Rapport de TP2 NLP - IFT3335

Yuchen Hui 20150470
YuYang Xiong 20151236
JiaDi Yu 20189854

Date: Dec.20 2022

Partie 0 Préface

Ce projet consiste à utiliser différents modèles de classification pour effectuer une NLP tâche: Désambiguïsation de sens du mot “interest” (un mot en englais).

Plus précisément, après avoir prétraité un jeu de données (corpus) de 2000+ phrases contenant différentes formes et utilisations du mot “interest”, nous en extrayons deux types d’information contextuelles comme caractéristiques:

1. l’ensemble des mots autour du mot à désambiguïser.
2. les catégories des mots autour du mot à désambiguïser.

Finalement, nous avons respectivement testé la performance de 5 différents algorithmes de classification (Naïve Bayes, Arbre de décision, Forêt aléatoire, SVM et MLP) sur ces deux types de caractéristiques.

Nous avons publié nos codes sur https://github.com/yujd10/3335-devoir2-topG

Partie I Prétraitement et choix de caractéristiques

Nous avons trouvé que le prétraitement du jeu de données et le choix des caractéristiques jouent un rôle important dans le projet.

Voici la pipeline que nous avons adopté dans ce processus:

1. Enlever les ‘stopwords’.
2. Enlever des ‘==========’ qui existent dans certaines phrases.
3. Enlever les crochets [ ] enfermant les groupes nominaux.
4. Enlever 3 occurrences du mot ‘MGMNP’ qui apparaît seul sans sa catégorie.

   ```
   rest_1/NN ] in/IN [ the/DT qintex/NP bid/NN ] for/IN [ MGMNP ] was/VBD disclosed/V
   ```

5. Split les phrases avec séparateurs ‘space’, localiser les occurrences du mot ‘interest’ et extraire leurs class labels (1,2,3,4,5 ou 6). Pour les occurrences de interest sous forme ‘*interest/catégorie’, nous pensons que ce sont des mots normaux au lieu des mots à désambiguïser les sens.

6. Remplacer “*interest/NN” et "interest{class label}/NN" par interest/NN, en tenant compte du fait qu’il peuvent être utilisés comme des mots de contexte pour autres occurrences de ‘interest’.
7. Distinguer deux types de contextes: mot, et catégories de mot, en lisant la partie gauche et la partie droite de ‘/’ .
8. Stemming. On effectue le stemming en utilisant le ‘*SnowballStemmer*’ du package *nltk.stem.*
9. Étant donné le window size, extraire {window size} mots (ou catégoriee) avant et après une occurrence de ‘interest’ comme caractéristique. L’image ci-dessous illustre 10 tel exemples de caractéristiques avec window size fixé à 2.

```
['expect', 'declin', 'rate', '.']
['indic', 'declin', 'rate', 'permit']
['rise', 'short-term', 'rate', '.']
['83.4', '%', 'energy-servic', 'compani']
['hold', 'compani', 'mechan', 'engin']
[',', 'plus', '.']
['curri', 'set', 'rate', 'refund']
['countri', "'s", ',', 'prompt']
['reduct', 'princip', 'way', 'debt']
['right', 'increas', '70', '%']
['show', 'strong', 'japanes', 'investor']
```

```
['VBP', 'NNS', 'NNS', '.']
['VB', 'VBG', 'NNS', 'VBP']
['NNS', 'JJ', 'NNS', '.']
['AB', 'NN', 'JJ', 'NN']
['VBG', 'NN', 'JJ', 'NN']
[',', 'CC', '.']
['NP', 'VBD', 'NN', 'NN']
['NN', 'POS', ',', 'VBD']
['NN', 'NN', 'NN', 'NN']
['NN', 'VB', 'AB', 'NN']
['VBP', 'JJ', 'JJ', 'NNS']
```

