Dauphine | PSL
UNIVERSITÉ PARIS

Rapport projet NLP

Madame PAPA Anna

MedAssist : Prédictions de diagnostics médicaux à partir de symptômes dans un texte

Projet réalisé par NDIAYE Maïrame et BRIHMOUCHE Yacine

Master 2 Intelligence Artificielle, Systèmes et Données

22 mai 2023

Table des matières

1 Présentation du projet

Dans ce rapport, nous présentons le travail accompli dans le cadre de notre projet de NLP. L'objectif principal de ce projet était d'appliquer différentes tâches de NLP à un sujet de notre choix.

Dans cette optique, nous vous proposons **MedAssist**, un outil de détection de diagnostics à partir de symptômes présents dans un texte en français. MedAssist est conçu comme un assistant médical capable d'extraire les symptômes d'un texte et de prédire le diagnostic correspondant à ces symptômes.

La figure 1 illustre le fonctionnement global de MedAssist, mettant en évidence les étapes clés telles que l'extraction des symptômes et la prédiction du diagnostic.

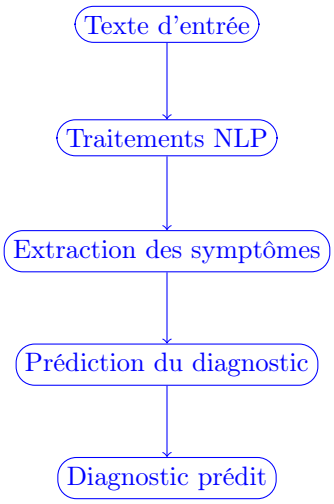

FIGURE 1 – Fonctionnement global de MedAssist

Exemples d'utilisation de MedAssist

- Médecin assistant : MedAssist peut être utilisé comme un outil d'assistance pour les médecins, leur permettant d'obtenir rapidement une liste de diagnostics possibles et de s'assurer de ne pas en oublier un.
- Patient en situation d'urgence : MedAssist peut être directement utilisé par un patient qui se trouve dans une situation d'urgence et n'a pas la possibilité de consulter un médecin immédiatement. Cela lui permettrait d'obtenir rapidement un diagnostic en attendant une prise en charge médicale adéquate.

Soulignons que MedAssist est conçu comme un assistant médical et ne remplace en aucun cas l'avis et l'expertise d'un professionnel de la santé. Nous vous présentons dans la suite les différentes étapes du projet ainsi que notre approche pour accomplir cette tâche.

1.1 Prise en main du projet

Pour lancer l'outil vous devez suivre les étapes suivantes :

1. Ouvrir votre terminal
2. Aller dans le répertoire où ce trouve le dossier du projet
3. Exécuter la ligne de code suivante : python "Projet_NDIAYE_BRIHMOUCHE.py" et patienter quelques secondes avant l'affichage de l'interface graphique.

2 Construction du corpus : Le Web Scrapping

Nous pouvons identifier trois grandes étapes dans la réalisation de ce projet. La première étape consiste à construire le corpus en utilisant le web scraping. Le web scraping est le processus d'extraction automatique de données à partir de sites web.

Pour notre tâche, nous nous sommes intéressés aux sites liés au domaine de la santé où nous pouvions trouver à la fois des descriptions de symptômes et les diagnostics médicaux correspondants. Nous avons examiné plusieurs sites potentiels tels que VIDAL [1], Hopital [2], Santé magazine [3], etc.

Finalement, nous avons choisi de nous concentrer sur VIDAL qui proposait des fiches détaillées et complètes sur différents maux et maladies.

2.1 Un peu de HTML

Une fois le site à scraper sélectionné, nous devions déterminer les informations à extraire. Pour cela, nous avons analysé la structure HTML des différentes pages afin de savoir comment récupérer les diagnostics et les descriptions des symptômes.

Voici ce que nous avons identifié :

- Tous les noms de diagnostics peuvent être extraits de la page principale en parcourant les balises <ul> de la classe "articles-list" et en récupérant le texte de la balise <a> à l'intérieur.
- En ce qui concerne les descriptions des symptômes, il faut accéder à une page secondaire spécifique à chaque diagnostic, accessible via 'https ://www.vidal.fr/'+**str** où **str** correspond à l'attribut href fourni par la balise <a> contenant le nom du diagnostic. Ensuite, plusieurs cas de figure peuvent se présenter :
    1. La description des symptômes est directement présente sur la page secondaire et peut être accédée soit via la balise de la classe "vlxtexte" avec l'attribut data-vlxtype="symptomes", soit via la balise <h2> contenant le mot "symptômes" dans son texte, et dans ce cas, la description est contenue dans les balises <p> et <ul> qui suivent.
    2. Sinon, les symptômes se trouvent sur une autre page indiquée par les balises <a>. S'il existe une balise <a> contenant le mot "Symptômes" ou "Diagnostic" dans son texte, il faut utiliser le lien href pour accéder à la nouvelle page et le cas 1. s'applique à nouveau. Sinon, il n'y a pas de description de symptômes.

