Prof. Amal Zouaq
Chargés de laboratoire : Dylan Farvacque & Antoine Lefebvre-Brossard

École Polytechnique de Montréal
Département Génie Informatique et Génie Logiciel
INF8460 – Traitement automatique de la langue naturelle

Projet INF8460
Automne 2020

1. DESCRIPTION

Dans ce cours, vous devez effectuer un projet final en équipes de trois. Ce projet vous permettra d'explorer un sujet du TAL plus en profondeur que ce qui est couvert par les cours magistraux et devoirs.

Vous devez implémenter un système fonctionnant avec des données en langage naturel, dans le but de tester certaines hypothèses sur le langage ou la technologie TAL. Vous devez également écrire un article scientifique décrivant votre approche.

Dans ce cours, l'idée est d'effectuer l'implémentation d'un système de question-réponse, c'est-à-dire qui est capable d'extraire une réponse à une question à partir d'un corpus de documents. Le projet sera organisé sous forme d'une compétition Kaggle.

Qu'est-ce que la tâche de question-réponse?

La tâche de question-réponse est la tâche de répondre automatiquement à une question factuelle posée dans un langage naturel. Dans le contexte de ce projet, les questions sont en anglais.
Afin de trouver la réponse à la question, il vous faudra être capable de trouver des documents qui sont pertinents pour la question puis extraire la réponse d'un ou des documents. Voici un exemple :

Entrée : Question : What causes precipitation to fall?
Étape 1 - trouver un passage pour répondre à la question : In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers".
Étape 2 – extraire la réponse à la question dans le passage : Réponse : gravity

2. LIBRARIES PERMISES

- Jupyter notebook
- NLTK
- HugginFace
- Pytorch

3. INFRASTRUCTURE

- Pour entraîner des modèles avec des approches neuronales. Pour ce faire, vous pouvez utiliser Google Colab. Cependant n'oubliez pas que les données ne sont pas persistantes entre les sessions, vous devez donc vous assurer de sauvegarder vos modèles.

- Vous pouvez aussi utiliser Kaggle qui offre environ 40h de GPU time/semaine (par personne). Notez que dans Kaggle, il vous est possible d'avoir jusqu'à 100GO de données persistantes privées.
- Finalement, il est aussi possible d'utiliser les GPU du local L4712. Dans ce cas, vous devez utiliser le dossier temp (voir le tutoriel VirtualEnv.pdf du tp3)

4. ECHÉANCE

Fin de la session. La date précise sera indiquée sur Moodle.

5. KAGGLE

Le projet se fera sous forme d'une compétition Kaggle. Vous devrez utiliser l'environnement Kaggle pour la soumission et l'évaluation de vos approches.
Pour tester votre système au fur et à mesure, vous aurez le droit à 4 soumissions par jour sur Kaggle. Vous verrez deux types de résultats sur votre « private leaderboard » et votre « public leaderboard » :
Le « private leaderboard » est calculé sur approximativement 70% des données de test et n'est visible qu'à la fin de la compétition. Le résultat final sera basé sur ce leaderboard. Si aucune soumission n'est choisie, la soumission avec le meilleur score sur le « public leaderboard » sera utilisée pour calculer le score sur le « private leaderboard ».
Le « public leaderboard » est calculé sur approximativement 30% des données de test, choisies aléatoirement par Kaggle. Ce score est public et est calculé sur la même tranche de donnée pour tous les participants.
Pour l'évaluation, vous devrez soumettre un fichier de données `submission.csv` du même format que le fichier `sample_submission`.csv (disponible sur le site de la compétition). Submission.csv devra contenir pour chaque ligne de votre ensemble de test, le passage et la réponse retournée par votre approche, selon le format indiqué dans la compétition.
Dans le projet INF8460, notre classement comprendra deux leaderboards distincts:

