M2 SDSC

UFR Mathématique et informatique

Université de Strasbourg

Mémoire Projet Master

Recherche d'incohérence dans un texte

Réalisé par:

- Amaury SENSENBRENNER
- Amine BELQASMI
- Juba TOUAM
- Nicolas WEISS

Encadrants:

- Aurélie LEBORGNE
- Gabriel FREY

1. Présentation du sujet

Le sujet que nous avons choisi de présenter ici est la recherche d'incohérences dans un texte. Pour faire simple, dans de nombreuses phrases, il est possible d'identifier une prémisse (condition logique sur laquelle l'argument s'appuie) et une hypothèse (conjecture réaliste s'appuyant sur la prémisse) par des méthodes d'apprentissage automatique. Il nous était indiqué d'utiliser les méthodes d'inférence à partir de langage naturel (NLI) afin de déterminer automatiquement les couples prémisse/hypothèse de différentes phrases d'un texte. À partir de là, il nous était demandé de mettre en place une application d'analyse de texte, d'annotation du texte et d'identifications et d'élimination de contradictions/redondances. L'application pourra être utilisée pour aider lors de l'écriture de textes ou de messages sur les réseaux sociaux (type Twitter).

Nous pouvons identifier trois grandes phases dans ce sujet :

- Création puis développement d'un modèle NLI (Natural Language Inference) pour identifier les couples prémisse/hypothèse et effectuer une classification d'un couple de phrases (implication, contradiction, neutre).
- Création d'une interface graphique afin d'accueillir notre modèle NLI. C'est-à-dire, un texte en entrée pour obtenir aussi un texte en sortie.
- Ajout d'options dans notre interface graphique afin que l'utilisateur puisse manipuler au mieux le modèle et l'interface graphique (après avoir implémenter le modèle dans l'interface).

La dernière étape sera de faire une fusion entre les deux premières phases afin que notre application soit complète.

2. Etat de l'art

Avant de commencer notre implémentation d'un modèle NLI (Natural Language Inference), nous avons fait des recherches sur des modèles déjà existant dans la littérature NLI. Nous avons décidé dans un premier temps d'utiliser les liens de documentations que nos professeurs nous ont fourni pour le projet. Le site NLP-progress fournit une dizaine de modèles qui obtiennent les meilleurs résultats dans leur domaine. Nous avons décidé de faire un état de l'art sur tous les modèles que propose le site afin de voir ce qu'il se fait de mieux dans ce domaine et dans plus spécialisation. Lors de cette analyse, nous avons aussi utilisé le site GLUE qui est le référentiel d'évaluation de la compréhension générale du langage. C'est une collection d'ensembles de données utilisés pour la formation, l'évaluation et l'analyse des modèles de NLP les uns par rapport aux autres, dans le but de conduire "la recherche dans le développement de systèmes généraux et robustes de compréhension du langage naturel". La collection se compose de neuf ensembles de données de tâches «difficiles et diverses» conçues pour tester la compréhension du langage d'un modèle, et est essentielle pour comprendre comment les modèles d'apprentissage par transfert comme le BERT sont évalués. Pour plus de détails sur les différents tests effectués sur GLUE afin de classifier un modèle, nous vous invitons à consulter le site web GLUE.

Grâce à ce site qui possède un classement des meilleurs modèles en fonction de leurs scores, nous avons pu étudier quelques modèles qui font partie du classement afin de savoir comment ils fonctionnent. Nous avons donc utilisé, pour notre état de l'art, les modèles suivants :

- BiLSTM+Attn
- CAFE
- GenSen
- RoBERTa
- Snorkel Metal
- Finetuned Transformer LM
- MT-DNN-ensemble
- Hierarchical BiLSTM Max Pooling
- XLNet-Large

Nous avons résumé, pour chacun d'eux, leurs positions dans le classement GLUE ainsi que le domaine où ils sont le plus efficaces. Nous avons aussi proposé un petit résumé des architectures des modèles ainsi que comment ils fonctionnent. Pour avoir plus d'informations sur chacun des modèles présentés ci-dessous, je vous invite à regarder le GIT où se trouve un résumé plus détaillé que ceux que vous pouvez trouver ci-dessous dans le dossier état de l'art, fait par nous même, de chaque modèle. Certains modèles sont spécialisés dans des tâches précises donc ils n'obtiennent pas forcément un bon score général mais ils peuvent être utiles dans notre cas.

BiLSTM+Attn:

Ce modèle ressemble très fortement au modèle Multi-task BiLSTM + Attn en ce qui concerne l'architecture, le modèle utilise un BiLSTM (LSTM bidirectionnels) à deux couches, 1500D avec max pooling et 300D GloVe word embeddings. Pour les tâches à une seule phrase, nous encodons la phrase et passons le vecteur résultant à un classificateur. Pour les tâches de paires de phrases, nous encodons les phrases indépendamment pour produire des vecteurs u, v, et passons [u , v ; |u - v| ; u * v] à un classificateur (détail de cette phase dans un autre état de l'art). Le classifieur est un MLP avec une couche cachée de 512D.

CAFE:

CAFE possède une architecture d'apprentissage profond pour l'inférence en langage naturel (NLI). L'objectif principal du modèle est, étant donné une prémisse et une hypothèse, de détecter si la seconde implique ou contredit la première.

L' architecture du modèle prend en entrée une prémisse et une hypothèse qui sont ainsi comparées, compressées puis propagées vers des couches supérieures pour obtenir un apprentissage amélioré de la représentation. Ensuite, des couches de factorisation sont adaptées pour une compression efficace et expressive des vecteurs d'alignement en caractéristiques scalaires, qui sont ensuite utilisées pour augmenter les représentations des mots de base.

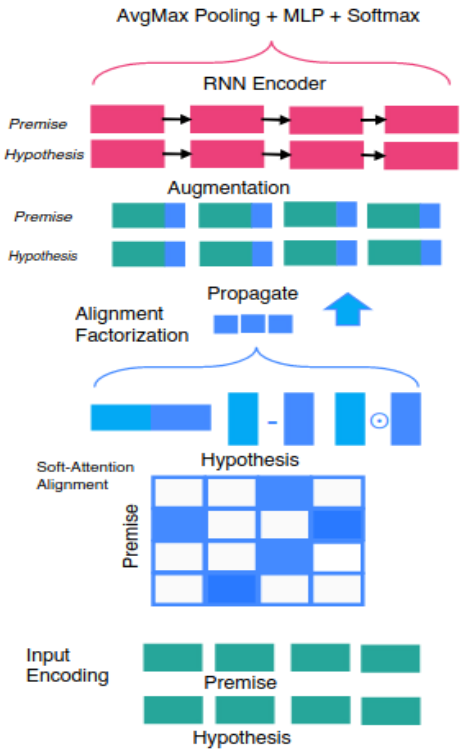


Figure 1: Architecture du modèle Cafe

GenSen:

GenSen est une technique pour apprendre des objectifs, des représentations de phrases en utilisant un apprentissage multitâche.

Concernant la préparation de l'entraînement multitâches :

Nous avons besoin d'un certain nombre de tâches, d'un encodeur partagé pour toutes les tâches et un nombre de décodeurs (du même nombre que celui des tâches). Maintenant, les prochaines étapes vont boucler : Nous récupérons une tâche i et son jeu de données associé pour créer des paires de valeurs entrée/sortie. La valeur d'entrée va alors être encodée pour créer une représentation qui va ensuite passer dans le décodeur associé à la tâche, pour générer une prédiction.

Le modèle utilisé est comme suit :

L'encodeur partagé utilise une table de recherche de mots communs et un GRU (Gated Recurrent Unit) qui est en quelque sorte un réseau de neurones artificiel qui peut non seulement utiliser des données simples mais également des séquences de données entières comme de la vidéo

RoBERTa:

BERT est un modèle de langage naturel. C'est un modèle qui modélise la distribution de séquences de mots ou de lettres. Il a par exemple été utilisé dans la prédiction d'un mot en fonction des mots qui le précédaient.

RoBERTa est une méthode d'apprentissage basée sur BERT mais améliorée, qui atteint des résultats similaires voire supérieurs à bon nombre de méthodes. Ces améliorations consistent notamment en l'agrandissement des données d'entraînement (et donc en entraînant le modèle plus longtemps), en enlevant l'objectif initial qui était la prédiction des mots suivants, en l'entraînant sur des phrases plus longues, et en changeant dynamiquement les motifs (les patterns) des données d'entraînement.

BERT prend en entrée 2 segments (chaque segment constitué de plus d'une phrase) séparés par des tokens spécifiques : Les segments étant x1,...,xn et y1,...,ym : [CLS],x1,...,xn,[SEP],y1,...,ym,[EOS] BERT aura alors 2 objectifs pendant l'entraînement :

- **Masked Language Model** (MLM) qui consiste à remplacer aléatoirement des tokens dans la donnée d'entrée par le token [MASK] afin d'ensuite prédire quel était le token d'origine.
- **Next Sentence Prediction** (NSP) qui consiste à prédire si deux segments se suivent ou non dans le texte d'origine.

Snorkel Metal:

MeTal est un framework utilisé pour la modélisation et l'intégration de sources de supervision faible avec des exactitudes, des corrélations et des granularités inconnues.

Voici comment MeTal fonctionne :

L'utilisateur donne en entrée un graphe de tâches labellisées (Y) (montrant les liens entre chaque tâche), un ensemble de points de données non labellisés (X), un ensemble de sources de supervision faible multitâches (S) (qui retournent tous un vecteur de labels de tâches pour l'ensemble de données précédent) et un deuxième graphe : la structure de dépendance entre ces sources.

Figure2: Schéma de MeTaL pipeline

L'architecture consiste en :

- Une couche d'entrée LSTM (Long Short-Term Memory, réseau de neurones artificiel pouvant utiliser des données simples comme des séquences de données tels des vidéos) bidirectionnelle partagée avec des incorporations pré-entraînées.
- Des couches intermédiaires linéaires partagées.
- Une couche finale linéaire pour chaque tâche.

