

Détection des effets indésirables d'un médicament à partir d'un texte

Eden Belouadah

Bouhaha Mariem

4 janvier 2018

1 Introduction

L'intelligence artificielle a connu de nombreuses applications sur le domaine médical. En effet, nous trouvons souvent des méthodes qui servent à faciliter la vie des patients et à aider les médecins à bien gérer et comprendre les causes des maladies de leur patients en plus d'avoir des statistiques et comprendre des informations difficiles à détecter par l'être humain sur la grande échelle.

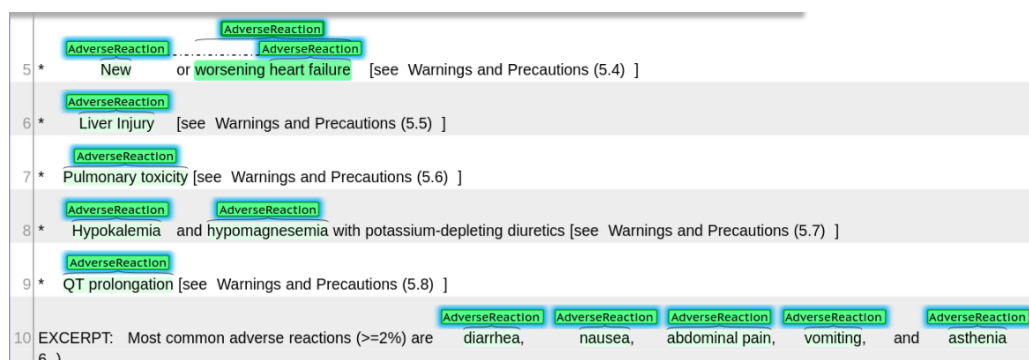
Ce projet consiste à développer un système automatique de détection des effets indésirables d'un médicament à partir d'une notice ou d'un texte quelconque. Nous utilisons pour cette tâche le Thésaurus MedDRA [1].

L'hypothèse de base est de tester Wapiti CRF [2] sur un fichier de base qui contient le mot et sa classe séparés par une tabulation.

2 Matériel

Les données mises à notre disposition pour cette tâche sont les suivantes :

- Un corpus de format *brat*, qui permet de visualiser les concepts d'effets secondaires pour bien comprendre la tâche et s'inspirer pour trouver des idées innovantes. Exemple :



5	*	New or worsening heart failure [see Warnings and Precautions (5.4)]
6	*	Liver Injury [see Warnings and Precautions (5.5)]
7	*	Pulmonary toxicity [see Warnings and Precautions (5.6)]
8	*	Hypokalemia and hypomagnesemia with potassium-depleting diuretics [see Warnings and Precautions (5.7)]
9	*	QT prolongation [see Warnings and Precautions (5.8)]
10		EXCERPT: Most common adverse reactions (>=2%) are diarrhea, nausea, abdominal pain, vomiting, and asthenia

Figure 1. Fichier de format *brat* pour la notice du médicament AMPYRA

- Un corpus qui contient 101 fichiers. Chaque fichier (de format tabulaire) correspond à une notice d'un médicament. Par exemple :

2	ADVERSE	AMPYRA.xml:S1:6:7	O
3	REACTIONS	AMPYRA.xml:S1:14:9	O
4			
5	Because	AMPYRA.xml:S1:27:7	O
6	clinical	AMPYRA.xml:S1:35:8	O
7	studies	AMPYRA.xml:S1:44:7	O
8	are	AMPYRA.xml:S1:52:3	O
9	conducted	AMPYRA.xml:S1:56:9	O
10	under	AMPYRA.xml:S1:66:5	O
11	widely	AMPYRA.xml:S1:72:6	O
12	varying	AMPYRA.xml:S1:79:7	O
13	conditions	AMPYRA.xml:S1:87:10	O
14	,	AMPYRA.xml:S1:97:1	O
15	adverse	AMPYRA.xml:S1:99:7	O
16	reaction	AMPYRA.xml:S1:107:8	O
17	rates	AMPYRA.xml:S1:116:5	O
18	observed	AMPYRA.xml:S1:122:8	O
19	in	AMPYRA.xml:S1:131:2	O
20	the	AMPYRA.xml:S1:134:3	O
21	clinical	AMPYRA.xml:S1:138:8	O

Figure 2. Fichier tabulaire pour la notice du médicament AMPYRA

La première colonne correspond au mot, la deuxième correspond à la position du mot dans le fichier d'origine d'où est construit le fichier tabulaire (Cette colonne est gardée mais elle sera ignorée pendant toutes les phases de traitement). La troisième représente la classe du mot, c'est à dire, est ce que c'est un effet indésirable ou non.

