Documentation du POC NPNearest

Cyril Vincent

**Introduction**

Utilisation des techniques d’IA pour rechercher les produits les plus proches d’un produit dans NextPage.

Utilisation d’un modèle de Deep Learning DAN (Deep Average Network) préentrainé de Google nommé USE (Universal Sentence Encoder) et d’un modèle de Machine Learning nommé Gestalt,Pattern Matching inventé par Ratcliff et Obershelp modifié par mes soins, le tout avec le framework Google TensorFlow 2.3.



**Installation**

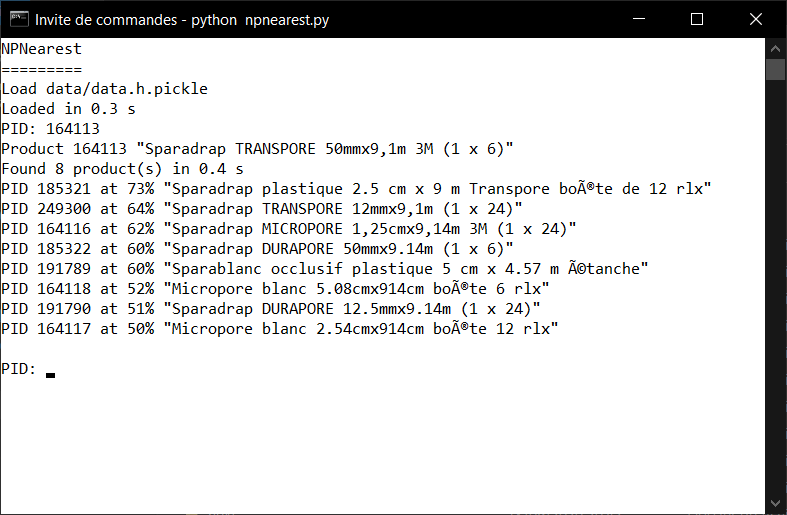
* Sur un poste qui n’a pas Visual Studio 2019, installer le package redistribuable Microsoft Visual C ++ pour Visual Studio 2015, 2017 et 2019 : <https://aka.ms/vs/16/release/vc_redist.x64.exe>
* Dézipper le .rar fournit

**NPNearest**

Pour exécuter NPNearest il faut ouvrir un invité de commande avec CMD, puis depuis le répertoire npnearest taper npnearest.exe



Entrer un product id, par exemple 164113



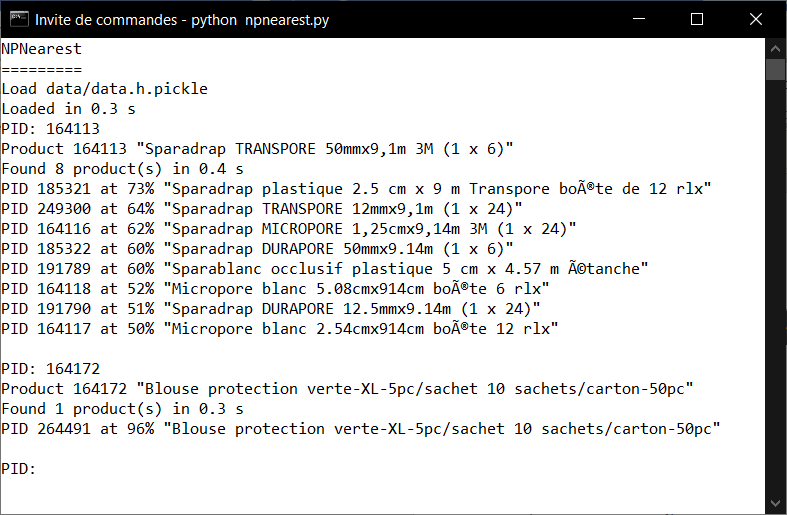
NPNearest a trouvé 8 produits proches ordonnées par un score en %

Signification des scores :

* >99% : c’est le même produit
* >95% : il y a des chances que ce soit le même produit
* >90% : produit extrêmement proche voir identique
* >70% : produit différent mais similaire
* >50% : produit différent mais avec quelques caractéristiques proches
* <50% : non significatif