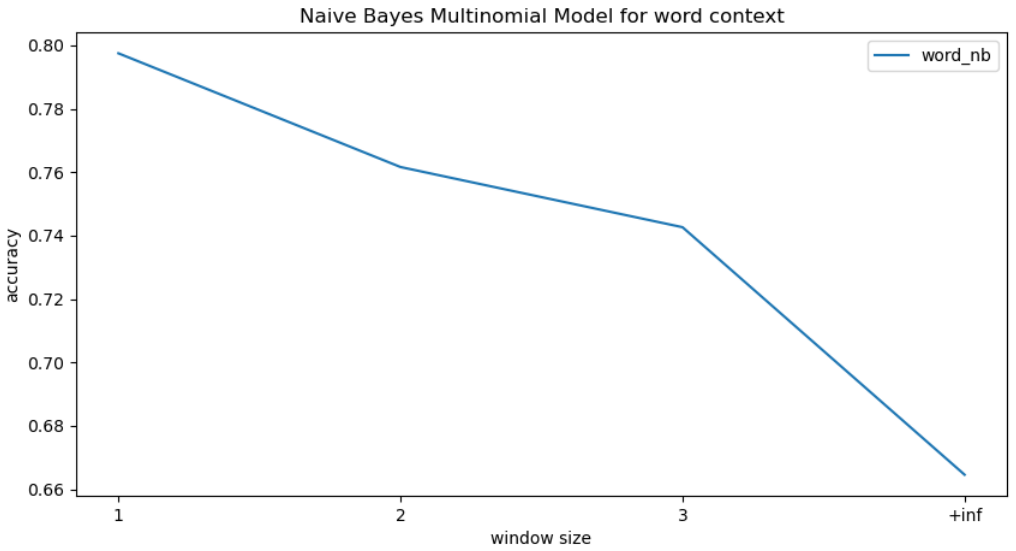
10. Transformation des mots et des catégories en vecteurs de caractéristiques et pondération ces caractéristiques grâce à la classe TfidfVectorizer du Scikit-Learn.
11. Split le jeu de données dont 80% d’échantillons pour l'entraînement et 20% pour le test.

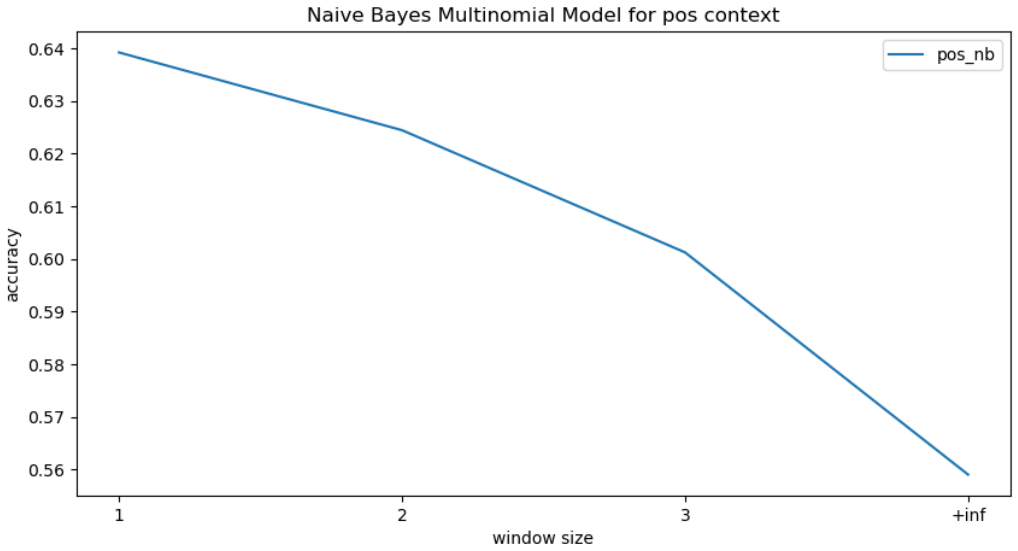
En conclusion, par ce TP, nous avons réalisé l'importance de préparation de jeu de données.