Une fois l'analyse de l'architecture HTML terminée, nous avons écrit un code Python pour extraire les descriptions des symptômes et les diagnostics en suivant l'architecture décrite précédemment. Pour réduire le risque de blocage par le site, nous avons mis un repos de quelques secondes à chaque accès de page.

Pour effectuer le scraping, nous avons utilisé Beautiful Soup, une bibliothèque Python utilisée pour l'analyse et l'extraction de données à partir de fichiers HTML et XML. Beautiful Soup facilite le web scrapping en permettant la recherche et l'extraction d'informations à partir de la structure d'un document de manière simple et intuitive. D'autres outils de scrapping existent en python tels que Scrapy, PyQuery etc...

Finalement, nous avions à notre disposition un corpus contenant une description de symptômes pour près de 212 maux et maladies. Le tableau ci-dessous représente une ligne du CSV du corpus.

1. Lien du site VIDAL
2. Lien du site Hopital
3. Lien du site Santé magazine

texte_symptomes	diagnostic
"La bronchite aiguë commence souvent avec une petite toux sèche. Ensuite apparaissent des glaires, d'abord limpides ou blanchâtres. En cas de surinfection bactérienne, les sécrétions deviennent jaunâtres et purulentes. Une bronchite aiguë s'accompagne souvent de fièvre, maux de tête, courbatures, rhume, sinusite et maux de gorge. Elle peut aussi être associée à une trachéite, une inflammation infectieuse de la muqueuse qui tapisse la trachée. La bronchite aiguë disparaît normalement en quelques jours."	Bronchite

3 Construction du dataset : L'annotation

La deuxième étape du projet a consisté en l'annotation des données textuelles en leur attribuant des informations spécifiques. Pendant cette étape, il s'agissait d'identifier et de marquer les parties pertinentes du texte en leur attribuant des balises, afin de constituer un ensemble de données d'entraînement. Différentes approches sont possibles pour réaliser cette tâche :

- **Annotation manuelle :** L'ensemble du processus d'annotation est effectué manuellement par des annotateurs humains, sans aucune assistance automatisée. Chaque exemple de texte est examiné par les annotateurs, qui attribuent les balises ou les étiquettes appropriées. Des outils tels que Brat, Labelbox, Prodigy, etc., peuvent être utilisés pour faciliter l'annotation manuelle. Cette approche permet de produire des jeux de données de haute qualité grâce à l'expertise humaine, mais elle peut être chronophage en raison du grand nombre d'exemples à annoter.
- **Annotation semi-automatique :** Cette approche combine l'expertise humaine avec des techniques d'automatisation pour faciliter l'étiquetage des données textuelles. L'objectif est de réduire la charge de travail manuel tout en maintenant une qualité élevée des annotations. Différentes techniques peuvent être utilisées, telles que l'utilisation d'expressions régulières pour repérer des motifs spécifiques, ou l'utilisation de modèles pré-entraînés pour détecter des patterns. Chacune de ces techniques présente ses avantages et ses inconvénients, et leur choix dépend souvent du contexte et du sujet du projet.

Dans notre projet, notre objectif était de baliser les différents symptômes présents dans les textes afin de construire un ensemble de données annotées. Nous nous sommes concentrés sur l'annotation semi-automatique en explorant deux approches différentes : l'utilisation d'expressions régulières et l'application du transfert d'apprentissage sur un modèle pré-entraîné.

3.1 Les expressions régulières pour l'annotation

Nous avons suivi plusieurs étapes pour préparer les données. Tout d'abord, nous avons appliqué une normalisation en mettant tout le texte en minuscules et en supprimant certains caractères spéciaux. Ensuite, nous avons construit un dictionnaire de symptômes qui comprend une liste exhaustive de symptômes courants et de leurs synonymes. Pour enrichir ce dictionnaire, nous avons utilisé ChatGPT, qui nous a fourni un certain nombre de symptômes courants, puis nous avons complété cette liste en ajoutant d'autres symptômes et synonymes. Au final, nous avons obtenu un dictionnaire comprenant 118 symptômes courants avec leurs descriptions et synonymes associés.