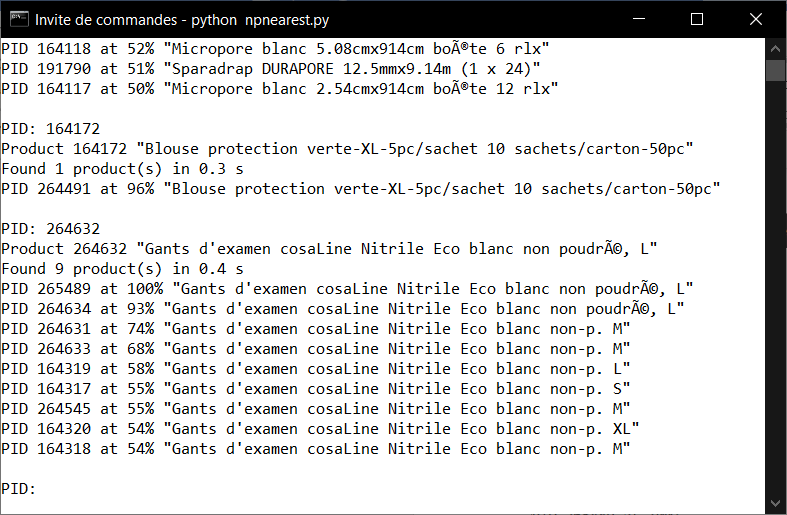
Dans cet exemple il n’a donc aucun produit identique et un produit différent mail similaire

Essayons avec le produit 164172



Regardons le bas de l’écran, dans ce cas il ne trouve qu’un seul produit mais qui est très proche voir identique.

Essayons avec 264632



Dans ce cas il trouve 9 produits dont 1 produit identique et 1 très proche, les autres sont similaires.

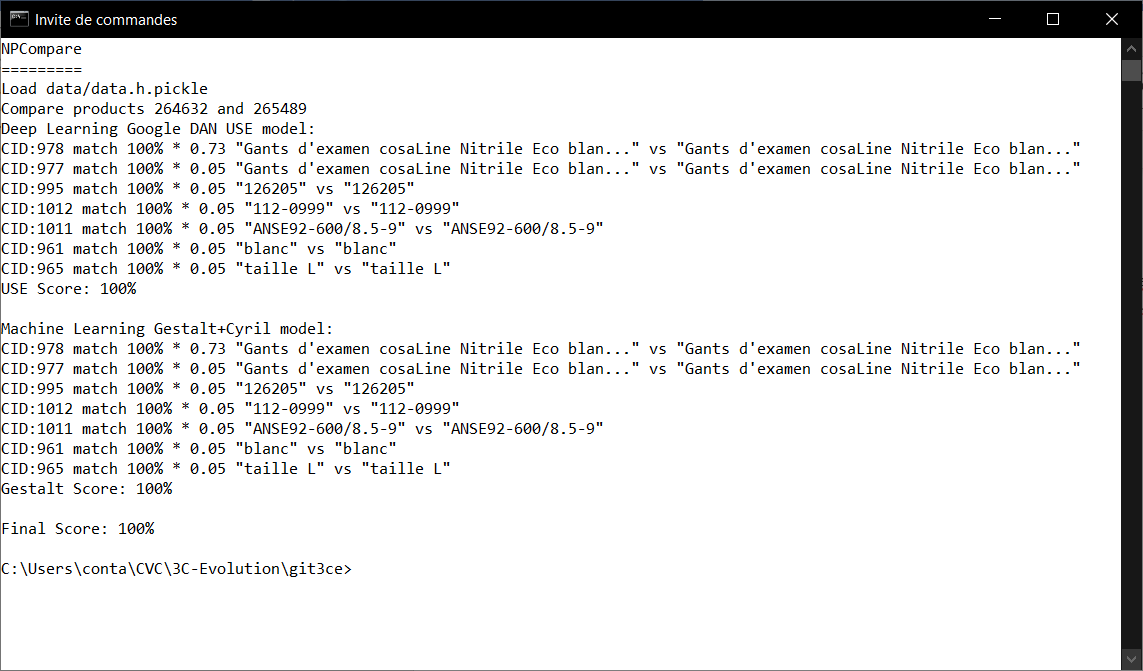
**NPCompare**

Afin de comprendre le calcul du score un autre programme est fourni : npcompare.exe

Il se démarre dans un invité de commande : cmd, il faut ensuite naviguer jusqu’au répertoire du programme avec la commande cd

Puis taper npcompare pid1 pid2

Par exemple : npcompare.exe 264632 265489



Cet outil compare 2 produits et donne le détail du score avec les 2 modèles.