Pas seulement les effets indésirables sont annotés dans les fichiers tabulaires mais tous les autres entités possibles sont annotées. L'étiquette "O" signifie que ce mot ne correspond à aucun concept, l'étiquette précédée par "B" signifie le début d'un concept et l'étiquette précédée par "I" signifie que nous sommes à l'intérieur d'un concept.

Nous nous intéressons donc aux deux étiquettes : B-AdverseReaction et I-AdverseReaction.

3 Méthodes et protocole expérimental

Avant d'entamer la partie traitement, nous avons divisé les fichiers du corpus en trois : Entraînement (60%), développement (20%) et test (20%). Nous faisons l'apprentissage sur l'ensemble d'entraînement, les expérimentations et le réglage des paramètres sur l'ensemble de développement et nous gardons l'ensemble de test pour le calcul des performances finales.

Ensemble	Entraînement	Développement	Test
Nombre de lignes	152274	73228	52908

Tableau 1. Distribution des lignes dans les trois ensemble de données

3.1 Méthodes employées

Les méthodes utilisées sont présentées sous forme d'étapes. Les performances seront représentées dans la section suivante.

3.1.1 Etape 0 : Hypothèse de base

La méthode de base est de considérer un fichier de la forme suivante :

mot ~~*nom_fichier*~~ *classe*

Le fichier de patrons utilisé consiste à prendre en considération le mot courant, les deux mot précédents et les deux mot suivants.

Les résultats obtenus avec ce fichier de patrons de base ne sont pas mal. Cependant, d'autres améliorations peuvent être faites.

3.1.2 Etape 1 : Bigrammes de classes

Nous gardons la même structure de fichier d'entrée et nous modifions le fichier de patrons pour prendre en considération la classe courante, le bigramme (classe précédente, classe courante) et le bigramme (classe courante, classe suivante).

3.1.3 Etape 2 : Racinisation

Nous ajoutons au fichier d'entrée la racine pour chaque mot. La racine permet de trouver la corrélation entre les exemples qui ont des mots qui partagent la même racine.

mot ~~*nom_fichier*~~ *racine* *classe*

Nous avons testé trois racinisateurs : *PorterStemmer*, *LancasterStemmer* et *SnowballStemmer*. Le meilleur résultat est obtenu avec *PorterStemmer* donc nous l'avons gardé pour les prochaines étapes.

3.1.4 Etape 3 : Etiquetage morpho-syntaxique

Nous nous sommes débrouillées avec l'étiqueteur de *NLTK* pour ajouter les étiquettes morpho-syntaxiques correspondantes aux mots. Le fichier d'entrée a maintenant cette forme :

mot ~~*nom_fichier*~~ *racine* *étiquette* *classe*

L'étiquetage morpho-syntaxique permet de bien déterminer le rôle du mot dans la phrase, et faire la différence entre les catégories grammaticales de manière à bien classer le mot.

3.1.5 Etape 4 : Ne pas tenir compte de la casse

Dans cette étape, nous n'avons pas travaillé sur la structure du fichier d'entrée mais nous avons modifié les patrons de telle manière à ne pas respecter la casse. Cette étape semble illogique vu que

certains effets indésirables peuvent effectivement commencer par des majuscules, mais ça a amélioré (un peu) les résultats.

3.1.6 Etape 5 : Lemmatisation

Dans cette étape, nous avons utilisé le *WordNetLemmatiser* pour ajouter les lemmes des mots traités. Ce lemmatiseur nécessite l'information sur la catégorie grammaticale du mot pour pouvoir mieux fonctionner. C'est pour cette raison que nous avons d'abord commencé par ajouter les étiquettes morpho-syntaxiques.

mot ~~*nom_fichier*~~ *racine* *étiquette* *lemme* *classe*

La lemmatisation n'a pas amélioré les résultats, donc nous l'avons enlevé du fichier d'entraînement.

3.1.7 Etape 6 : Représentation vectorielle

Le modèle *word2vec* de la bibliothèque *gensim* est utilisé pour ajouter des embeddings représentant les mots comme d'autres attributs dans le fichier d'entrée. La taille des embeddings est fixée à 5. $\text{embedding}=[X_1, X_2, X_3, X_4, X_5]$. La représentation vectorielle permet de donner une représentation similaire pour les mots proches entre eux.

mot ~~*nom_fichier*~~ *racine* *étiquette* X_1 X_2 X_3 X_4 X_5 *classe*

La représentation vectorielle a été appris à partir du corpus d'entraînement lui même et malheureusement n'a pas amélioré les résultats.