Partie II Entrainement et Test

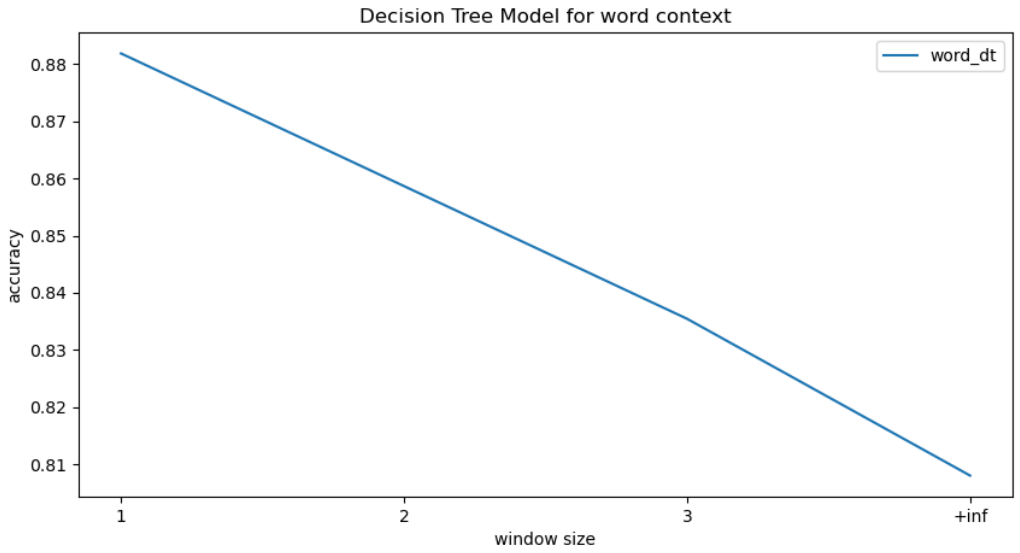
Après avoir vectorisé toutes les phrases que nous avons en utilisant TfidfVectorizer, nous pouvons maintenant les utiliser pour entraîner différents modèles et les tester. Dans cette partie, nous avons entraîné différents modèles, notamment Naive Bayes Multinomial, Decision Tree, Random Forest, SVM et Multi-Layer Proceptron pour différentes données vectorisées, le contexte sous forme de mots et de POS (avec différentes tailles de fenêtre: 1,2,3 et la phrase entière). Pour les modèles MLP, nous avons formé et testé avec différents nombres de neurones cachés (5, 10, 15 neurones).

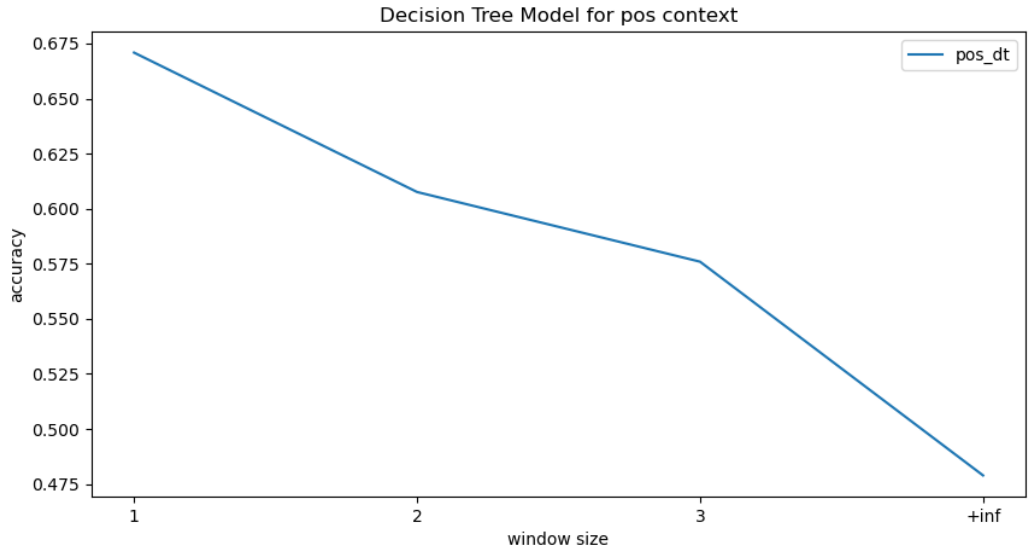
1. Naive Bayes Multinomial
    a. En utilisant les mots comme information contextuelle, nous pouvons voir que le taux de précision diminue au fur et à mesure que nous augmentons la taille de la fenêtre, et a atteint le pire(0.6645) dans le cas de l'utilisation de la phrase entière. Le meilleur est quand window_size = 1(0.7974).
    b. Même cas pour l'utilisation du POS comme information contextuelle, le taux de précision le plus élevé (0,5590) est obtenu lorsque la taille de la fenêtre est de 1.