Ne pas hésiter à agrandir la fenêtre.

La partie Deep Learning Google DAN USE model donne le résultat du calcul avec le Deep Learning.

Pour chaque caractéristiques (CID) un score est affiché multiplié par une pondération puis le texte comparé

Dans cet exemple : CID:978 match 100% \* 0.73 "Gants d'examen cosaLine Nitrile Eco blan..." vs "Gants d'examen cosaLine Nitrile Eco blan..." veut dire que pour la carac 978 le score est 100% avec une pondération à 0.73 puis les 2 textes comparés

USE Score détermine le score total du modèle de Deep Learning.

La même chose est faite pour le modèle de Machine Learning Gestalt.

Le score final est la moyenne des 2 scores

Le modèle USE est le plus performant, il a comme petit défaut d’être sévère et de générer un peu des faux négatifs. Les poids des caractéristiques sont modifiés automatiquement par le modèle en se réentrainant afin de gérer automatiquement les caractéristiques avec un poids important.

Le modèle Gestalt+Cyril est un peu moins performant et il a tendance à générer des faux positifs. Les poids sont modifiés par mon algorithme afin de mieux gérer les caractéristiques importantes.

La moyenne des 2 modèles donne un très bon résultat en évitant les faux négatifs et les faux positifs

**Etude de cas**

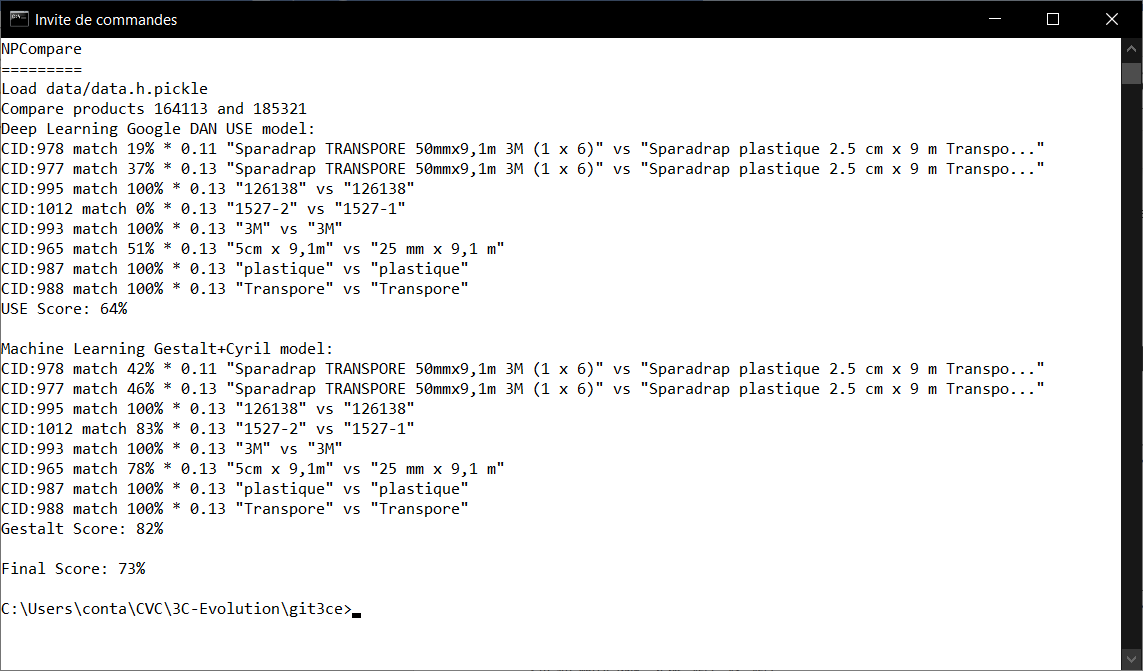
L’exemple précédent avec 264632 265489 est trop simple car tout match à 100%, il s’agit dont du même produit.