Une fois le dictionnaire construit, nous avons développé la fonction *annotation_symptomes* présentée ci-dessous. Cette fonction prend en entrée un texte et utilise des expressions régulières pour rechercher les symptômes répertoriés dans le dictionnaire. Elle place ensuite des balises <symptome> et <\symptome> autour des symptômes identifiés :

```
def annotation_symptomes(text):
    for keyword, symptom_list in symptomes_dict.items():
        for symptom in symptom_list:
```

```
            pattern = r'(?<!<symptome>)\b' + re.escape(symptom) + r'\w*\b(?!<\/
                symptome>)'
            text = re.sub(pattern, '<symptome>' + keyword + '</symptome>', text
                , flags=re.IGNORECASE)
    return text
```

Finalement, nous avions à notre disposition un ensemble de données annoté pour toutes les descriptions de symptômes. Le tableau ci-dessous représente une ligne du CSV d'annotation.

texte_annote	diagnostic
la bronchite aiguë commence souvent avec une petite <symptome>toux<\symptome> sèche. ensuite apparaissent des glaires, d'abord limpides ou blanchâtres. en cas de surinfection bactérienne, les sécrétions deviennent jaunâtres et purulentes. une bronchite aiguë s'accompagne souvent de <symptome>fièvre<\symptome>, <symptome>fièvre<\symptome>, <symptome>douleur musculaire<\symptome>, rhume, sinusite et <symptome>maux de gorge<\symptome>. elle peut aussi être associée à une trachéite, une <symptome>sensation de brûlure<\symptome> infectieuse de la muqueuse qui tapisse la trachée. la bronchite aiguë disparaît normalement en quelques jours.	Bronchite

Cette approche présente plusieurs avantages, notamment sa rapidité d'implémentation et sa flexibilité pour les modèles de symptômes. En utilisant des expressions régulières, nous avons pu construire une liste conséquente de symptômes couramment rencontrés dans les textes. Cependant, nous reconnaissons que cette liste, bien qu'abondante, demeure incomplète étant donné la multitude de symptômes existants et de leurs synonymes potentiels.

Afin de surmonter cette limite, nous avons exploré l'utilisation de l'apprentissage par transfert. Notre objectif était d'exploiter un modèle pré-entraîné en français qui serait en mesure de repérer les symptômes dans un texte. En adoptant cette approche, nous espérions améliorer la capacité de détection des symptômes, notamment en gérant les variations linguistiques et les synonymes plus efficacement.

L'intégration de l'apprentissage par transfert dans notre processus d'annotation nous permettrait de bénéficier des connaissances préalables du modèle et potentiellement d'obtenir de meilleurs résultats dans l'identification des symptômes.

3.2 Le transfert d'apprentissage pour la détection de symptômes

Le **transfer learning** est une approche d'apprentissage automatique qui consiste à utiliser les connaissances acquises par un modèle pré-entraîné sur une tâche spécifique et à les transférer à une autre tâche similaire. Dans notre projet, nous avons exploré l'existence de modèles pré-entraînés spécifiquement conçus pour la détection de symptômes. Nous avons identifié BioBERT, ClinicalBERT et SciBERT qui sont adaptés aux domaines de la santé et des sciences. Cependant, ces modèles sont entraînés en anglais, alors que nos données sont en français.

Une approche possible aurait été de traduire nos données en anglais, d'utiliser l'un de ces modèles pour détecter les symptômes, puis de reconvertir les résultats en français. Cependant, cela ne nous aurait pas permis de traiter directement la tâche de transfert d'apprentissage dans sa langue d'origine.

Pour cette raison, nous nous sommes tournés vers des modèles plus généraux qui ont été entraînés en français. Parmi ceux-ci, nous avons considéré des options telles que CamemBERT, FlauBERT et RobBERT. Après évaluation, nous avons choisi de poursuivre notre tâche en utilisant le modèle CamemBERT, qui a été pré-entraîné sur un large corpus de texte en français.