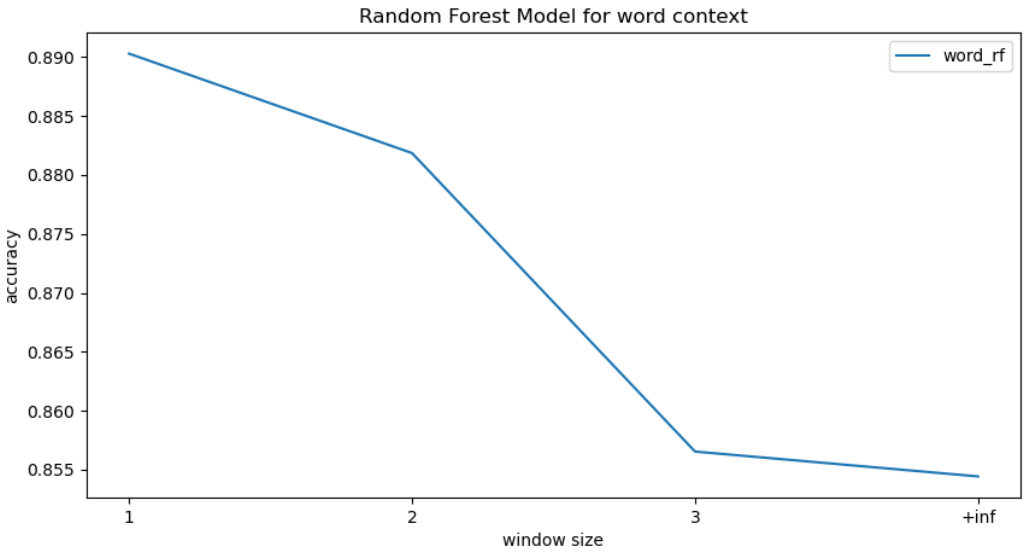


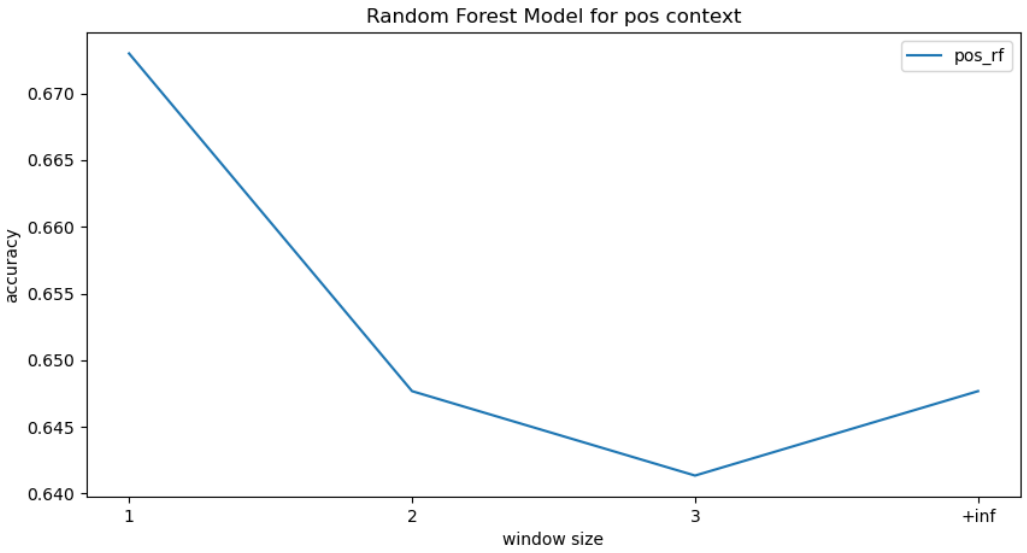
2. Decision Tree
    a. En utilisant les mots comme information contextuelle, nous pouvons voir que le taux de précision diminue toujours au fur et à mesure que nous augmentons la taille de la fenêtre, et a atteint le pire(0.8122) dans le cas de l'utilisation de la phrase entière. Le meilleur est quand window_size = 1(0.8816).
    b. Même cas pour l'utilisation du POS comme information contextuelle, le taux de précision le plus élevé (0,6706) est obtenu lorsque la taille de la fenêtre est de 1.