**Exemple avec 2 produits quasi identiques mais avec une caractéristique différente : 164172 et 264491**



Tout est identique sauf la caractéristique 955. Le score de 96% indique qu’il s’agit certainement du même produit. Si la caractéristiques 995 ne participe pas assez à la discrimination des deux produits il suffit d’augmenter son poids.

**Cas ambigus de 2 produits avec des ressemblances mais différents :**



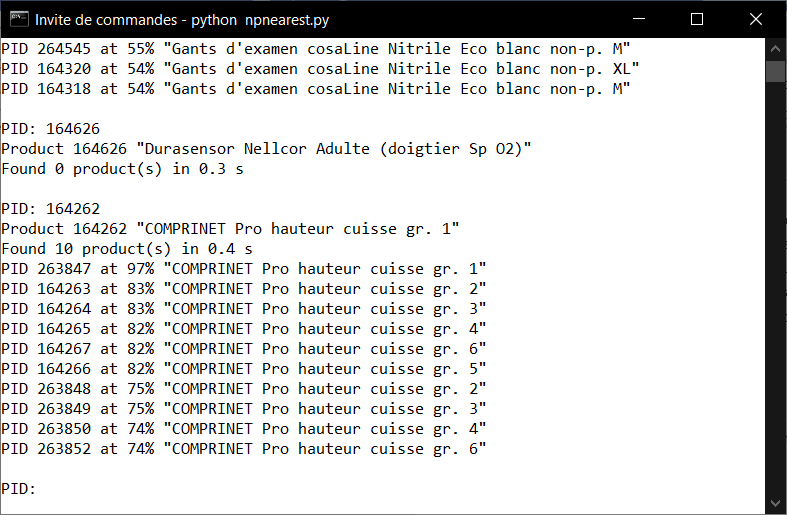
On voit que les différences sont bien perçues mais il y a quand même assez de point commun pour signifier 73%, ce qui signifie que ce sont des produits différents mais avec des points communs.

On voit également que USE est toujours plus sévère que Gestalt, à raison dans ce cas.

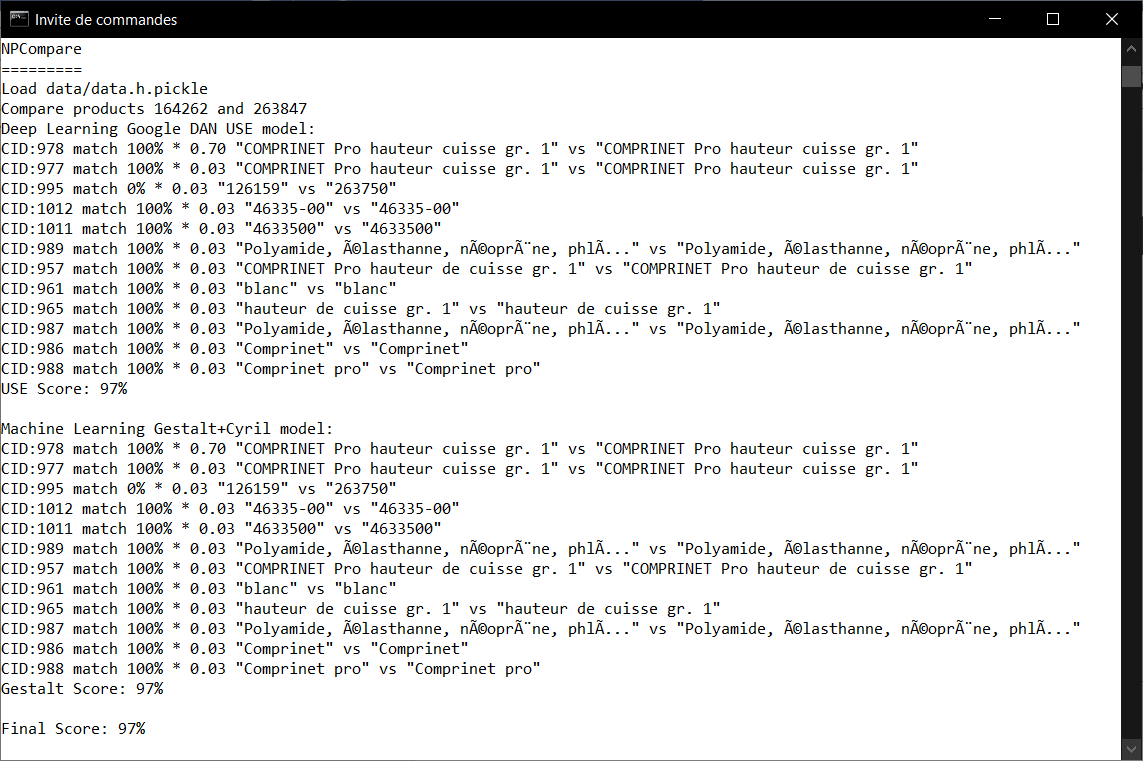
A l’inverse du cas précédent si la caractéristique 965 participe trop à la discrimination des deux produits il suffit d’augmenter sa pondération.