Finetuned Transformer LM:

Un Language Model (LM) est un modèle qui apprend à prédire le prochain mot d'une phrase en fonction de sa connaissance des mots précédents. En faisant cela, le LM apprend à comprendre la langue du corpus d'entraînement.

Son architecture :

- C'est un modèle deep learning comme par exemple RNN.
- Encoder : à partir d'une phrase en entrée, il va produire un vecteur d'activation qui représente sa compréhension du modèle.
- Classifier : il s'agit de l'ensemble des dernières couches d'un LM qui à partir du vecteur d'activations précédent va prédire un mot sous la forme d'un vecteur d'embeddings.

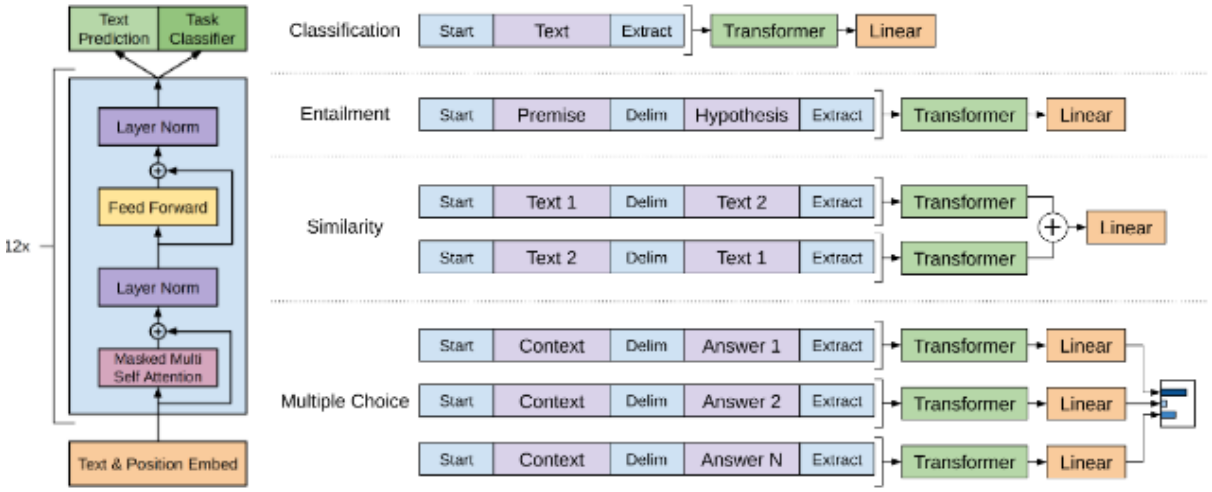


Figure 3: Architecture de Finetuned Transformer LM

A droite de l'image, les transformations d'entrée pour affiner différentes tâches. Le modèle convertit toutes les entrées structurées en séquences de jetons à traiter par le modèle préformé, suivi d'une couche linéaire + softmax.

MT-DNN-ensemble:

MT-DNN s'appuie sur un modèle proposé par Microsoft en 2015 et intègre l'architecture réseau de BERT, un modèle de langage de "transformer" bidirectionnel pré-entraîné proposé par Google l'année dernière.

L'implémentation de MT-DNN est basée sur les implémentations PyTorch de MT-DNN3 et BERT4 .De plus, MT-DNN utilise Adamax comme optimiseur.

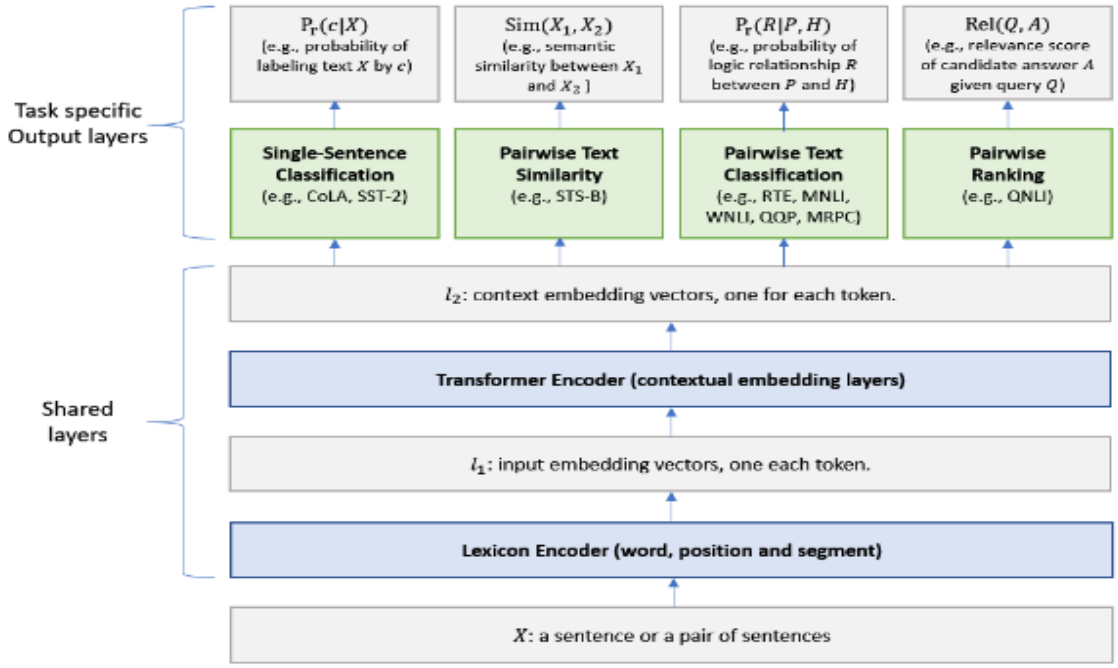


Figure 3: Architecture du modèle MT-DNN-ensemble

Comme le montre la figure ci-dessus, les couches de bas niveau du réseau (c'est-à-dire les couches d'encodage de texte) sont partagées entre toutes les tâches, tandis que les couches supérieures sont spécifiques aux tâches, combinant différents types de tâches NLU (Natural language understanding). Comme le modèle BERT, MT-DNN est formé en deux phases : pré-formation et mise au point. Mais contrairement à BERT, MT-DNN ajoute l'apprentissage multitâche (MTL) dans les phases de réglage fin avec plusieurs couches spécifiques aux tâches dans son architecture de modèle.

L'entrée X, qui est une séquence de mots (soit une phrase soit un ensemble de phrases regroupées) est d'abord représentée comme une séquence de vecteurs d'intégration, un pour chaque mot, dans l1. Ensuite, le codeur transformer capture les informations contextuelles pour chaque mot et génère les vecteurs d'intégration contextuelle partagés dans l2. C'est la représentation sémantique partagée qui est entraînée par nos objectifs multitâches.

Enfin, pour chaque tâche, des couches supplémentaires spécifiques à la tâche génèrent des représentations spécifiques à la tâche, suivies des opérations nécessaires à la classification, à la notation de similarité ou au classement de pertinence.

Hierarchical BiLSTM Max Pooling:

L' architecture proposée suit une approche basée sur l'intégration de phrases pour NLI.

Le modèle propose une hiérarchie de couches BiLSTM (LSTM bidirectionnels) et max pooling qui implémente une stratégie de raffinement itératif et produit des résultats de pointe sur l'ensemble de données SciTail ainsi que des résultats solides pour SNLI et MultiNLI.
Les BiLSTM utilisent deux LSTM pour s'entraîner sur l'entrée séquentielle.

Les cellules LSTM (Long Short Term Memory) possèdent une mémoire interne appelée cellule. La cellule permet de maintenir un état aussi longtemps que nécessaire. Cette cellule consiste en une valeur numérique que le réseau peut piloter en fonction des situations.

L'idée de base derrière ces approches est d'encoder séparément les phrases de prémisse et d'hypothèse, puis de les combiner à l'aide d'un classificateur de réseau neuronal.

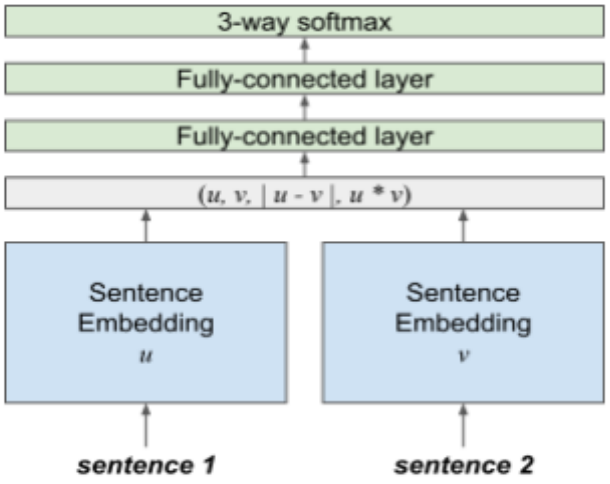


Figure 4: Architecture Hierarchical BILSTM Max Pooling

Le modèle illustré sur l'image ci-dessus contient des incorporations de phrases pour les deux phrases d'entrée, où la sortie des incorporations de phrases est combinée à l'aide d'une heuristique introduite par Mou, en rassemblant la concaténation (u, v), la différence absolue par élément |u - v|, et le produit par élément u * v. Le vecteur combiné est ensuite transmis à un perceptron multicouche à 3 couches (MLP) avec un classificateur softmax à 3 voies.

Les deux premières couches du MLP utilisent à la fois l'abandon et une fonction d'activation ReLU.

XLNet-Large:

XLNet est un modèle autorégressif (AR) généralisé pour la compréhension de langage naturel où chaque mot dépend de tous les mots précédents. Il est généralisé parce qu'il capture le contexte de façon bidirectionnelle au moyen d'un mécanisme de modélisation par permutation de langage ('permutation language modeling' (PLM)) en entraînant un modèle AR sur toutes les permutations possibles de mots dans une phrase tout en conservant l'ordre des mots dans la séquence originale.