- Un leaderboard pour les modèles qui utilisent un Plongement Contextuel Pré-entrainé (PCP) comme BERT pour la représentation de passages et l'ordonnancement. Si vous utilisez un modèle PCP uniquement dans la partie extraction de réponse, vous pouvez participer au leaderboard Non-PCP aussi.
- Un leaderboard pour les modèles non-PCP uniquement (notez que les vecteurs de mots comme word2vec, GloVe et FastText sont autorisés dans la division non-PCP)

Nous avons décidé de cette distinction principalement pour deux raisons:
1. Les équipes qui utilisent PCP devront télécharger du code et importer des poids pré-entrainés, ce qui signifie que de grandes parties de leur code proviendront de l'extérieur. Pour ces équipes, nous concentrerons notre évaluation sur ce que l'équipe a ajouté en plus du modèle PCP.
2. Les approches PCP sont grandement susceptibles de surpasser les meilleurs modèles non PCP. Bien que cela soit très performant, le fait d'utiliser BERT ou un modèle PCP à lui seul ne vous donnera pas une note élevée dans ce projet. Vous devez vous concentrer sur des moyens plus créatifs pour améliorer les performances, de BERT par exemple.
Chaque équipe pourra décider de soumettre soit à un leaderboard, soit aux deux.

6. DESCRIPTION DES DONNEES

Dans ce projet, vous utiliserez deux ensembles de données et un corpus. Chaque jeu de données est décomposé en données d'entrainement (train), de validation (dev) et de test (test).
Nous ne mettrons à votre disposition que les données d'entrainement et de validation. Les données de test ne contiennent pas la réponse correcte et doivent être complétées avec les résultats de votre système. Les résultats que vous verrez sur le leaderboard Kaggle seront calculés sur une portion des données de test

durant la compétition et sur toutes les données de test à la fin pour l'évaluation finale (voir la section Kaggle).

Inf8460_A20_corpus :
Nous mettons à votre disposition un corpus de passages que vous **devez** utiliser lors de la recherche de passages. Votre système complet ne doit utiliser que ce corpus et rien d'autre.

Inf8460_A20_dataset :
Nous vous fournissons un ensemble de données qui associe une question, un passage, et une réponse qui est directement extraite du passage. C'est cet ensemble de données que nous utiliserons pour la compétition Kaggle et pour l'évaluation de votre performance finale dans la compétition. Notez que certains passages contiennent des balises HTML et qu'il vous faudra potentiellement procéder à un prétraitement de ces passages pour les enlever.
Cet ensemble de données est composé de trois sous-ensembles :

- Train : ensemble d'entraînement que vous allez principalement utiliser pour entraîner votre système d'extraction de données. Il est de la forme <QuestionID, Question, PassageID, Réponse>. Le but est donc d'entrainer votre modèle à extraire la réponse à la question dans le passage
- Validation : De la même forme que le Train, il vous permet de valider votre entraînement et de tester les performances de certains modules.
- Test : Un ensemble secret qui est utilisé pour évaluer votre système complet (Recherche de passage + extraction de réponse). Il est de la forme <QuestionID, Question>. Votre système doit trouver dans le corpus **Inf8460_A20_corpus** un document pertinent afin d'en extraire la réponse et reporter la réponse dans un fichier de soumission `submission.csv` de la forme <QuestionID, PassageID, Réponse>.

7. ETAPES DU PROJET

A partir du notebook inf8460_projet qui est distribué, vous devez réaliser les étapes suivantes. (Noter que les cellules dans le squelette sont là à titre informatif il est fort probable que vous rajoutiez des sections au fur et à mesure de votre projet).