**Cas d’un faux positif potentiel**

164262 : COMPRINET Pro hauteur cuisse gr. 1 possède beaucoup de produit quasi similaire où seul le grammage change.

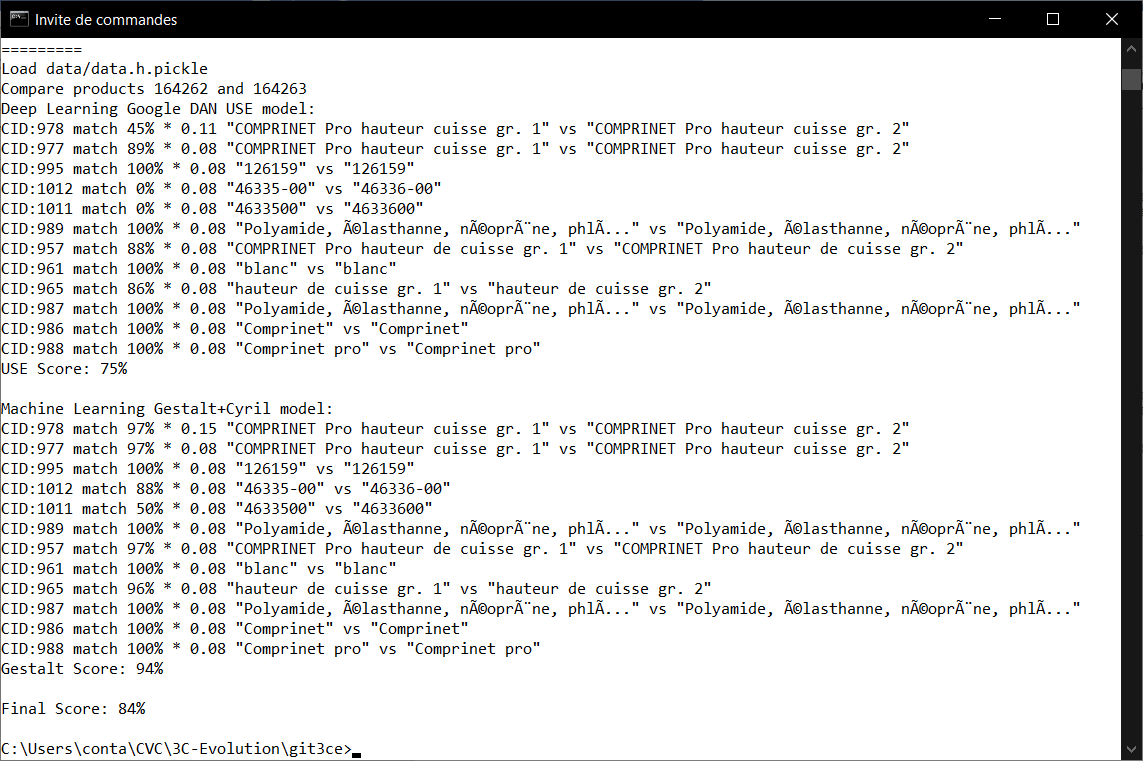


Le produit le + proche est 263847 à 97%, vérifions



Il s’agit peut-être du même produits malgré un différence sur la caractéristique 995

Le second produit le + proche est 164263, vérifions :