Les modèles XLNet de base et large ont les mêmes hyper-paramètres d'architecture que BERT, ce qui donne des modèles de même grandeur.

XLNET est un modèle où le jeton suivant dépend de tous les jetons précédents. XLNET est "généralisé" car il capture le contexte bidirectionnel au moyen d'un mécanisme appelé "modélisation du langage de permutation". Il intègre l'idée de modèles auto-régressifs et de modélisation de contexte bidirectionnelle, tout en surmontant les inconvénients de BERT.

Pour implémenter XLNET, le transformer est modifié pour ne regarder que la représentation cachée des jetons précédant le jeton à prédire.

3. Résultats Obtenus

Les scores d'accuracy obtenus pour les différents modèles proviennent tous de la plateforme GLUE, ces résultats renseignent sur différentes tâches de NLI tel que les précisions de la classification (implication, neutre, contradiction) des différents modèles, pour avoir ainsi les mêmes tests effectués sur chaque modèle, afin d'avoir des résultats comparables. Des résultats sont aussi disponibles sur le site NLP-progress mais nous n'allons pas nous attarder dessus pour le moment car c'est un score obtenu en fonction d'un corpus de texte spécifique. Voici les résultats obtenus sur le site GLUE :

- RoBERTa : 88.1
- MT-DNN-ensemble : 87.6
- Snorkel Metal : 83.2
- Hierarchical BiLSTM Max Pooling : 70.0
- GenSen : 66.1
- BiLSTM + Attn : 65.6
- CAFE : (non classé car trop spécialisé)
- Finetuned Transformer LM : (non classé car trop spécialisé)
- XLNet-Large : (non classé car manque une tâche où il n'y a pas de score sinon moyenne d'environ 85.5)

On peut voir que les modèles RoBERTa, MT-DNN-ensemble et Snorkel Metal obtiennent de bons résultats d'ensemble. Ce sont des modèles potentiellement utilisables.

De plus, le but de notre projet est d'avoir un modèle solide dans les méthodes d'inférence à partir de langage naturel. Notre modèle doit donc obtenir de bons résultats à l'aide du corpus MultiNLI, ce que nous allons voir sur le site NLP-progress.

MultiNLI : Le corpus Multi-Genre Natural Language Inference (MultiNLI) est un corpus à large couverture pour l'inférence en langage naturel, composé de 433 000 paires de phrases écrites par l'homme étiquetées avec implication, contradiction et neutre. Contrairement au corpus SNLI, qui tire la phrase de prémisse des légendes d'images, MultiNLI se compose de paires de phrases de dix genres distincts d'anglais écrit et parlé. L'ensemble de données est divisé en ensembles d'entraînement (392 702 paires), de développement (20 000 paires) et de test (20 000 paires).

Voici donc un classement obtenus à l'aide de ce corpus sur un jeu d'entraînement et d'évaluation :

Model	Matched	Mismatched	Paper / Source	Code
RoBERTa (Liu et al., 2019)	90.8	90.2	RoBERTa: A Robustly Optimized BERT Pretraining Approach	Official
XLNet-Large (ensemble) (Yang et al., 2019)	90.2	89.8	XLNet: Generalized Autoregressive Pretraining for Language Understanding	Official
MT-DNN-ensemble (Liu et al., 2019)	87.9	87.4	Improving Multi-Task Deep Neural Networks via Knowledge Distillation for Natural Language Understanding	Official

Figure 5: Aperçu de résultats pour quelques modèles choisis

Les scores ci-dessus proviennent du site NLP-progress qui est un référentiel pour suivre les progrès du traitement du langage naturel (NLP), y compris les ensembles de données et l'état de l'art actuel pour les tâches NLP les plus courantes.

On peut voir que Roberta obtient encore une fois les meilleurs scores. Nous allons utiliser ce modèle afin de constituer notre application d'analyse de texte, d'annotation du texte et d'identifications et d'élimination de contradictions/redondances.

4. Tests effectués sur le jeu de données Contradictory

Comme vous avez pu le voir ci-dessus, nous avons choisi le modèle Roberta comme modèle de référence pour réaliser notre projet. Pour cela, nous avons décidé de tester les deux modèles suivants : Roberta et Bi-LSTM, qui est un modèle concurrent de Roberta, tous les deux pré-entraînés à la classification de couples de phrases (implication,neutral,contradiction). Au vue de certains problèmes techniques tels que l'utilisation assez hasardeuse de certains modèles ou juste le fait que nous n'avons pas les droits d'accès d'utilisation de certains modèles, nous avons donc décidé de faire une comparaison uniquement entre les deux meilleurs modèles par rapport aux résultats obtenus sur GLUE et NLP-progress. Notre idée de base était d'effectuer le test ci-dessous sur l'ensemble des modèles que nous avons étudiés mais ceci étant impossible selon les raisons ci-dessus.

Nous avons donc créé deux Notebooks, un pour chaque modèle afin que vous puissiez vérifier les résultats si vous le désirez. Vous pouvez retrouver les deux Notebook sur le dépôt GIT dans le dossier "Modèle". Nous avons donc utilisé le jeu de données provenant du corpus SNLI. Le corpus SNLI (Stanford Natural Language Inference) contient environ 550 000 paires hypothèses/prémisses. Les modèles sont évalués en fonction de leur précision. Pour effectuer nos tests, nous avons décidé d'utiliser uniquement les 200 premières lignes du corpus SNLI.

Nous avons donc fait une prédiction à l'aide des deux modèles et obtenu, pour chacun d'eux, une matrice de confusion que vous pourrez retrouver ci-dessous.

1 - Roberta

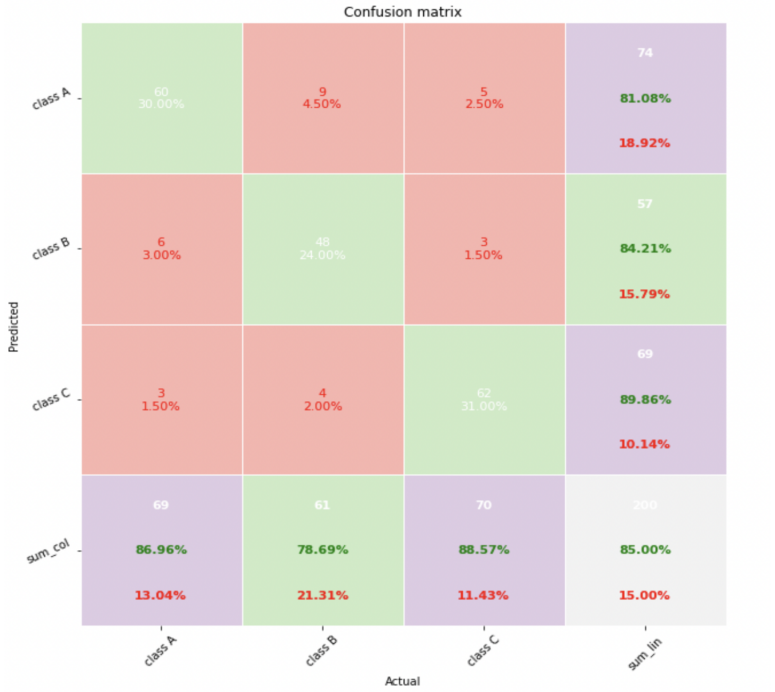


Figure 6: Matrice de confusion de jeu test de modèle Roberta

2 - Bi-LSTM

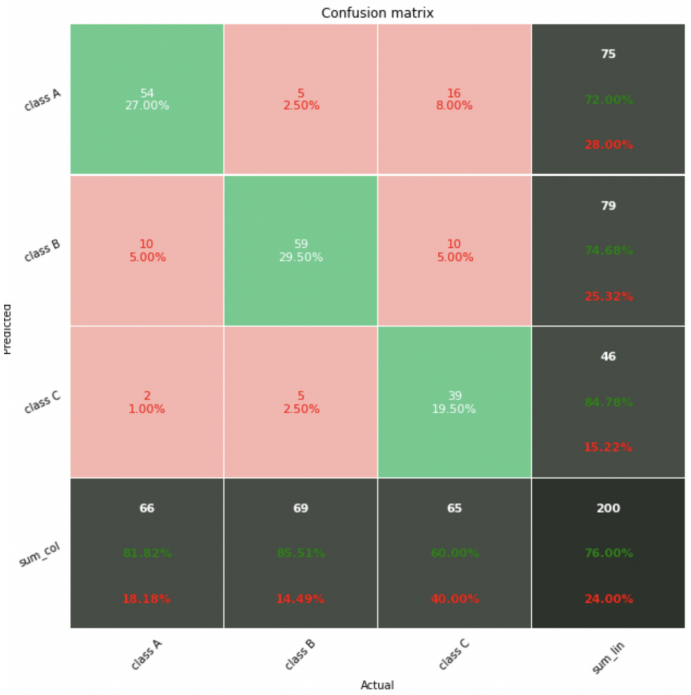


Figure 7: Matrice de confusion de jeu test de modèle Bi-BLSTM

3 - Discussion

Comme vous pouvez le voir sur les matrices de confusions ci-dessus, le modèle Roberta obtient une bien meilleure Accuracy que le modèle Bi-LSTM (85% contre 76%). Cependant, on peut émettre quelques réticences sur la qualité de ces résultats car, comme vous le savez, les modèles ne sont pas entraînés de la même façon. De fait aussi, des différences de jeu de données d'entrainements et de test. Nous allons donc suivre les résultats obtenus sur GLUE, NLP-progress ainsi que le test que nous avons effectué ici, c'est-à-dire garder le modèle Roberta comme modèle référent de notre projet. Le résultat que nous avons obtenu ici ne nous donne aucune contre-indication, au contraire, il nous confirme encore une fois que le modèle Roberta donne de meilleur résultat de classification.