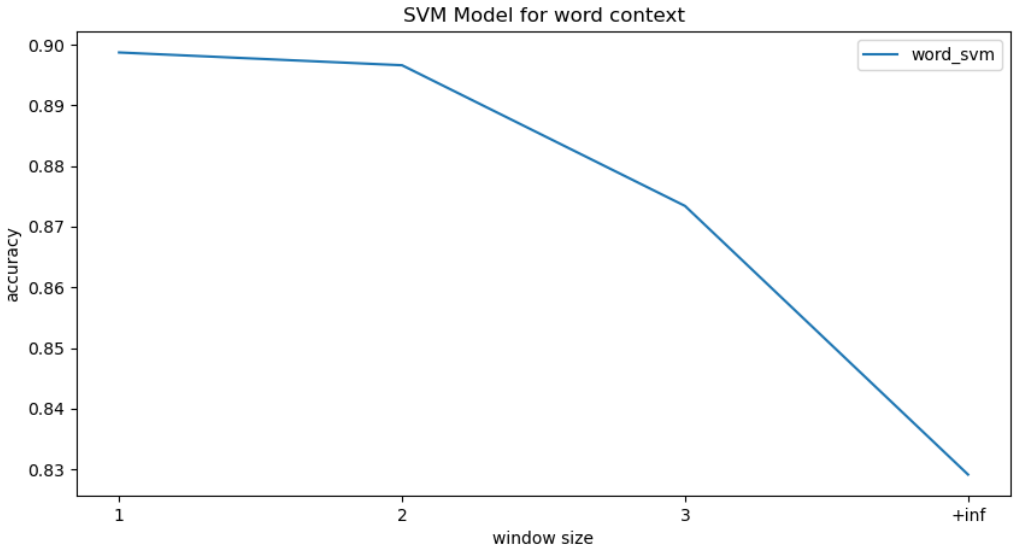


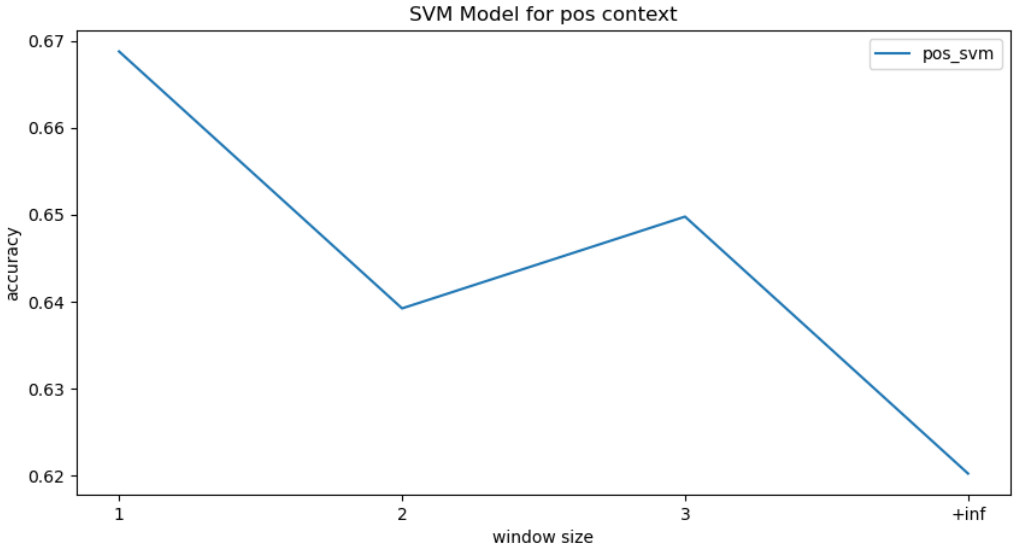
3. Random Forest
    a. Lorsque l'on utilise le contexte des mots, le taux de précision diminue également au fur et à mesure que l'on augmente la taille de la fenêtre, mais il y aura une baisse significative entre les tailles de fenêtre 2 et 3, puis une baisse lente jusqu'à l'utilisation complète d'une phrase.
    b. Lorsque l'on utilise le POS comme information contextuelle, le taux de précision réduit jusqu'à une taille de fenêtre de 3, puis commence à augmenter lentement jusqu'à ce que la taille de la fenêtre ne soit plus limitée.