Attention il s’agit d’un faux positif potentiel

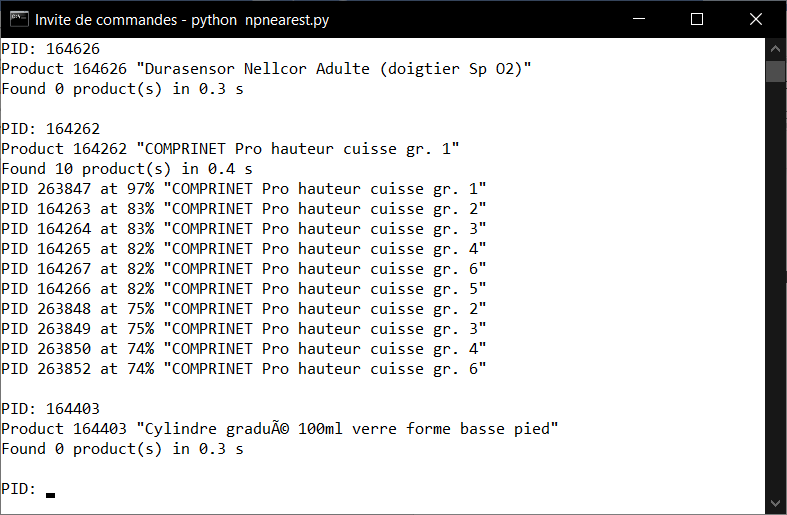
En effet, tout se ressemble beaucoup, mais ce n’est pas le même produit

On voit que USE détecte bien que ce n’est pas le même produit avec un score de 75%

A l’inverse Gestalt donne un faux positif de 94%

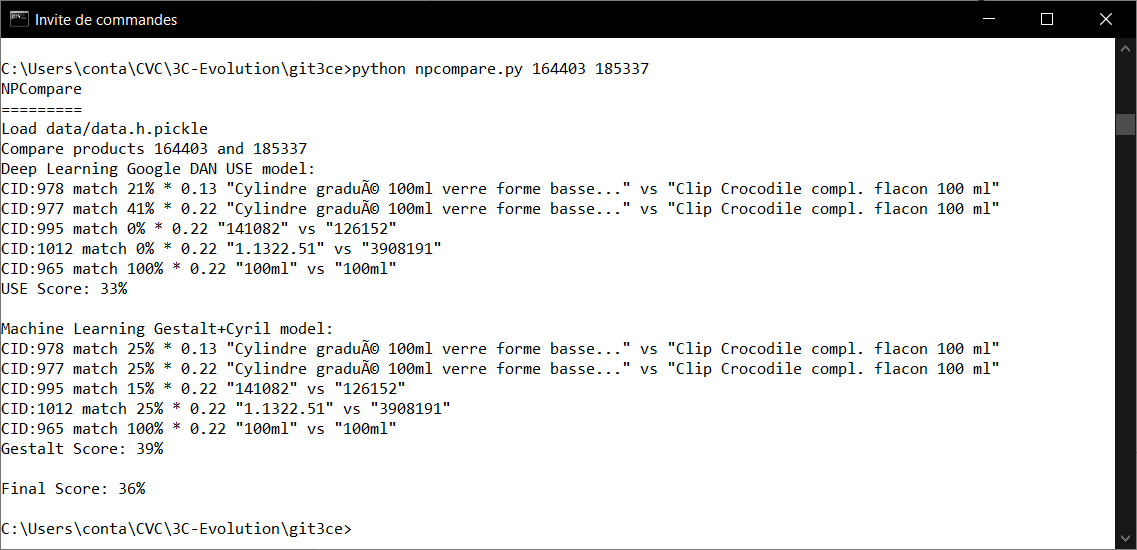
La moyenne des deux donne un résultat cohérant : 84%, il s’agit de 2 produits très proches mais non identiques