7.1. Représentation et recherche de passages

Les passages et questions de votre ensemble de données doivent d'abord être représentés et indexés pour ensuite pouvoir faire une recherche de passage pour répondre à une question.

a) Vous devez tout d'abord implanter un « baseline » qui est de représenter chaque document et question en utilisant un TF-IDF des n-grammes qu'ils contiennent. N'oubliez pas de prendre en compte le pré-traitement du corpus, la réduction de dimension, et toute technique que vous jugerez utile pour pouvoir améliorer la tâche de question-réponse finale.

b) Dans un deuxième temps, vous devez utiliser des modèles plus sophistiqués de représentation, tels que les plongements lexicaux pré-entrainés comme GloVe ou fastText, ou des représentations de phrases ou de caractères ou des modèles de plongements contextuels pré-entrainés. Certaines représentations peuvent être optimisées en tenant compte de la question et du passage. Vous pouvez aussi proposer des techniques plus avancées en regardant ce qui est fait dans l'état de l'art et/ou en présentant une idée originale. N'oubliez pas de citer vos sources (articles, code) si vous en utilisez.

7.2. Ordonnancement des passages

Maintenant que vous avez une représentation de vos passages et de la question, il faut être capable de déterminer quel passage sera le plus pertinent pour la question posée. Il vous faut donc retrouver un top-N de passages utiles pour répondre à la question. Ces passages devront être ordonnés du plus pertinent au moins pertinent.

Dans une approche de base, vous pouvez évaluer la similarité entre la représentation de la question et celle de chaque passage et retourner les N passage les plus similaires où N est le plus petit nombre possible vous permettant la meilleure performance finale.
Suivant votre représentation de passage, plusieurs techniques de calcul de similarité sont possibles.
Pour des représentations creuses, un simple produit scalaire ou cosinus entre les deux représentations est possible. Il est aussi possible d'utiliser des mesures de distance (distance euclidienne, Manhattan, cosinus, etc.).
Il est aussi possible d'utiliser une approche basée sur l'apprentissage machine afin de déterminer un score de pertinence d'un passage pour une question donnée.
Finalement, rien ne vous empêche d'implémenter un ordonnancement qui combinera une approche traditionnelle utilisant un calcul de distance ou un produit scalaire afin de trouver un premier top-N passages suivie d'une approche basée sur l'apprentissage machine sur ces passages, d'utiliser des attributs (*features*) pour identifier les plus pertinents parmi ceux sélectionnés précédemment et déterminer le top-N final.
A cette étape, vous devez produire un fichier `passage_submission.csv` qui contient pour toutes les questions de l'ensemble de test le top-N des passages retournés par votre système pour y répondre. Le fichier doit respecter le format suivant <QuestionID, PassageID, Rang> :
<Question, Passage1, Rang1>
….
<Question, PassageN, RangN>
Notez que le N sera à déterminer en fonction de votre système mais qu'il doit être le plus petit possible pour obtenir le maximum de points.

7.3. Extraction de réponses

À cette étape vous devriez avoir N documents ordonnés pour une question donnée. Il ne vous reste alors plus qu'à trouver la bonne réponse dans tous ces documents.
Pour ce faire, les systèmes actuels emploient des approches tirées de la tâche de MRC (Machine Reading Comprehension). En résumé, pour déterminer la position de départ et de fin de la réponse dans un document, on utilise un réseau de classification qui a pour but, pour chaque token du document, de le classifier comme token de début ou de fin. La ressource suivante explique comment ajouter ce genre de classificateur en aval d'un modèle de langue comme BERT :
https://www.youtube.com/watch?v=l8ZYCvgGu0o

Pour ce projet, nous voulons aussi que le modèle responsable de l'extraction de réponse soit capable de dire qu'un document ne contient pas la réponse à la question. Dans ce cas il doit retourner <No Answer>.

Si vous jetez un coup d'œil au classement SQuAD qui est une tâche similaire dans l'extraction de réponse (https://rajpurkar.github.io/SQuAD-explorer/), vous remarquerez que BERT [https://arxiv.org/pdf/1810.04805.pdf] (ou ALBERT [https://arxiv.org/pdf/1810.04805.pdf%5d]) sont utilisés dans les meilleurs projets.
Une fois que vous avez des réponses candidates pour chaque document, il faut sélectionner la meilleure réponse. Pour ce faire, plusieurs techniques existent : Vous pouvez prendre la meilleure prédiction parmi les candidats (soit à partir de leur score normalisé ou leur logit). Il est aussi possible de combiner toutes les réponses identiques provenant de différents documents, il faut alors combiner les scores. Finalement il existe aussi des techniques plus avancées comme la normalisation globale des scores des candidats (https://arxiv.org/pdf/1908.08167.pdf).
Aussi, n'oubliez pas que certaines questions n'ont tout simplement pas de réponse possible, il faut donc les détecter et indiquer <No Answer>.