4. SVM
    a. En utilisant les mots comme information contextuelle, nous pouvons voir que le taux de précision diminue toujours au fur et à mesure que nous augmentons la taille de la fenêtre, et a atteint le pire(0.8289) dans le cas de l'utilisation de la phrase entière. Le meilleur est quand window_size = 1(0.8985).
    b. Lorsque l'on utilise le POS comme information contextuelle, le taux de précision réduit quand la taille de fenêtre augmente de 1 à 2, puis commence à augmenter quand la taille de fenêtre augmente de 2 à 3, finalement recommence à diminuer jusqu'à ce que la taille de la fenêtre ne soit plus limitée.



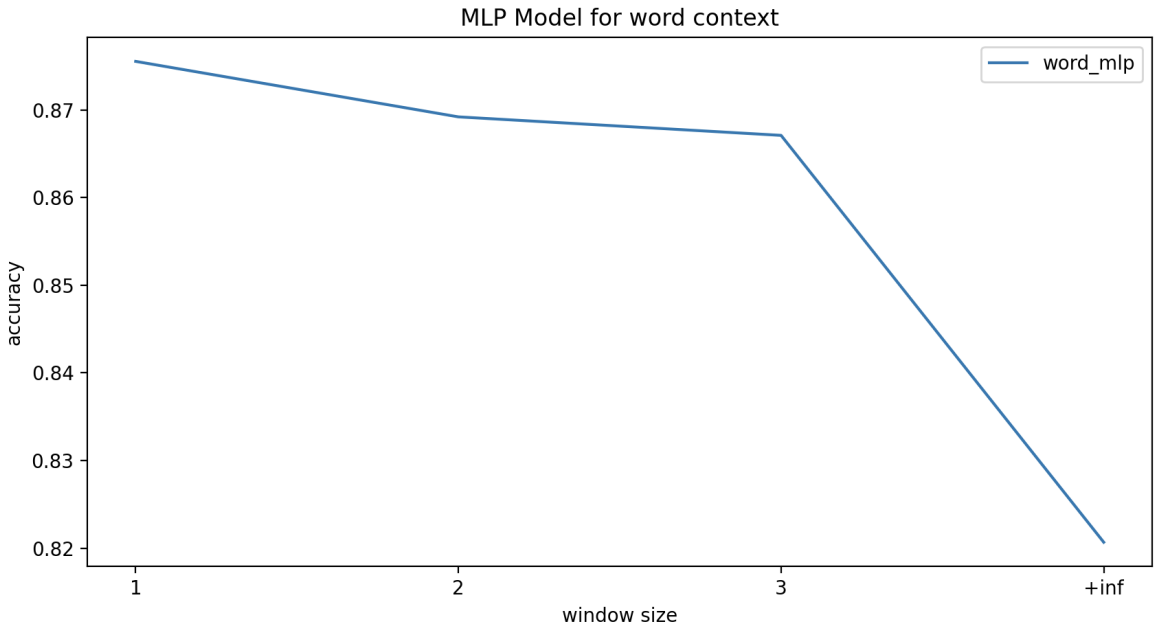


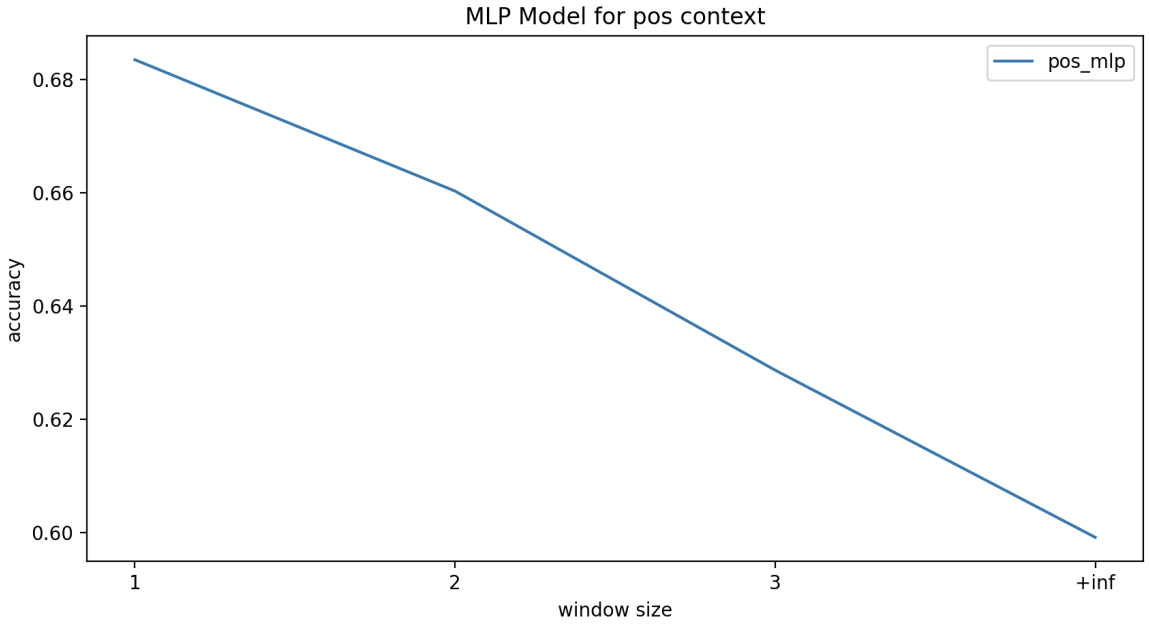
5. MLP (Multi-Layer Perceptron)

   Nous ne considérons d'abord que le nombre de neurones cachés = 10:

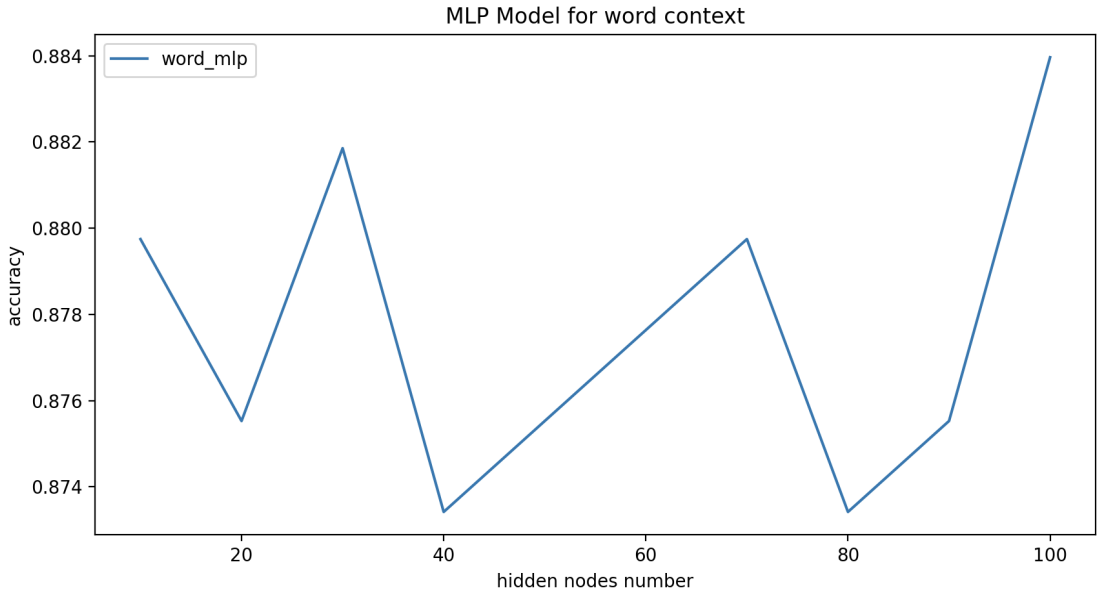
   a. En utilisant les mots comme information contextuelle, nous pouvons voir que le taux de précision diminue toujours au fur et à mesure que nous augmentons la taille de la fenêtre, et a atteint le pire(0.8200) dans le cas de l'utilisation de la phrase entière. Le meilleur est quand window_size = 1(0.8755).