**Cas d’un produit sans produit proche, un faux négatif potentiel**



164403 n’a pas de produit proche détecté

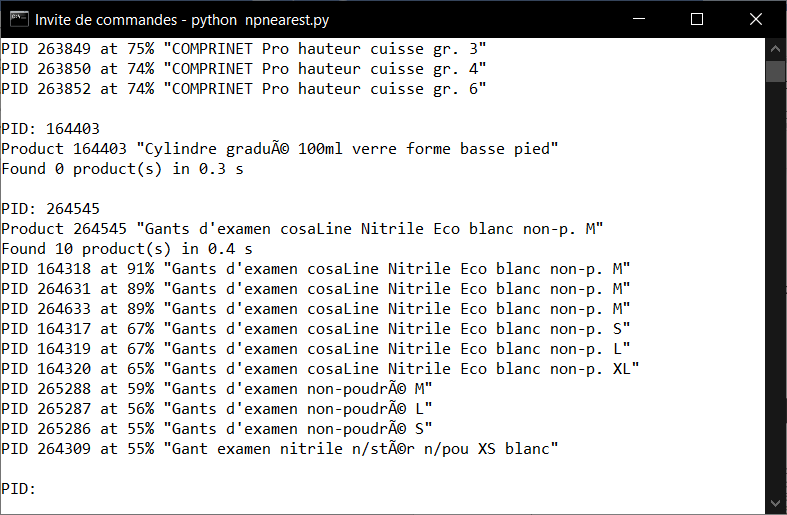
En recherchant dans les modèles, le produit le plus proche semble être le 185337, vérifions qu’il ne s’agit pas d’un faux négatif