5. Objectifs

Voici les différents objectifs que nous nous sommes fixés au début du projet MASTER après une réunion :

- Créer un logiciel de détection/suppression des contradictions.
- Implémentation de modèles NLI performants dans le logiciel.
- Plusieurs options sont disponibles : Seuil de validation / Taille de fenêtre (1-5) / Language / Détails sur la classification des phrases / Ajout d'une phrase dans le texte pour analyse / Comparaison de 2 phrases.
- Texte/Paragraphe en entrée (15/20 phrases maximum). Exemple: Paragraphe d'un roman ou un article d'un journal.
- Sortir le même texte sauf que noter en rouge les contradictions. On laisse le choix à l'utilisateur pour éventuellement supprimer du texte les contradictions/redondances.
- Bouton qui permet d'ouvrir une fenêtre pour donner des informations supplémentaires à l'utilisateur sur les couples prémisse/hypothèse.
- Bouton "Reset" afin de vider les deux zones de texte ainsi que le dataframe contenant les informations supplémentaires.
- L'utilisateur à le choix de les supprimer (contradictions) ou non via une option (un button).

6. Organisation du travail

a. Méthodologie :

Concernant l'organisation du travail, nous avons utilisé le logiciel de gestion de versions décentralisé GIT afin que tout le monde puisse accéder facilement à la dernière version du projet ainsi que de déposer facilement les modifications que nous avons apportées au projet. Nous avons aussi utilisé la plateforme de communication Discord afin de communiquer au sein de notre groupe. Sur celui-ci, nous avons créé différents "channels" en fonction des sujets qu'on devait évoquer : par exemple une section état de l'art pour avancer sur l'état de l'art, une section choix du modèle pour savoir quel genre de modèle nous allions

intégrer à notre interface graphique ou encore une section "todo-list" pour savoir quelles sont les tâches qu'il faut accomplir afin de faire avancer le projet. Comme nous avons, en moyenne, une réunion par semaine avec les professeurs référents, nous utilisons ce temps aussi pour faire une réunion entre nous afin de voir l'avancée du projet ainsi que de définir les différentes tâches à réaliser pour la semaine prochaine. Grâce à GIT, nous avons aussi mis en place un système de ticket qui se nomme sur le site "issues" afin de voir quelle personne travaillait sur quelle tâche. Grâce au tableau intégré aussi dans l'interface GIT, nous pouvons voir comment chaque personne avançait sur sa tâche. De plus, nous avons aussi fait le choix de tester nos modèles sur des Notebook afin que n'importe quel utilisateur puisse voir les résultats obtenus directement sur GIT sans exécuter toutes les commandes pour effectuer ce test. Grâce à cela, on pouvait juste consulter le dépôt GIT.

b. Qui a fait quoi:

Au lieu d'écrire plusieurs longs paragraphes communs pour savoir ce que chacun a fait lors de l'avancement du projet, on vous propose ici de voir les différentes tâches réalisées par chacun des membres de l'équipe sous forme de paragraphes individuels écrits par chacun. Évidemment, chaque tâche peut être vérifiée sur le GIT dans la section issus où une issue est assignée à la personne qui l'a réalisée ainsi que les différents commits disponibles sur GIT.

En Détail:

SENSENBRENNER Amaury:

Pour ma part, j'ai dans un premier temps effectué différentes recherches, comparé les résultats obtenus et fait un état de l'art sur les différents modèles NLI suivants :

- MT-DNN-ENSEMBLE
- Multi-task BiLSTM + Attn
- Hierarchical BiLSTM Max

Lorsque mes collègues ont tous fini d'écrire leurs rapports sur les autres modèles NLI qu'ils avaient à traiter, j'ai effectué une synthèse de tous les modèles ainsi qu'une comparaison poussée des différents scores obtenus sur les différents sites tels GLUE ou NLP-progress.
C'est donc moi qui me suis occupé de la rédaction complète du document de l'état de l'art.

Dans un second temps, lorsque mes collègues se sont penchés sur l'utilisation d'un modèle pré-entrainé NLI, j'ai schématisé puis construit, à l'aide du langage de programmation Python et de la librairie Tkinter, l'interface finale de notre application. J'ai effectué plusieurs incrémentation sur cette interface pour répondre au mieux au besoin de nos demandes qui évoluent chaque jour afin de mieux répondre à la problématique initiale. J'ai par ailleurs, aussi participé à la "mise en service" du modèle en corrigeant certains problèmes qu'avaient mes collègues au niveau des notebooks déposés sur GIT (notamment pour la labellisation). Je me suis aussi penché sur la question de l'identification des prémisses et des hypothèses qui nous a pris pas mal de temps pour au final conclure que c'est une tâche bien trop complexe. En effet, une phrase peut être à la fois une prémisse et une hypothèse. Nous avons donc décidé d'utiliser une fenêtre coulissante pour notre modèle au lieu d'utiliser un système d'identification de couples prémisse / hypothèse.

De plus, j'ai aussi créé moi même les différentes fonctions de Tokenization, d'extraction de prémisse/hypothèse dans un texte ainsi que des fonctions simples comme par exemple

souligner une phrase en fonction de sa labellisation dans la zone de texte de sortie ou encore le code d'une fenêtre coulissante qui nous permet d'effectuer nos analyses sur des paires de phrases bien définies. J'ai aussi effectué la connexion entre la zone de texte d'entrée, le modèle et la zone de texte de sortie.

Enfin pour finir, je me suis penché sur l'écriture du mémoire dans sa globalité ainsi que la présentation du projet (création diapositive). J'ai écrit chaque catégorie moi-même. j'ai par ailleurs, un peu été la personne qui gérait les issus du GIT en les créant, les faisant avancer dans le tableau des issues et les fermaient lorsque la tâche était accomplie avec succès après vérification.

WEISS Nicolas:

Concernant ma partie, j'ai tout d'abord effectué plusieurs recherches, observé des résultats et fait un état de l'art sur des modèles NLI :
- RoBERTa
- Snorkel MeTaL
- GenSen

Après ces états de l'art et la comparaison des résultats observés de tous les modèles que j'ai recherchés et ceux que les autres ont recherchés, nous avions décidé de créer un modèle RoBERTa qui avait les meilleurs résultats. Suite à cette décision, j'ai commencé la création d'un modèle RoBERTa de zéro. Après de multiples recherches et tentatives dans des notebooks, puis les tentatives d'un autre membre de l'équipe, nous avions observé que notre code et notre modèle demandait des ressources trop importantes et avions donc pris la décision de manière commune de partir sur un modèle pré-entraîné.

J'ai donc ensuite intégré le modèle pré-entraîné dans notre interface et notre code en créant différentes fonctions pour télécharger le modèle, le sauvegarder en local pour éviter de le retélécharger à chaque fois, le charger lorsqu'il était déjà téléchargé et enfin pour l'utiliser sur des prémisses et hypothèses.

Après cela, tout comme les autres membres du groupe, j'ai effectué des recherches concernant l'extraction automatique de prémisses et hypothèses dans un texte, sans succès. Nous avons pris d'autres décisions concernant cela par la suite.

J'ai aussi travaillé sur l'interface, sur son côté responsive. En effet, nous avions quelques problèmes d'affichage comme des éléments qui sortaient de l'écran de certains membres du groupe, notamment liés au système d'exploitation, à la résolution de l'écran ou encore à la taille de la police par défaut du système. Pour contrer cela, j'ai rendu toute notre interface responsive, qui s'adapte à chaque taille d'écran possible et adapte la taille et position des éléments visuels et de la police d'écriture. Cela s'adapte aussi dynamiquement à chaque fois que l'on modifie la taille de la fenêtre.

Après cela, j'ai effectué l'intégration du modèle CamemBERT (modèle pour la langue française) dans notre interface et notre code en ajoutant une option de sélection de langue (entre anglais et français) et en créant une autre fonction de traitement des prémisses et hypothèses par le modèle, en reprenant celle existante pour le modèle RoBERTa que nous avions précédemment créée mais en l'adaptant aux différences de retour du modèle et de ses paramètres.

J'en avais également profité pour corriger certaines erreurs dans l'interface concernant les erreurs d'orthographe ou améliorations du placement de certaines instructions dans le code.

Enfin, pour finir, j'ai travaillé sur l'écriture du mémoire, sur l'écriture d'un certain nombre de textes. J'ai également passé en revue l'intégralité du mémoire pour corriger les erreurs de grammaire ou d'orthographe que je repérais.

TOUAM Juba:

Pour ma part, Comme tout le monde j'ai commencé par effectuer un état de l'art pour des modèles NLI :

- XLNet-Large
- Finetuned Transformer LM
- CAFE

Après cet état de l'art, je me suis intéressé à comprendre comment ces modèles sont implémentés. Comme mentionné par Nicolas ci-dessus, j'ai participé à implémenter un modèle Roberta avec nos propres ressources, cela en respectant l'architecture de modèle roberta, mais sans succès pour des raisons mentionnées précédemment. Pour ces raisons nous avions compris que nous n'avons pas le choix que d'utiliser des modèles pré-entraînés.

Ensuite, j'ai passé beaucoup de temps, pour comprendre quelles sont les données, les architectures des modèles pré-entraînés, afin de s'assurer que ces modèles correspondent à nos attentes, de fait que cette étape est décisive, j'ai participé à créer un test d'un autre modèle roberta pré-entrainé.

Afin d'intégrer le modèle pour la version en anglais, on était dans l'obligation de tokenizer le texte, et d'extraire les premise/hypothesis, afin de résoudre cette problématique, j'ai suggéré une autre façon de tokenizer hors celle proposée par Amaury, j'ai aussi implémenté la première façon d'extraire les premise/hypothesis, cela en constituant un dataframe, pour ce faire, on considérait la première phrase comme premise et la comparait avec toutes les autres phrases considérées comme hypothesis, et ainsi de suite. Mais Amaury a vite repris le travail et a créé une fenêtre glissante.

