intéressants à connaître. Plutôt que d'en choisir un, nous avons donc décidé de nous séparer en deux équipes et de coder le modèle deux fois, une fois pour chacun des deux outils.

L'avantage de procéder ainsi a été que chacun participe directement au développement du Transformer : se séparer en binômes permet à chacun de contribuer davantage au code et ainsi d'acquérir une compréhension solide des modèles que nous manipulons.

Notre code est organisé comme suit :

- Modèles n-gram
- Transformers
  - Code écrit avec TensorFlow
  - Code écrit avec PyTorch
- Commun
- Traitement des données
- Échantillonnage (méthodes de génération de texte à partir d'un modèle)

La totalité du code produit jusqu'à présent fait environ 1200 lignes de code Python, dont environ la moitié est contenue dans la partie "Transformers".

Pour partager et organiser notre code, nous avons utilisé l'outil de collaboration Git, et mis en place un dépôt de code commun sur GitHub qui nous permet de suivre toutes les contributions faites par les membres et d'accéder aux anciennes versions du code si besoin.

Le code des deux modèles est d'ores et déjà fini. Pour vérifier le bon fonctionnement de nos modèles, nous les avons entraînés sur un petit échantillon de données pour aboutir à un surapprentissage, c'est-à-dire l'équivalent d'un apprentissage par coeur des données par le modèle. Nos modèles ayant passé cette vérification, nous pouvons procéder à l'entraînement sur une plus grande base de données pour tenter de générer du texte réaliste.

4 Perspectives

Les modèles codés étant fonctionnels, nous allons maintenant les entraîner sur un plus gros volume de données à l'aide des serveurs de l'association d'informatique Tuxae ou des cartes graphiques de la salle informatique 2014. Nous pourrons ainsi

comparer les performances de la génération de textes de nos modèles n-gram et de nos modèles Transformer (TensorFlow et PyTorch).

Par ailleurs, nous nous intéresserons aux performances de modèles déjà entraînés tels que GPT-2 et BERT en mettant en perspective diverses considérations telles que la performance, le nombre de paramètres, le corpus considéré...

Enfin, notre encadrant nous a proposé d'élargir notre projet en travaillant sur une thématique supplémentaire dans la perspective de publier un article de recherche (en plus d'enrichir notre rapport). Nous sommes à l'étape de la définition du sujet, les principales pistes sont :

- Contraindre les modèles au moment de l'inférence
- Analyser l'attention pendant l'inférence

Nous prévoyons de statuer sur le sujet final lors de notre prochaine réunion avec notre encadrant (mercredi 26 février).