Pour réaliser cette tâche nous nous sommes tournés vers les transformers d'*Hugging Face* pour le chargement du modèle pré entrainé :

```
SYMPTOME_LABEL = "SYMPTOME"
NON_SYMPTOME_LABEL = "NON_SYMPTOME"
LABEL_TO_ID = {NON_SYMPTOME_LABEL: 0, SYMPTOME_LABEL: 1}

model_name = 'camembert-base'
model = CamembertForTokenClassification.from_pretrained(model_name, num_labels=
    len(LABEL_TO_ID))
tokenizer = CamembertTokenizer.from_pretrained(model_name)
```

Nous avons ensuite chargé le fichier CSV annoté et avons fait un pré traitement des données à travers la tokenisation en mots et labelisation des données :

```
tokenized_texts = []
labels = []
for row in data:
    row = re.sub(r"\.|,|:|;", "", row)
    row = re.sub(r"(\S)'(.)", r"\2", row)
    symptoms = re.findall(r"<symptome>(.*?)<\/symptome>", row)
    if symptoms:
        tokens = re.findall(r"<symptome>(.*?)<\/symptome>|(\S+)", row)
        filtered_tokens = [token[0] if token[0] else token[1] for token in
            tokens if token[0] not in mots_courants and token[1] not in
            mots_courants]
        tokenized_texts.append(filtered_tokens)
        labels.append([SYMPTOME_LABEL if token[0] else NON_SYMPTOME_LABEL for
            token in tokens])
    else:
        tokens = row.split()
        filtered_tokens = [token for token in tokens if token not in
            mots_courants]
        tokenized_texts.append(filtered_tokens)
        labels.append([NON_SYMPTOME_LABEL] * len(tokens))
```

Nous avons par la suite converti les données en tenseurs PyTorch pour qu'elles soient dans le format accepté par CamemBERT et avons créé le dataset PyTorch :

```
input_ids = []
attention_masks = []
label_ids = []

for tokens, label in zip(tokenized_texts, padded_labels):
    encoded_input = tokenizer.encode_plus(tokens, padding='max_length',
        truncation=True, max_length=max_label_length , return_tensors='pt')
    input_ids.append(encoded_input['input_ids'].squeeze())
    attention_masks.append(encoded_input['attention_mask'].squeeze())
    label_ids.append(torch.tensor([LABEL_TO_ID[l] for l in label], dtype=torch.
        long))

input_ids = torch.stack(input_ids)
attention_masks = torch.stack(attention_masks)
label_ids = torch.stack(label_ids)

dataset = torch.utils.data.TensorDataset(input_ids, attention_masks, label_ids)
```

Enfin, nous avons divisé nos données en ensemble d'entrainement et validation et avons défini les différents paramètres pour entraîner le modèle :

```
1
2  train_dataset, val_dataset = torch.utils.data.random_split(dataset, [train_size
       , val_size])
3
4  train_dataloader = torch.utils.data.DataLoader(train_dataset, batch_size=4,
       shuffle=True)
5  val_dataloader = torch.utils.data.DataLoader(val_dataset, batch_size=4, shuffle
       =False)
6
7  # Training
8  device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
9  model.to(device)
10
11 optimizer = torch.optim.AdamW(model.parameters(), lr=0.00001)
12
13 num_epochs = 100
14 for epoch in range(num_epochs):
15     print("Epoch : ",epoch)
16     model.train()
17     total_loss = 0
18     for batch in train_dataloader:
19         batch = tuple(t.to(device) for t in batch)
20         inputs = {'input_ids': batch[0], 'attention_mask': batch[1], 'labels':
               batch[2]}
21         outputs = model(**inputs)
22         loss = outputs.loss
23         total_loss += loss.item()
24
25         optimizer.zero_grad()
26         loss.backward()
27         optimizer.step()
```

Malheureusement, le modèle que nous avons développé n'a pas atteint les performances que nous espérions. D'une part, on avait à notre disposition peu de données textuelles annotées pour pouvoir effectuer un transfert learning efficace pour notre tâche. D'autre part, la distribution déséquilibrée des classes entre SYMPTOME et NON-SYMPTOME entraine une difficulté à distinguer les deux classes, car le modèle peut se contenter de prédire la classe majoritaire tout le temps pour atteindre un minimum local facilement. Il existe différentes techniques pour pallier ce problème comme le re-échantillonnage, attribution de poids différents aux exemples des deux classes... mais nous avons opté pour d'autres approches pour éviter ce problème. Lors de l'évaluation de notre modèle, nous avons observé une loss relativement élevée, atteignant 0.0905 pour l'entraînement et 0.1077 pour la validation.