Au milieu, de l'avancement, j'ai effectué un travail annexe, qui n'est pas utilisé dans ce projet, toutefois, je vais le mentionner, ce travail consiste à créer une modèle pour détecter les premise/hypothesis, en s'appuyant sur le modèle BERT, un rapport détaillant le travail effectué est disponible sur GIT.

Je me suis aussi occupé de trouver un modèle pré-entraîné pour la version française. Comme dans l'état de l'art, nous n'avons pas présenté un modèle traitant de texte français, donc j'ai effectué les recherches nécessaires, afin de trouver le modèle qui nous convenait. Le seul modèle existant est Camembert, qui est une version française du modèle Roberta. J'ai repris ce modèle, je l'ai appliqué sur Le NLI, en le testant avec le jeu de données contradictory en prenant que les premise/hypothesis français. Au passage j'ai testé aussi le modèle Roberta avec le jeu de données contradictory en prenant que les premise/hypothesis en anglais. Deux rapports sont disponibles sur Git, expliquant la démarche prise.

J'ai aussi participé à la rédaction de rapport.

BELQASMI Amine:

Concernant ma partie, j'ai participé à la réalisation des grandes étapes de ce projet**.**

En commençant par l'état de l'art : reconnaître et comprendre les modèles et outils proposés dans ce thème (modèles d'apprentissage automatique NLI, architecture, base de données, etc.)

Ensuite, chercher des modèles d'apprentissage automatique pré-entraînés, les implémenter avec des différentes bases de données afin d'apporter une comparative entre les modèles choisis.

Apprendre à extraire des prémisses et des hypothèses dans un texte, comment les travaux effectués sur ce thème extrait des paires de phrases dans un "texte" d'une manière cohérente (potentiellement une hypothèse et une prémisse)
Participer à la réalisation de l'interface : récupérer le résultat du modèle, réécrire le texte en mettant en rouge les contradictions/redondances, afficher la phrase ou les phrases contradictoires avec une phrase sélectionnée …

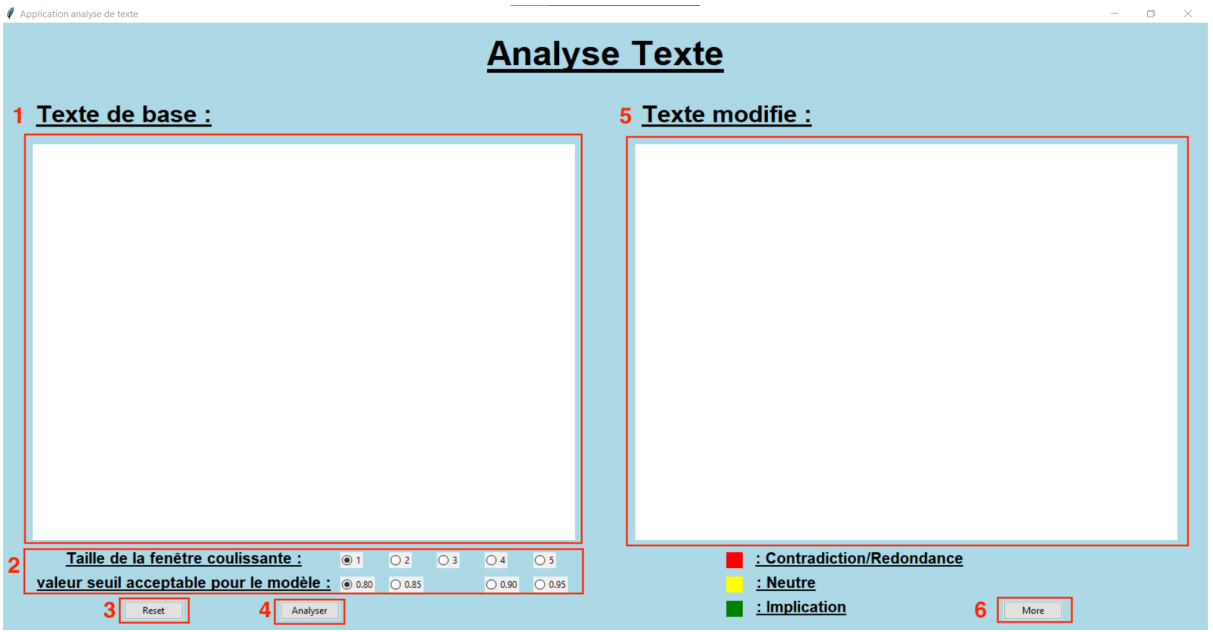
Comprendre et commenter les raisons des limitations de l'application, comment pouvoir améliorer les résultats obtenus …

Réaliser la deuxième version de l'application afin de limiter les incohérences de résultats de la première version.

7. Présentation des outils développés

Comme dit ci-dessus, nous avons développé une interface graphique en Python en utilisant la librairie graphique Tkinter. Nous avons développé deux versions de notre interface pour répondre au mieux à la demande de nos professeurs. Nous allons, dans un premier temps, vous présenter la première interface puis dans un deuxième temps la seconde interface qui possède quelques différences avec la première. Les deux interfaces font des tâches un peu différentes mais l'idée reste la même.

Interface V1 :



Pour chaque numéro, nous allons vous faire un bref résumé de l'utilité de la case associée à celui-ci.

1 - La fenêtre textuelle à gauche de l'écran permet d'écrire ou de copier/coller le texte/paragraphe que l'on veut analyser. Il sert d'entrée à notre modèle par la même occasion.

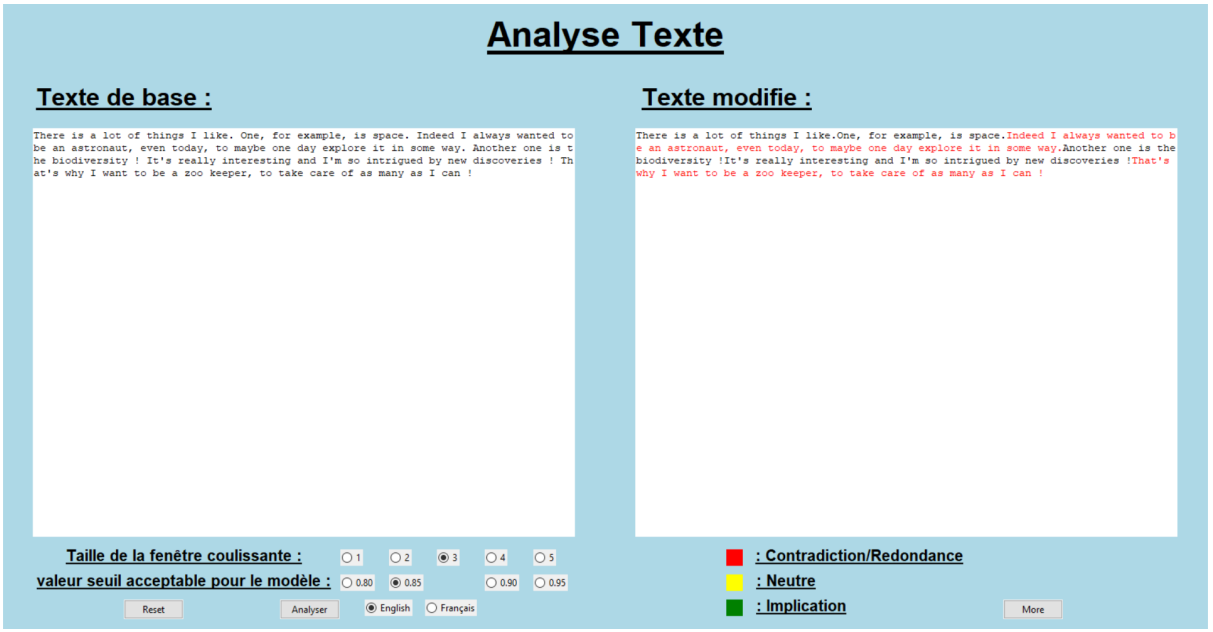
2 - La case numéro 2 permet de définir les options du modèle. Vous pouvez observer deux options : la taille de la fenêtre coulissante ainsi que la valeur du seuil acceptable pour le modèle. Pour la fenêtre coulissante, notre modèle fait des prédictions sur des paires de phrases; il prend une phrase et la couple avec la phrase suivante lorsque l'option de la taille de la fenêtre est à 1 (cela nous fait une comparaison). Lorsqu'on augmente cette option, par exemple à 3, nous allons prendre une phrase et la coupler avec la suivante, puis la coupler avec la suivante de la suivante et encore une fois avec la suivante de la deuxième (cela nous fait 3 comparaison comme la taille de la fenêtre). Plus la taille de la fenêtre est grande, plus

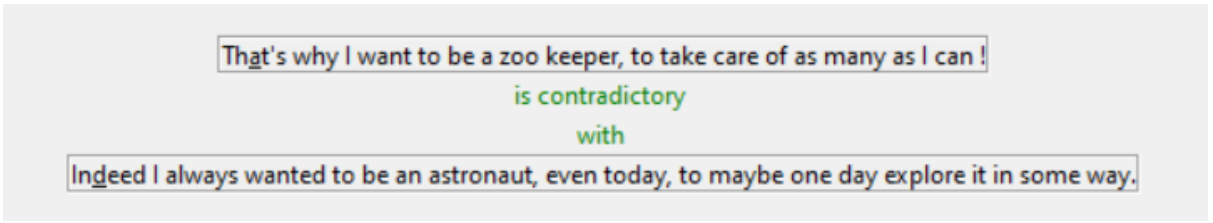
elle nous permet de faire des analyses loins (distance séparant deux phrases) dans le texte. La valeur seuil acceptable pour le modèle est quand est-ce qu'on va dire que cette prédiction est valide à nos yeux ou non. En effet, le modèle nous retourne une prédiction avec une valeur entre 0 et 1 et plus cette valeur est proche de 1, plus la prédiction est forte et risque d'être juste. Grâce à ce seuil, on va pouvoir juger plus durement les prédictions de notre modèle.