3.1.8 Etape 7 : Elimination des classes inutiles

Dans cette étape, nous éliminons toutes les classes qui ne nous intéressent pas (par exemple I-Drug et B-Drug). Et cela pour alléger le travail du modèle et accélérer le temps d'apprentissage. Donc les seules classes gardées sont : B-AdverseReaction, I-AdverseReaction et O.

3.1.9 Etape 8 : Elimination des mots vides

Cette étape consiste à éliminer les mots vides du corpus. Nous avons testé deux méthodes : Appel à la liste *stopwords* de *NLTK* et élimination des mots selon leur catégorie grammaticale (éliminer les ponctuations et les conjonctions par exemple). Cette étape n'a pas amélioré les résultats non plus.

3.2 Protocole expérimental

Afin de comparer les méthodes et voir si une méthode est de bonne qualité, nous regardons d'abord les critères *token error* et *sequence error*, puis nous mettons l'accent sur la F-mesure des deux classes *B-AdverseReaction* et *I-AdverseReaction*.

4 Résultats

Les performances obtenues sur l'ensemble de développement sont les suivantes :

Etape	0	1	2	3	4	5	6	7	8
Tokens error	5.59%	5.53%	5.18%	4.94%	4.80%	4.89%	5.36%	3.84%	4.96%
Sequences error	35.59%	30.09%	29.58%	27.33%	26.62%	26.90%	27.07%	23.71%	23.62%
F(B-AdvReaction)	72%	73%	74%	75%	77%	77%	72%	77%	75%
F(I-AdvReaction)	62%	66%	68%	68%	69%	69%	61%	68%	67%

Tableau 2. Résultats obtenus pour les différentes étapes

Les étapes hachurées sont les étapes que nous n'avons pas pris en considération dans le modèle final. Les résultats finaux sont les suivants :

Ensemble	Développement	Test
Tokens error	3.84%	3.25%
Sequences error	23.71%	16.71%
F_{mesure}(B-AdverseReaction)	77%	86%
F_{mesure}(I-AdverseReaction)	68%	74%

Tableau 3. Résultats finaux pour le développement et le test

5 Discussion des résultats

Nous remarquons que les résultats s'améliorent d'une étapes à une autre, sauf pour certaines étapes que nous avons décidé de ne pas prendre en considération. Les résultats sur l'ensemble de Test sont très bons par rapports aux résultats que nous avons obtenu lors de la phase de développement.

Il n'y a pas une étape qui a permis de faire un grand saut dans les résultats, mais chaque étape a contribué d'un petit pourcentage dans l'amélioration des résultats. La présentation des résultats détaillés permet de voir l'importance de chaque étape dans le processus d'apprentissage.

5.1 Et si on s'intéresse que si l'étiquette est AdverseReaction ou non ?

Une petite expérience simplificatrice a été faite qui sert à déterminer si un mot est un effet indésirable ou non. C'est à dire, nous avons considéré que les deux étiquettes B-AdverseReaction et I-AdverseReaction sont la même étiquette AdverseReaction. Les résultats obtenus sont les suivants :

Ensemble	Développement	Test
Tokens error	3.72%	3.11%
Sequences error	23.18%	16.50%
$F_{measure}(\text{AdverseReaction})$	76%	82%

Tableau 4. Résultats de l'expérience simplificatrice

Nous remarquons que l'hypothèse simplificatrice n'a pas beaucoup affecté les résultats du modèle.

6 Conclusion

L'hypothèse de base a donnée un résultat acceptable mais il fallait l'enrichir encore pour améliorer les résultats. Parmi les perspectives envisageables, on peut penser à se servir d'autres ressources externes qui permettent de bien caractériser les mots. Ces ressources peuvent être médicales aussi bien que linguistiques.

Ce projet représente un exemple de nombreuses applications de l'apprentissage automatique qui sont dédiées au domaine médical. La détection des effets indésirables peut être étendue à d'autres tâches similaires telles la détection des traitements et des symptômes. Le domaine médical reste un domaine très vaste et plein de défis pour les informaticiens intéressées par l'amélioration de la vie quotidienne des patients et des médecins.

Références

- [1] Thésaurus MedDRA. <https://www.meddra.org/glossary>
- [2] Wapiti - A simple and fast discriminative sequence labelling toolkit, <https://wapiti.limsi.fr>.