7.4. Pipeline complet

Au final, votre pipeline devra retourner une réponse à une question ou indiquer qu'il n'y en a pas. C'est ce que nous évaluerons dans Kaggle.

7.5. Le plus

Si vous utilisez une approche standard à la BERT uniquement dans l'extraction de réponse, vous pourrez atteindre jusqu'à 80% des points pour cette étape. Les 20% restants sont ce que vous pourrez proposer d'innovateur et de créatif pour améliorer les performances de votre système.
Notez que vous pouvez/devez être créatifs et innovateurs dans chaque étape du pipeline : représentation des passages, ordonnancement et extraction de réponses.

7.6. Évaluation

Bien que la compétition Kaggle ne juge que l'extraction de réponse finale du pipeline, vous devez aussi présenter dans votre article une évaluation des étapes suivantes :

Recherche et Ordonnancement des passages sur le jeu de validation	**Extraction de réponse sur le jeu de validation**	**Système complet sur le leaderboard de Kaggle**
Précision top-10 Précision top-N	EM et F1	EM

- Pour la recherche et l'ordonnancement de passages, vous devez nous indiquer la précision top-10 et la précision top-N où N correspond au nombre de passages nécessaires pour obtenir la meilleure performance possible de votre système à cette étape.
- L'étape d'extraction de réponse doit évaluer la performance de votre système sur l'ensemble de validation de in8460_dataset. Elle devra être évaluée avec deux métriques: le score Exact Match (EM) et le score F1.
  - Exact Match est une mesure binaire (c'est-à-dire vrai / faux) qui indique si la sortie du système correspond exactement à la réponse (ground truth). Par exemple, si votre système a répondu à une question par «Trudeau» mais que la réponse était «Justin Trudeau», vous obtiendrez un score EM de 0 pour cet exemple. C'est une métrique stricte.
  - F1 est une métrique moins stricte - c'est la moyenne harmonique de précision et de rappel. Dans l'exemple « Trudeau », le système aurait une précision de 100% (sa réponse est un sous-ensemble de la réponse) et 50% de rappel (il ne comprenait qu'un seul des deux mots dans la réponse), donc un F1 score de 2 × prédiction × rappel / (précision + rappel) = $2 * 50 * 100 / (100 + 50)$ = 66,67%.
  - Lorsqu'une question n'a pas de réponse, les scores F1 et EM sont tous deux de 1 si le modèle prédit une non-réponse et de 0 sinon.
  - Vous pouvez vous inspirer du script de SQuAD2 pour obtenir ces mesures : https://worksheets.codalab.org/rest/bundles/0x6b567e1cf2e041ec80d7098f031c5c9e/contents/blob/
  - La performance du système complet sur Kaggle sera mesurée sur l'extraction de réponse finale de votre pipeline au moyen de la métrique CategorizationAccuracy qui correspond à la métrique EM définie ci-dessus Vous devez comparer votre performance dans votre article à celle des deux meilleures équipes du leaderboard de Kaggle.

LIVRABLES

Vous devez remettre sur Moodle:

1- *Le code* : Un Jupyter notebook en Python qui contient le code tel que soumis dans l'environnement Kaggle implanté avec les librairies disponibles pour ce cours (Python, Keras, NLTK, scikitLearn, etc.) ainsi que votre fichier de soumission de données de test. Le code doit être exécutable sans erreur et accompagné des commentaires appropriés dans le notebook de manière à expliquer les différentes fonctions et étapes dans votre projet. Un fichier requirements.txt doit indiquer toutes les librairies / données nécessaires. Les critères de qualité tels que la lisibilité du code et des commentaires sont importants. Nous nous réservons le droit de demander une démonstration ou la preuve que vous avez effectué vous-mêmes les expériences décrites dans votre rapport. *Attention, en aucun cas votre code ne doit avoir été copié de projets potentiellement existants*.