Lors des tests du modèle sur de nouveaux textes, ses performances dans la détection des symptômes étaient décevantes. Malgré les difficultés rencontrées, nous tenions à présenter les travaux que nous avons accomplis, car nous avons investi beaucoup de temps et d'efforts. Cette expérience nous a permis d'acquérir une compréhension plus approfondie de la tâche de transfer learning et d'apprendre de nombreuses leçons précieuses. Dans la suite du projet, nous avons donc utilisé l'ensemble annoté que nous avons obtenu grâce aux expressions régulières.

4 Modélisation

La modélisation constitue la dernière étape de notre projet. Notre objectif est de créer un modèle de machine learning capable de répondre à la question initiale : *comment prédire les diagnostics médicaux les plus probables à partir d'un texte décrivant des symptômes ?*

Une première approche consiste à générer une représentation vectorielle de la description de symptômes d'un patient. Notre objectif est que cette représentation soit pertinente et informative sur la

condition du patient. Après, à partir de la représentation vectorielle des maladies (le "ground truth" texte décrivant les symptômes associés à chaque maladie) et celle du texte qu'on reçoit en entrée (la description du patient), on mesure la distance cosinus entre toutes les paires (maladie - description patient). Le choix de la distance cosinus est dû à sa robustesse aux grandes dimensionnalités des données. Et notre diagnostic au final n'est que la paire avec la plus petite distance.

Pour ce faire, nous avons testé plusieurs techniques de vectorisation :

1. La vectorisation à l'aide de la technique **TF-IDF** (Term Frequency-Inverse Document Frequency) qui est une méthode couramment utilisée pour représenter des documents textuels sous forme de vecteurs numériques. Cette technique permet de capturer l'importance relative des mots dans un document par rapport à une collection de documents. Lors de ce processus, plusieurs transformations sont appliquées afin de limiter le vocabulaire aux mots les plus pertinents pour notre tâche.

2. La vectorisation à l'aide d'un modèle de langue. Cela implique l'utilisation d'un réseau de neurones pré-entraîné sur une tâche de modélisation de langue, tel que la prédiction du mot suivant dans une phrase. Nous avons opté pour le **modèle Camembert de Hugging Face** qui est un modèle de langue basé sur l'architecture des transformers et qui a été pré-entraîné sur une grande quantité de données textuelles en français. Notre méthode de vectorisation consiste à extraire la représentation de l'avant-dernière couche du réseau pour obtenir une représentation dense et basse dimensionnelle du texte du patient. L'idée ici était de retenter une utilisation du modèle Camembert pour la modélisation directement et voir s'il pouvais donner des résultats plus prometteurs que lors de la tentative d'annotation avec le transfert learning. À nouveau, les résultats n'ont pas été satisfaisants.

Notre deuxième approche est de formuler notre problème en une tâche de classification où les diagnostics sont les labels que nous cherchons à prédire. Pour réaliser cette tâche, nous avons suivi trois étapes clés :

1. Extraction des symptômes à l'aide des balises : L'étape d'annotation nous a permis de baliser les différents symptômes dans les textes. Nous avons utilisé une expression régulière pour extraire les symptômes du texte, et ces symptômes ont été stockés dans des listes correspondant à chaque diagnostic.

2. Encodage des différentes listes de symptômes en utilisant la représentation TF-IDF (Term Frequency-Inverse Document Frequency) : Dans notre cas, le document représente la liste de symptômes et le corpus correspond à l'ensemble de toutes les listes de symptômes. TF-IDF ne capture pas les relations contextuelles et sémantiques entre les mots, mais se concentre sur leur importance statistique dans le corpus. Cela nous permet de pondérer davantage les termes spécifiques à un diagnostic, sans tenir compte de l'ordre des symptômes.

3. Entraînement de modèles de classification pour la détection des diagnostics : Nous avons considéré différents modèles tels que les KNN (K Plus Proches Voisins), les arbres de décision, les algorithmes de boosting tels que le Gradient Boosting ou XGBoost, ainsi que les SVM (Machines à Vecteurs de Support). Finalement, nous avons choisi de retenir les modèles KNN et SVM qui ont donné des résultats cohérents et similaires.

Pour le modèle KNN, nous avons paramétré ce dernier avec un nombre de voisins égal à 3 et une métrique de similarité de cosinus. Cette mesure calcule le cosinus de l'angle entre deux vecteurs pour évaluer leur similarité. Elle est indépendante de la taille des vecteurs et se focalise uniquement sur leur direction. Dans notre cas, les vecteurs de symptômes peuvent avoir des tailles différentes, et cette mesure se révèle également efficace en termes de calcul. Quant au modèle SVM, nous l'avons paramétré avec C=1.0, en utilisant un noyau gaussien et une valeur de gamma égale à "scale".