3 - Le bouton "reset" permet de réinitialiser le modèle et d'effacer tout le texte présent dans les cases 1 et 5 afin de recommencer une analyse de texte.

4 - Le bouton "analyser" permet de lancer les prédictions de notre modèle. Il va prendre en entrée le texte que nous lui avons fourni dans la case 1 avec les options fournies dans la case 2. On pourra retrouver le résultat dans la case 5 de notre interface.

5 - Comme dit précédemment, la fenêtre textuelle de droite affiche la sortie (ou le résultat) de notre modèle. Nous aurons donc un texte semblable au texte de départ sauf que les phrases considérées comme redondantes ou contradictoires seront écrites en rouge afin de laisser le choix à l'utilisateur de les supprimer ou non. On peut le voir sur la photo ci-dessous (la deuxième photo est une petite fenêtre qui s'ouvre lorsqu'on clique sur une phrase colorée en rouge, elle nous permet de savoir avec quelle(s) autre(s) phrases elle est contradictoire ou redondante) :

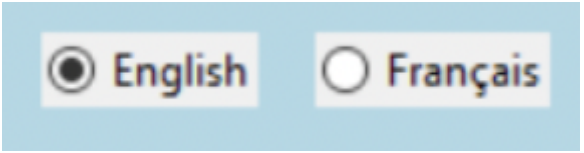




6 - Le bouton "More" permet d'afficher une nouvelle fenêtre qui donnera des détails sur chaque prédiction de paires de phrases que le modèle a fait sous la forme d'un dataframe. On peut voir en détail l'interface graphique de ce bouton ci-dessous :

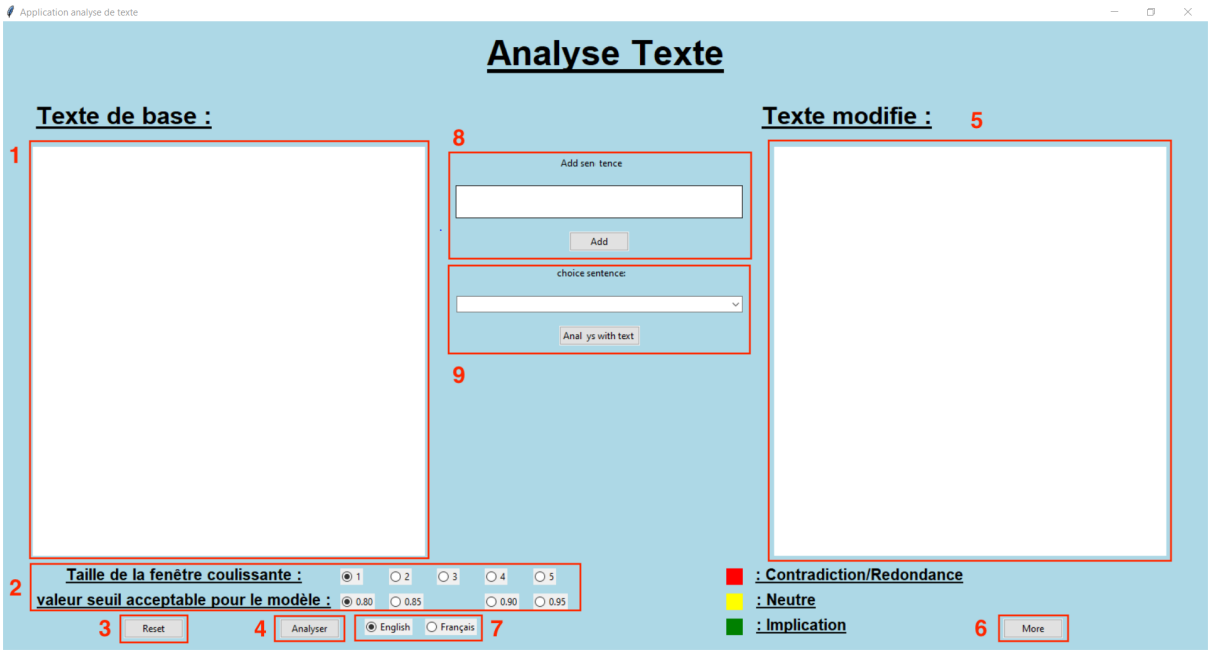
premise	hypothesis
There is a lot of things I like.	One, for example, is space.
There is a lot of things I like.	Indeed I always wanted to be an astronaut, eve n today, to maybe one day explore it in some w ay.
There is a lot of things I like.	Another one is the biodiversity !
One, for example, is space.	Indeed I always wanted to be an astronaut, eve n today, to maybe one day explore it in some w ay.
One, for example, is space.	Another one is the biodiversity !
One, for example, is space.	It's really interesting and I'm so intrigued b y new discoveries !
Indeed I always wanted to be an astronaut, eve n today, to maybe one day explore it in some w ay.	Another one is the biodiversity !
Indeed I always wanted to be an astronaut, eve n today, to maybe one day explore it in some w ay.	It's really interesting and I'm so intrigued b y new discoveries !
Indeed I always wanted to be an astronaut, eve n today, to maybe one day explore it in some w ay.	That's why I want to be a zoo keeper, to take care of as many as I can !
Another one is the biodiversity !	It's really interesting and I'm so intrigued b y new discoveries !

7 - Après quelques ajustements de dernière minute qui ne sont pas disponibles sur la photo ci-dessus, nous avons ajouté une option qui permet au modèle de prendre en compte du texte anglais et du texte français (un modèle différent pour chaque langage). En fonction de la langue du texte d'entrée, l'utilisateur devra soit cocher la case français soit la case anglais. Voir photo ci-dessous ou voir l'interface V2 où cette option est située au même endroit (case 7).



Interface V2 :

La deuxième version de notre interface permet de proposer aux utilisateurs la possibilité d'ajouter des phrases à analyser avec les phrases du texte donné. La différence entre les deux versions est que la deuxième version permet de comparer les phrases entrées par l'utilisateur au centre de l'interface avec les phrases du texte.



Pour chaque numéro, nous allons vous faire un bref résumé de l'utilité de la case.

1 - La fenêtre textuelle à gauche de l'écran permet d'écrire ou de copier/coller le texte/paragraphe que l'on veut analyser. Il sert d'entrée à notre modèle par la même occasion.

2 - La case numéro 2 permet de définir les options du modèle. Vous pouvez observer deux options : la taille de la fenêtre coulissante ainsi que la valeur du seuil acceptable pour le modèle. Pour la fenêtre coulissante, notre modèle fait des prédictions sur des paires de phrases; il prend une phrase et la couple avec la phrase suivante lorsque l'option de la taille de la fenêtre est à 1. Lorsqu'on augmente cette option, par exemple à 3, nous allons prendre une phrase et la coupler avec la suivante, puis la coupler avec la suivante de la suivante et encore une fois avec la suivante de la deuxième. Plus la taille de la fenêtre est grande, plus elle nous permet de faire des analyses loins (distance séparant deux phrases) dans le texte. La valeur seuil acceptable pour le modèle est quand est-ce qu'on va dire que cette prédiction est valide à nos yeux. En effet, le modèle nous retourne une prédiction avec une valeur entre 0 et 1 et plus cette valeur est proche de 1, plus la prédiction est forte et risque d'être juste. Grâce à ce seuil, on va pouvoir juger plus durement les prédictions de notre modèle.

3 - Le bouton "reset" permet de réinitialiser le modèle et d'effacer tout le texte présent dans les cases 1 et 5 afin de recommencer une analyse de texte.

4 - Le bouton "analyser" permet de lancer les prédictions de notre modèle. Il va prendre en entrée le texte que nous lui avons fourni dans la case 1 avec les options fournies dans la case 2. On pourra retrouver le résultat dans la case 5 de notre interface.

5 - Comme dit précédemment, la fenêtre textuelle de droite affiche la sortie (ou le résultat) de notre modèle. Nous aurons donc un texte semblable au texte de départ sauf que les phrases considérées comme redondantes ou contradictoires seront écrites en rouge afin de laisser le choix à l'utilisateur de les supprimer ou non.

6 - Le bouton “More” permet d’afficher une nouvelle fenêtre qui donnera des détails sur chaque prédiction de paires de phrases que le modèle a fait sous la forme d’un dataframe.

7 - Nous avons aussi ajouté une option qui permet au modèle de prendre en compte du texte anglais et du texte français. En fonction de la langue du texte d’entrée, l’utilisateur devrait soit cocher la case français soit la case anglais.

8 - Le groupe de fonctionnalité de la case 8 permet d’écrire une phrase que l’on veut ajouter au texte final à l’aide d’une fenêtre textuelle et d’un bouton qui va envoyer les informations au modèle. La nouvelle phrase va être traitée comme les autres phrases et elle sera surlignée en rouge dans le texte final s' il s’agit d’une contradiction / redondance avec une autre phrase précédente.

9 - Le groupe de fonctionnalité de la case 9 permet de choisir une phrase présente dans le texte final à l’aide d’une liste déroulante afin de comparer chaque autre phrase du texte de sortie avec celle-ci. C’est une autre approche que la fenêtre coulissante que nous avons mise en place dans l’interface numéro 1.

8. Analyse des données résultats

Notre analyse des données résultats se fera en deux parties vu que notre application possède deux modèles pré-entrainés : le modèle Camembert pour le texte en français et le modèle Roberta pour le texte en anglais. De plus, pour effectuer nos tests, nous avons utilisé le jeu de données Contradictory my dear watson, composé de couples prémisse/hypothèse annotés comme suit (0 pour implication, 1 pour neutral, 2 pour contradiction), dans différentes langues. Pour que nous puissions l’utiliser à la fois sur le modèle utilisé pour le français et à la fois sur le modèle utilisé pour l'anglais, car comme mentionné auparavant, c’est un jeu de données existant en plusieurs langues. Rappelons que dans les deux cas, nos modèles ne se sont pas entraînés avec ce jeu de données, le but ici c’est de tester les deux modèles avec des données qui leurs seront nouvelles, pour s’assurer de la fiabilité de nos modèles.