2- *Un lien GoogleDrive* vers les modèles nécessaires pour exécuter votre notebook

3- *Votre fichier submission.csv et votre fichier passage_submission.csv*

4- *Un article* : Vous devez rédiger et soumettre un rapport décrivant votre projet sous forme d'article. Le rapport doit être structuré comme un document de conférence typique, et suivre le format indiqué dans la section *Format de l'article.* Vous devez utiliser la version de soumission du style NAACL disponibles sur https://naacl2019.org/calls/papers/. Vous pouvez utiliser le modèle LaTeX ou Word. La longueur de votre rapport doit être de 5 pages de contenu. Cela comprend le texte, les chiffres, annexes et équations, et les références.

5- Un document *contributions :* Décrivez brièvement la contribution de chaque membre de l'équipe. Tous les membres du projet sont censés contribuer au projet. Bien que chaque membre puisse effectuer différentes tâches, vous devez vous efforcer d'obtenir une répartition égale du travail. En particulier, tous les membres du projet devraient participer à la conception du projet, être impliqués dans la rédaction et participer activement à la réflexion et à l'implémentation du code.

SOUMISSION FINALE

Votre projet final sera remis à la fin du cours à une date fixée sur Moodle. Cela correspondra à la date de la fin de la compétition.

EVALUATION DU PROJET

De manière générale, votre niveau d'effort sera démontré par la richesse et la difficulté des approches que vous proposerez, ainsi que par leur originalité.
Bien que vous ne soyez pas obligés de proposer une idée originale, les meilleurs projets se distingueront par leur originalité (et pourront faire potentiellement l'objet d'une publication). L'originalité ne doit pas nécessairement être une approche complètement nouvelle - des modifications modestes mais bien motivées des modèles existants sont très utiles, surtout si elles sont accompagnées d'une bonne analyse. Si vous pouvez montrer quantitativement et qualitativement que votre changement améliore un modèle de l'état de l'art (et encore mieux, explique quel problème particulier il résout et comment), alors vous aurez rempli les objectifs du projet.

Plus précisément, votre projet sera évalué sur les points suivants:

Code : (65%)
Le 65% sera distribué comme suit : Représentation et recherche de passages (25%), Ordonnancement de passages (15%), Extraction de réponse (25%)

- Richesse des architectures et des modèles proposés
- Originalité
- Implantation d'un système de référence (baseline) à des fins de comparaison
- Exécution correcte du code

- Qualité du code
- Documentation

Article (35%)

- Profondeur du contenu et richesse du projet
- Justification de la contribution et de l'apport par rapport aux travaux antérieurs
- Méthodologie : claire, correcte, complète, reproductible
- Analyse générale de l'état de l'art et positionnement des approches proposée par rapport aux limites de l'état de l'art
- Rédaction : clarté, concision
- Citation de sources
- Utilisation d'ensembles de données et d'outils/librairies appropriés

FORMAT DE L'ARTICLE

Vous trouverez ci-dessous des indications sur la rédaction de votre rapport pour le projet final. Veuillez l'utiliser comme guide général pour structurer votre rapport.
Un article en TAL expérimental "standard" comprend les sections suivantes:

1. Introduction

Motivez et décrivez de manière abstraite le problème que vous abordez et comment vous le résolvez. Quel est le problème? Pourquoi est-ce important? Quelle est votre approche de base? Une brève discussion sur la manière dont cela s'inscrit dans des travaux connexes dans le domaine est également souhaitable. Résumez les résultats de base et les conclusions que vous présenterez.

2. Méthodologie

2.1 Définition de la tâche/objectif

Définissez précisément le problème que vous abordez (c'est-à-dire spécifiez formellement les entrées et les sorties). Expliquez pourquoi il s'agit d'un problème intéressant et important.