En comparant les deux modèles sur des exemples de tests, nous avons constaté que les résultats étaient souvent similaires. Cependant, lorsque les résultats divergeaient, le modèle KNN tendait à être plus réaliste que le modèle SVM. Par exemple, sur le texte "Je me sens fatigué, j'ai des frissons, j'ai mal à la tête, j'ai une impression de souffle réduit et j'ai une perte de l'odorat. Je suis à bout, qu'est-ce que j'ai ?", le modèle KNN a renvoyé des diagnostics très proches de la réalité tels que *Covid-19*, *Grippe* et *Amyotrophies spinales proximales*. En revanche, le modèle SVM a donné comme diagnostics *Covid-19*,

Cancer du poumon et *Asthme*, ce qui n'est pas réaliste.

La comparaison des modèles que nous avons réalisée repose sur des observations empiriques et nos propres intuitions en tant qu'êtres humains. Cependant, une évaluation approfondie des modèles aurait nécessité un jeu de données de test avec un "ground truth" établi par des experts médicaux, qui détermineraient les diagnostics les plus probables pour chaque texte. Nous aurions ainsi pu mesurer les performances des modèles en termes de pourcentage de diagnostics correctement prédits par rapport à ce "ground truth". Malheureusement, nous n'avons pas eu accès à un tel jeu de données et avons dû nous fier à nos connaissances limitées.

Finalement, nous avons choisi de conserver le modèle KNN pour notre interface graphique, car il produit des résultats assez cohérents par rapport aux descriptions de symptômes de maladies basées sur le site de l'Assurance maladie.

En ce qui concerne l'aspect visuel, nous avons développé une interface graphique simple où l'utilisateur est invité à décrire ses symptômes, et le modèle KNN renvoie les trois diagnostics les plus proches de cette description. Vous trouverez ci-dessous une capture d'écran de cette interface.

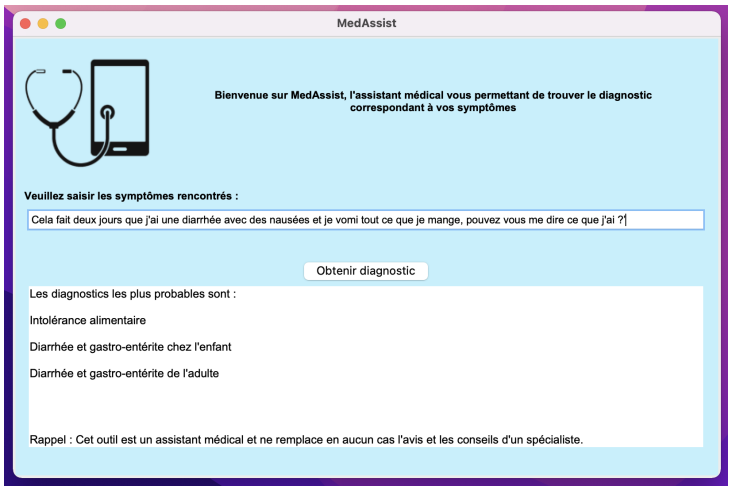

FIGURE 2 – Interface graphique de MedAssist

5 Conclusion

En conclusion, ce projet de NLP a été une expérience enrichissante qui nous a permis de plonger dans le domaine complexe de l'analyse de symptômes médicaux pour prédire des diagnostics probables. Cependant, nous avons également identifié certaines limites importantes tout au long du processus.

Une des principales limites réside dans le besoin d'une expertise médicale approfondie à chaque étape du projet. Depuis la création du corpus de données, en passant par l'annotation précise des symptômes jusqu'à la validation des prédictions du modèle, il est crucial de collaborer étroitement avec des experts médicaux pour garantir la fiabilité et la pertinence des résultats. Sans cette expertise, les erreurs et les biais peuvent se glisser dans le processus, compromettant ainsi la précision et la confiance des prédictions du modèle. Une amélioration envisageable de notre projet consisterait à proposer un traitement adéquat pour les diagnostics identifiés, afin d'avoir un impact réel sur la prise en charge médicale.

Malgré les difficultés, ce projet nous a permis d'acquérir une meilleure connaissance en NLP. Nous avons appris à naviguer dans un corpus de données médicales, à appliquer des techniques de traitement du langage naturel telles que l'annotation et la vectorisation, et à utiliser des modèles de machine learning pour effectuer des prédictions.