1) Modèle Camembert

Comme nous n'avons pas introduit ce modèle dans notre état de l'art, nous allons le faire ci-dessous, avant de présenter les résultats que nous avons obtenus.

a. Camembert - Introduction

Camembert est une version francophone de modèle BERT, pré-entraîné sur 138 GB de texte français. Ce modèle a été rendu public par les équipes de Facebook AI Research associées aux chercheurs de L’INRIA.

Le fait qu’il soit entraîné sur un grand corpus de texte représente un avantage pour résoudre les problématiques liées à la classification de texte et traduction de texte car le ré-entraîner coûterait cher en ressources. Camembert est testé et évalué sur différentes tâches pour du traitement de texte en français, la reconnaissance d'entités nommées (NER) ou l'inférence en langage naturel NLI.

b. Camembert - Architecture

Camembert, comme Roberta et Bert, est un transformateur multicouche bidirectionnel. Camembert utilise l'architecture de base de Bert, très similaire à Roberta. La différence principale étant l'utilisation du masquage des mots entiers (whole-words masking en anglais) et l'utilisation de la tokenization sentencePiece.

Le **masquage des mots entiers** consiste à masquer tous les sous-mots correspondant à un mot en une seule fois dans une séquence (token, phrase, paragraphe). Ceci est fait parce qu'ils veulent pré-entraîner un modèle bidirectionnel. La plupart du temps, le réseau verra une phrase avec un jeton [MASK], et il sera entraîné à prédire le mot qui est censé être là.

La tokenisation **sentencePiece** est un analyse lexicale (tokenizer et detokenizer) de texte non supervisé utilisé principalement pour les systèmes de génération de texte basés sur un réseau neuronal (ANN) où la taille du vocabulaire est prédéterminée avant l'entraînement du modèle neuronal. Il entraîne des modèles à partir des modèles de tokenisation et détokenisation à partir de phrases. Il traite ainsi ces phrases comme des séquences de caractère unicode. Cette méthode est rapide et légère : la vitesse de segmentation est d'environ 50 000 phrases/secondes et l'empreinte mémoire d'environ 6 Mo.

Nous n'avons pas présenté l'architecture de Camembert plus en détail ici car elle est fortement similaire à celle de Roberta qui est déjà présentée dans l'état de l'art. Les seules différences sont le masquage des mots entiers et la tokenization sentencePiece.

c. Camembert - Modèle de détection NLI pré-entrainé

Pour une version en français de notre API, nous avons utilisé un modèle pré-entraîné issu du site HuggingFace qui a été entraîné sur les données XNLI. XNLI est un jeu de données composé de prémisses et d'hypothèses écrites et parlées. Ce corpus est composé de 14 langues qui, parmi elles, le français. D'ailleurs, c'est le seul jeu de données NLI conséquent existant en langue française.

Nous avons fait le choix d'utiliser un modèle pré-entraîné à cause du manque de ressources locales et des performances GPU. Nous étions dans l'impossibilité de reproduire l'entraînement du modèle Camembert avec le jeu de données XNLI (et de Roberta par la même occasion) sur le jeu de données XNLI. Toutefois pour s'assurer des performances de classification enregistrées dans ce dernier, nous l'avons testé sur le jeu de données Contradictory my dear watson disponible sur le site Kaggle comme dit précédemment.

d. Camembert - Tests et résultats du modèle

Dans un premier temps, le modèle pré-entraîné Camembert obtient les "Accuracys" suivantes pour les données de validation et de test. Ces résultats sont disponibles directement sur la page du modèle HuggingFace.

Set	Accuracy
validation	81.4
test	81.7

Figure 8 :Accuracy de data test,validation de modèle pré-entraîné

Dans un second temps, nous avons effectué nos tests sur le jeu de données "Contradictory my dear watson" comme précisé précédemment. Pour évaluer le modèle pré-entraîné, nous avons utilisé les données d'entraînement, de fait que nous n'avons pas suffisamment de données de test. Comme ces données n'étaient pas utilisées pour entraîner le modèle camembert, qui est entraîné par XNLI français, cela nous permet de vérifier la fiabilité de ce modèle, car ce sont des données différentes. La quantité des données de test (pour chaque label : "entailment", "neutral" et "contradiction") sont les suivantes :

label	premise	hypothesis
0	133	133
1	129	129
2	128	128

Figure 9: Quantité Données test

À l'aide de ce jeu de données de test, nous avons obtenu une accuracy de 83,3%. Ce qui est en soi plutôt correct pour une quantité de données de test aussi faible.

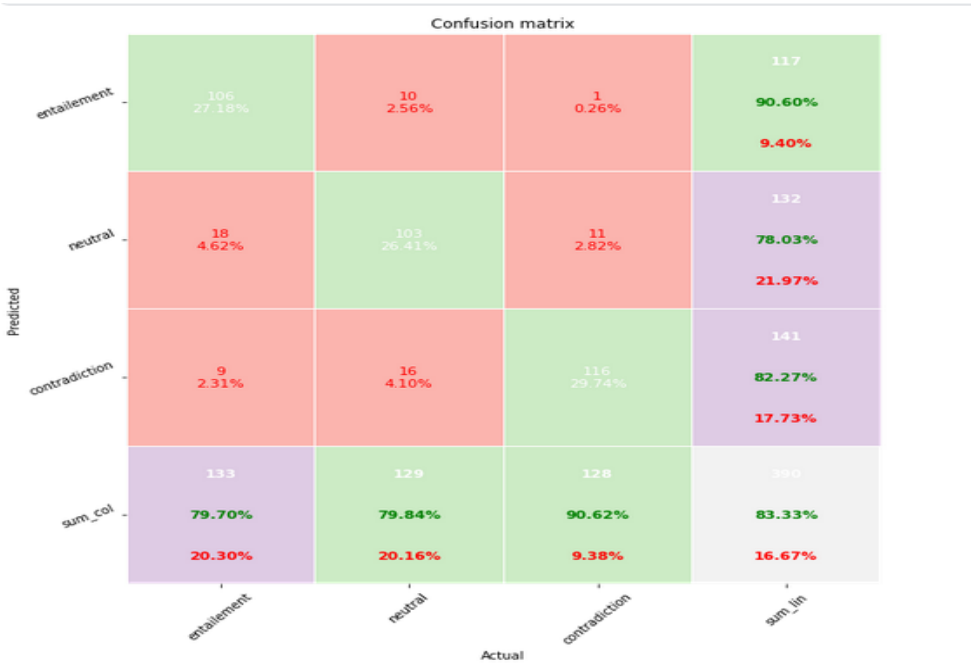


Figure 10: Matrice de confusion Camembert

Nous observons ci-dessus la matrice de confusion du modèle Camembert. En effet ce modèle n'est pas parfait mais possède des résultats satisfaisants, n'ayant que 16,67% des couples de prémisse-hypothèse mal classés. Il est impossible d'avoir un modèle classant de manière totalement parfaite tout couple de prémisse-hypothèse que l'on lui passe, il y aura toujours des exceptions.
Cela peut-être dû aux données d'entraînement, en effet nous n'avions pas la main sur les données utilisées dans l'entraînement du modèle, ce dernier étant un modèle pré-entraîné. Et il est également impossible d'avoir toutes les situations possibles dans ce jeu de données, donc impossible de tout couvrir.
Au-delà de cela, il est aussi difficile d'avoir une analyse lexicale et sémantique parfaite. Tout en prenant en compte qu'il faudrait que notre modèle garde en mémoire certains éléments notamment liés aux emplacements géographiques ou temporels ou liés aux personnes concernées par la phrase. Cela serait utile pour éviter une mauvaise classification de couples de phrases qui indépendamment ont l'air contradictoires mais qui en fait ne le sont pas si l'on sait que l'une est dans le passé et l'autre dans le futur ou si elles concernent des personnes ou personnages différents.

Nous pouvons donc valider la fiabilité de notre modèle d'analyse de texte en français : Camembert.

2) Modèle Roberta

a) Roberta - Introduction

Nous allons juste faire un rappel ici comme quoi nous avons présenté en détail, plus haut dans le rapport, toute l'architecture du modèle Roberta ainsi que différents résultats obtenus avec ce modèle dans l'état de l'art disponible plus haut.

b) Roberta - Modèle de détection NLI pré-entraîné

Pour avoir une version en anglais de notre API, nous avons utilisé un modèle pré-entraîné issu de la librairie AllenNLP qui est entraîné sur les données SNLI. C'est un ensemble de données uniquement en anglais, représentant les légendes d'images, où ces dernières sont utilisées comme prémisses. Cependant, les hypothèses sont créées manuellement par les travailleurs de Mechanical Turk.

c) Roberta - Tests et résultats du modèle

Dans un premier temps, le modèle pré-entraîné Roberta obtient une accuracy de 94% pour les données de tests SNLI (une partie du jeu de données qui a aussi été utilisé pour entraîner le modèle). On peut émettre une petite réserve sur cette accuracy à cause de la taille de l'échantillon de tests qui est assez petit.

gold_label	sentence1	sentence2
contradiction	100	100
entailment	104	104
neutral	96	96