2.2 Définition de l'algorithme/méthode/technique

Décrivez de manière assez détaillée l'algorithme que vous utilisez pour résoudre ce problème. Une description pseudocode de l'algorithme que vous utilisez est souvent utile. Illustrez à travers un exemple concret, en montrant comment votre algorithme traite cet exemple. L'exemple doit être suffisamment complexe pour illustrer tous les aspects importants du problème, mais suffisamment simple pour être facilement compris. Dans votre cas, on s'attend à ce que vous décriviez les étapes de représentation de passage, d'ordonnancement et d'extraction de réponse ainsi que la partie « Plus ».

3. Évaluation expérimentale

3.1 Ensemble de données, métriques et baselines

Décrivez la méthodologie expérimentale que vous avez utilisée. Quels sont les critères que vous utilisez pour évaluer votre méthode? Quelles hypothèses spécifiques votre expérience teste-t-elle? Quelles sont les variables dépendantes et indépendantes? Quelles sont les données d'entrainement/ test utilisées? Quelles mesures de performance avez-vous collectées et comment les présentez-vous et les analysez-vous? Les comparaisons avec des méthodes concurrentes abordant le même problème sont particulièrement importantes. Dans votre cas, n'oubliez pas de présenter les résultats de votre baseline TF-IDF et de approches Non-PCP pour la représentation des passages. Idéalement, vous devriez avoir aussi un baseline pour chaque partie de votre pipeline. A vous de voir lequel et de le décrire.

3.2 Résultats

Présentez les résultats quantitatifs de vos expériences. Les données graphiques sont souvent meilleures que les tableaux mais pas toujours. Quelle est la performance de votre approche comparée à un « baseline » ou à un ou des systèmes de l'état de l'art ? Si vous avez obtenu de meilleurs résultats, la différence est-elle statistiquement significative ? Ici vous devez vous comparer brièvement aux résultats Kaggle et indiquer les résultats de votre « baseline » et de chacune des étapes du pipeline tel qu'indiqué dans le projet.

3.3 Discussion
Votre hypothèse est-elle vérifiée? Quelles conclusions sur les forces et les limites de votre méthode par rapport aux autres méthodes? Comment expliquer les résultats en termes de propriétés sous-jacentes de l'algorithme et / ou des données ?

4. Travaux connexes
Vous devriez citer des sources appropriées et pertinentes. Répondez aux questions suivantes pour chaque travail connexe qui aborde le même problème ou un problème similaire. Quel est leur problème et leur méthode? En quoi votre problème et votre méthode sont-ils différents? Pourquoi votre problème et votre méthode sont-ils meilleurs?

5. Travaux futurs et conclusion
Quelles sont les principales limites de votre méthode actuelle? Pour chaque limite, proposer des ajouts ou des améliorations qui permettraient de les résoudre.
Résumez brièvement les résultats importants et les conclusions présentés dans le document. Quels sont les points les plus importants illustrés par votre travail? Comment vos résultats vont-ils améliorer la recherche future et les applications dans le domaine?

CODE D'HONNEUR

Règle 0: Le plagiat de code est bien évidemment interdit.
Règle 1: Vous êtes autorisés à utiliser le code, les bibliothèques ou les données existants de type PCP que vous souhaitez. Cependant, vous devez clairement citer vos sources et indiquer quelles parties du projet sont votre travail. Si vous utilisez ou empruntez du code à des bibliothèques externes, décrivez comment vous utilisez le code externe et fournissez un lien vers la source dans votre article. Vous serez noté en fonction de ce que vous ajoutez au travail des autres.
Règle 2: Vous êtes libres de discuter des idées et des détails de mise en œuvre avec d'autres équipes. Cependant, vous ne pouvez en aucun cas consulter le code d'une autre équipe INF8460, ou incorporer leur code dans votre projet.
Règle 3: Vous ne pouvez pas partager votre code publiquement (par exemple, dans un dépôt GitHub public) tant que le cours n'est pas fini.