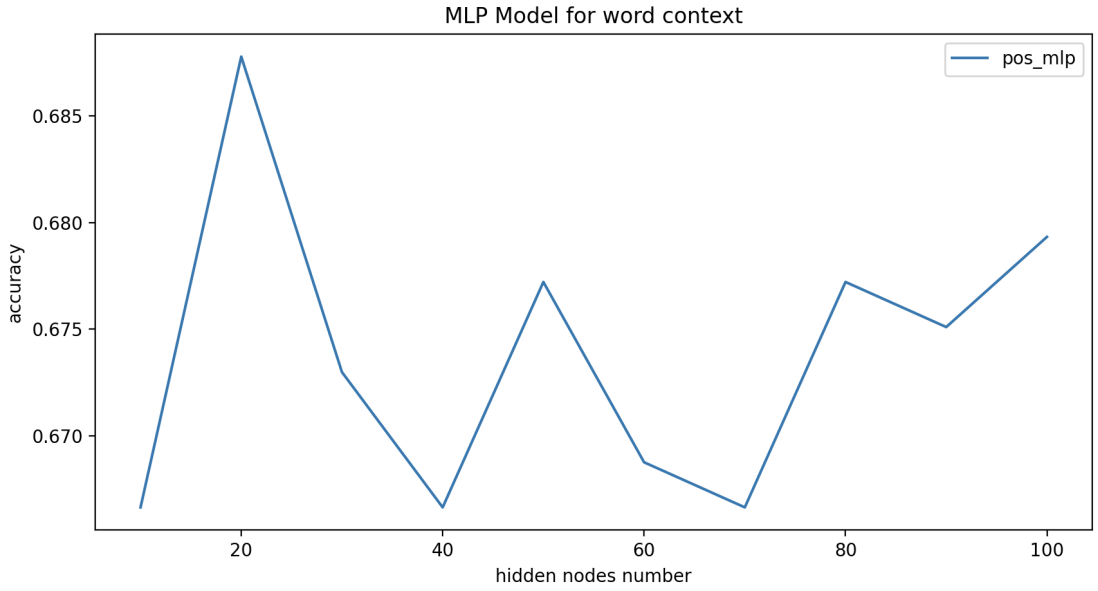
   b. Même cas pour l'utilisation du POS comme information contextuelle, le taux de précision le plus élevé (0,6838) est obtenu lorsque la taille de la fenêtre est de 1.





Maintenant nous comparez les différents résultats au fur et à mesure que le nombre de neurones change, fixons le format de fenêtre à 1 puis changer le nombre de neurones de la couche cachée (10, 20, 30, …, 100): nous trouvons que l'exactitude ne converge pas au fur et à mesure que le nombre de neurones augmente.





Finalement, nous avons trouvé que l'exactitude de l'ensemble de test, lorsque nous utilisons les mots autour comme l'ensemble d'entraînement, est toujours plus élevée que lorsque les catégories autour sont utilisées comme l'ensemble d'entraînement: sauf que l'exactitude la plus élevée du Multinomial Naïve

Bayes est seulement de presque 80%, les autres algorithmes ont tous une exactitude au moins plus élevée que 80%, même proche de 90%. Ainsi nous pouvons conclure que le contexte de mots avant et après est plus approprié pour l'entraînement.

Pourquoi l'exactitude du Multinomial Naïve Bayes est considérablement plus basse que les autres algorithmes? Une des possibles raisons est que le Multinomial Naïve Bayes est un classifieur linéaire tandis que les autres sont non-linéaires, qui ont une plus grande capacité et peuvent simuler des fonctions plus complexes.