Figure 11: Quantité de données test

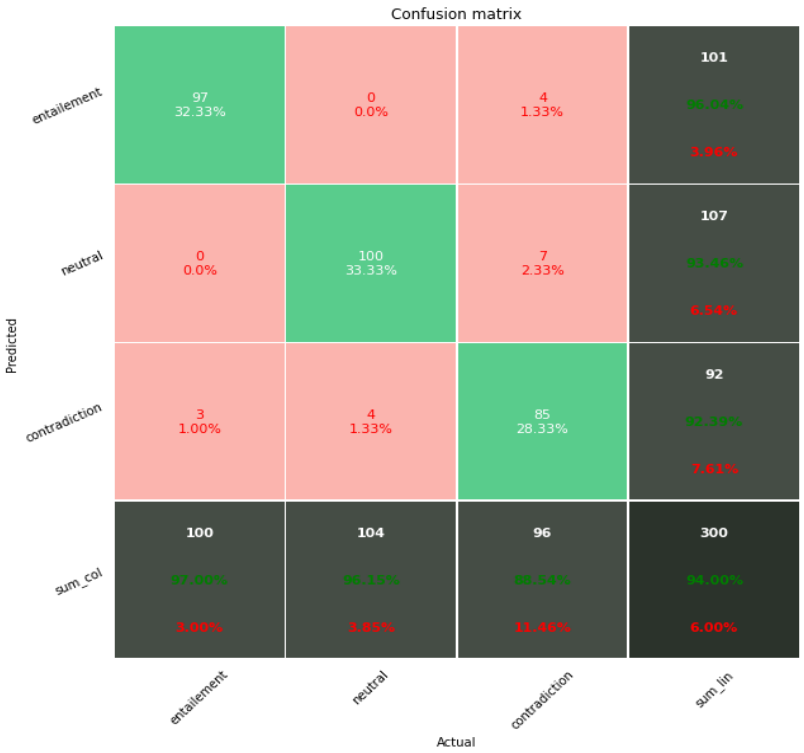


Figure 12: Matrice de confusion Roberta données SNLI

Dans un second temps, nous avons effectué une deuxième phase de test à l'aide de jeu de données Contradictory my dear watson disponible sur Kaggle comme précisé plus haut dans le rapport. On a décidé d'utiliser ces données parce que ces données seront nouvelles pour le modèle pré-entrainé avec SNLI, cela nous permettra de tester vraiment la fiabilité de modèles et les performances enregistrées par ce modèle. Dans ce rapport de test, notre modèle Roberta obtient une accuracy de 83,8%.

gold_label	premise	hypothesis
0	172	172
1	165	165
2	163	163

Figure 13: Quantité de données test contradictory

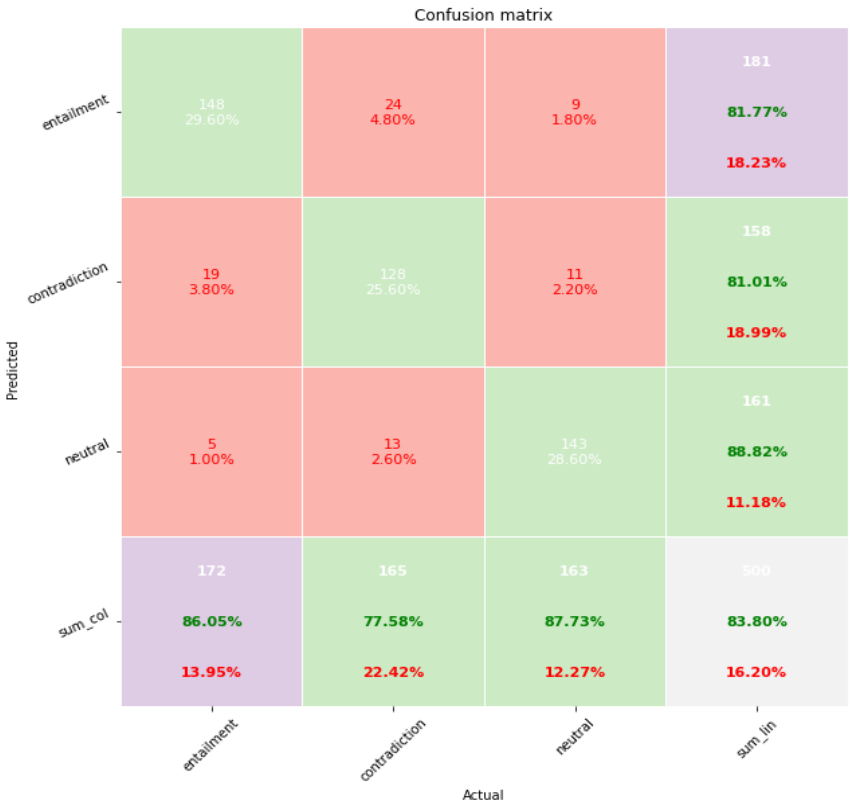


Figure 14: Matrice de confusion Roberta données Contradictory

Nous remarquons une différence dans les performances de modèle sur les deux tests que nous avons effectués. Tel que nous avons enregistré une baisse de performance de prédiction sur les données test contradictory, cela s'explique du faite que ces données sont nouvelles pour le modèles pré-entraîné, ceux sont des données différentes de SNLI. Rappelons que SNLI sont des données constituées à partir des légendes d'images, où ces dernières sont utilisées comme prémisses, les hypothèses sont créées manuellement par les travailleurs de Mechanical Turk. De là, nous constatons que l'issue (la manière dont les données sont constituées comme par

exemple que les données SNLI proviennent de légendes d'images, mais qu'on ne sache pas d'où provient exactement le jeu de données "Contradictory my dear watson") des données influence sur la qualité du modèle.

Tout comme le modèle Camembert, ce modèle n'est pas parfait mais possède des résultats très satisfaisants. De même pour celui-ci, il est impossible d'avoir un modèle classant de manière totalement parfaite tout couple de prémisse-hypothèse, il est impossible de couvrir toutes situations possibles dans un jeu de données qui lui servirait de jeu d'entraînement.
Il est également difficile d'avoir une analyse lexicale et sémantique parfaite : notre modèle devrait dans l'idéal garder en mémoire certains éléments liés aux emplacements géographiques ou temporels ou liés aux personnes concernées par la phrase.
Avoir ces éléments en mémoire permettrait une classification encore meilleure mais cela n'est pas pris en compte par le modèle.

Le modèle Roberta le même score de classification que notre modèle français mais nous ne pouvons pas les comparer car le jeu de test n'est pas le même malgré que la taille des deux jeux de test soit presque identique.

Malgré cela et comme pour le modèle Camembert, nous pouvons donc valider la fiabilité de notre modèle d'analyse de texte en anglais : Roberta.

3) Évaluation et justification des choix grâce à des tests spécifiques

- Modèle pré-entraîné car nous n'avons pas les ressources suffisantes afin de créer et développer nous-même un modèle NLI car très coûteux (fonds non débloqué par l'université).

- Impossibilité d'identifier dans un texte une prémisse et une hypothèse donc nous avions effectué nos algorithmes sur des couples pré-définis (fenêtre coulissante par exemple).

- Choix des deux modèles d'analyse de textes imbriqués dans notre interface pertinent car ils ont, tous les deux, obtenu de bons résultats. Ce qui atteste de leur fiabilité. Le modèle Camembert a été choisi pour la version française car c'est celui qui se rapproche le plus du modèle Roberta. Il n'a juste pas été entraîné sur le même jeu de données.

9. Perspective / conclusion

L'objectif de ce projet consistait à la recherche d'incohérences dans un texte. À la suite du travail réalisé, nous pouvons avancer que notre modèle de détection d'incohérence est fonctionnel. Malgré que le modèle détecte des contradictions qui ne doivent pas être détectées, principalement à cause des précisions de prédictions qui s'élèvent à 93% et 83% respectivement pour les modèles pré-entrainés Roberta et Camembert, représentant des taux d'erreur de prédictions considérables. De plus, la précision du modèle Roberta a diminué lorsqu'on l'a testé sur un nouveau jeu de données, ce qui explique que la provenance et le

type du jeu de données influencent les performances de détection. C'est pour ça que nous enregistrons des résultats qui varient selon le contexte. Nos modèles ne gardent pas en mémoire les informations de lieux, de temps ou de personnes concernées dans une phrase ou un groupe de phrases. Par ailleurs, plus la quantité de texte traitée est grande, plus nous enregistrons des incohérences qui ne doivent pas être détectées, ce qui nous a poussé à ne traiter qu'une quantité de texte limitée.

Toujours afin d'obtenir de meilleurs résultats de détection d'incohérence, certaines voies d'amélioration de constitution de nos modèles semblent prometteuses. A titre d'exemple, en entraînant le modèle avec seulement un type de données précis, on s'assurera de la pertinence du modèle au moment de son application, par exemple si on l'a entraîné avec des données issues d'un contexte politique, on sera certain que la qualité du modèle dégradera en dehors de ce contexte.

Les performances et résultats du modèle ne sont pas les seuls points d'améliorations possibles, en effet les options ou paramètres que nous avons sur notre interface sont limités. Par exemple, nous proposons actuellement sur notre interface un choix entre le français et l'anglais, ce qui sélectionne le modèle associé. Nous pourrions imaginer ajouter des modèles pour différents autres langages afin de couvrir plus de possibilités quant à la langue du texte d'origine. La langue pourrait ne pas être la seule option que l'on retoucherait, en effet nous pourrions permettre à l'utilisateur de rentrer directement dans un champ de saisie la valeur seuil acceptable pour le modèle ainsi que la taille de la fenêtre coulissante. Ce qui laisserait plus de liberté quant à l'analyse du texte d'origine.

Maintenant, en dehors du modèle ou des options de notre interface, d'autres perspectives s'offrent à nous, notamment en ce qui concerne la détection de prémisses et d'hypothèses. Elle qui constituait tout de même une grande partie du sujet, nous avons décidé de trouver une alternative à un algorithme de détection. En effet, leur détection est à ce jour encore un problème ouvert pour lequel nous n'avons pas de solution vraiment efficace. Il y a beaucoup trop de facteurs qui entrent en jeu afin de savoir si une phrase est une prémisse ou une hypothèse d'une autre.