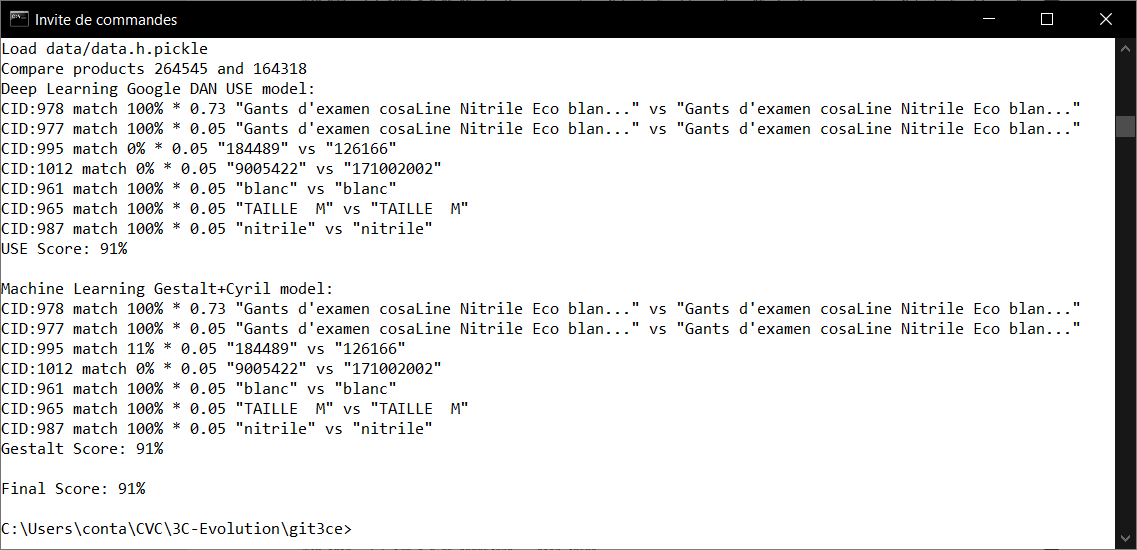


Il y a bien des petites ressemblances mais pas assez pour être > 50%

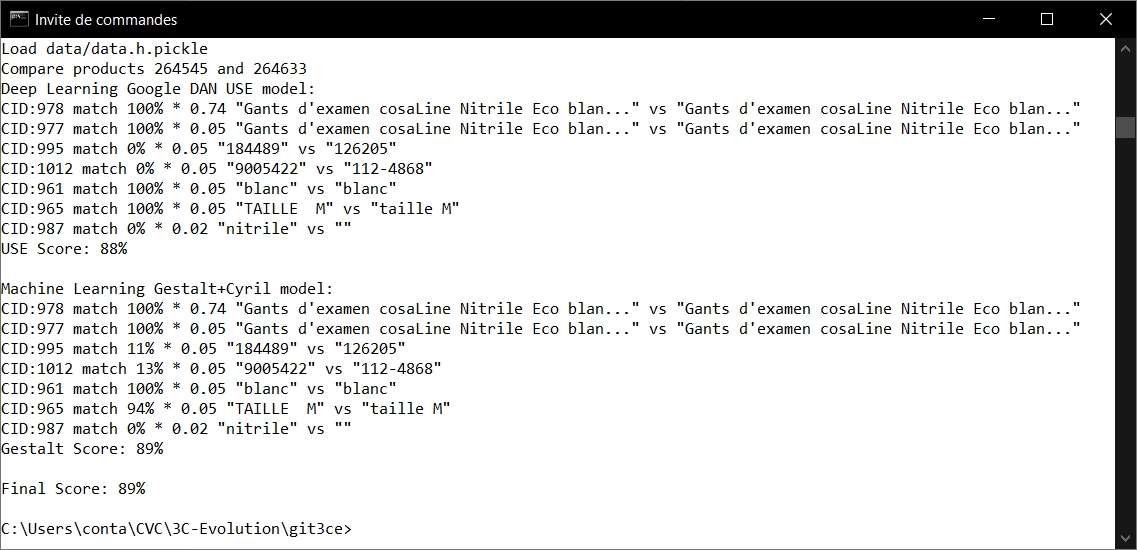
**Produit avec beaucoup de doublons**

Le produit 264545 "Gants d'examen cosaLine Nitrile Eco blan" semble avoir beaucoup de doublons





164318 semble être un doublon

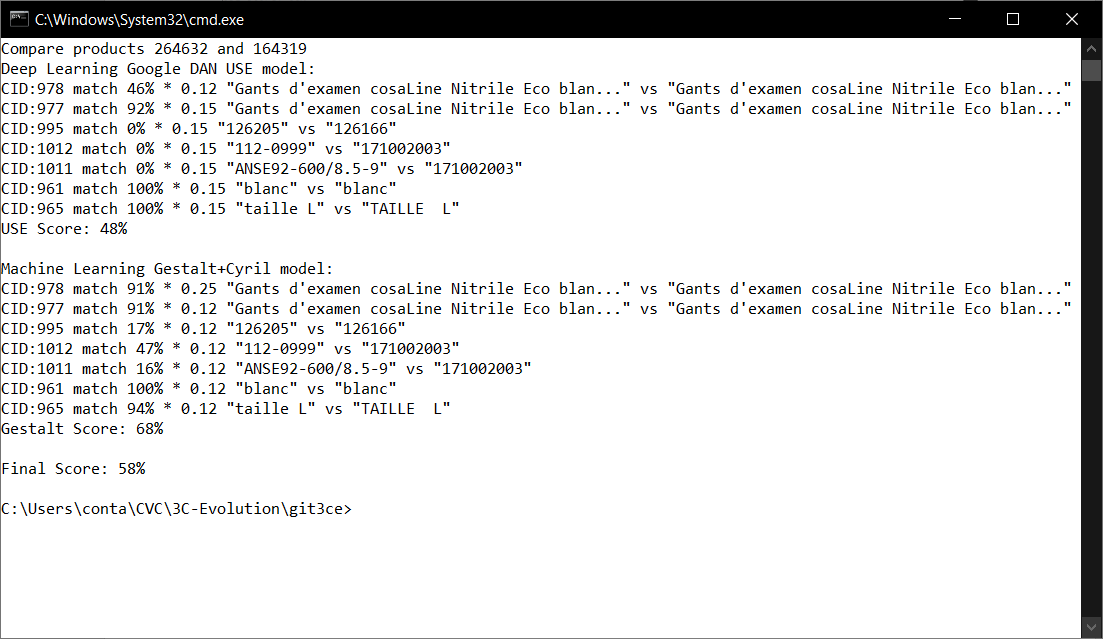


264633 bien que légèrement inférieur à 90% est peut-être un doublon (à vérifier). Idem pour 264631

Les autres produits ont quasiment le même nom seul les tailles changent. Le programme détecte bien qu’il s’agit de produits différents avec des scores <70% .

**Cas difficile où tout diffère et pourtant c’est peut-être le même produit**

264632 et 164319 est un cas difficile

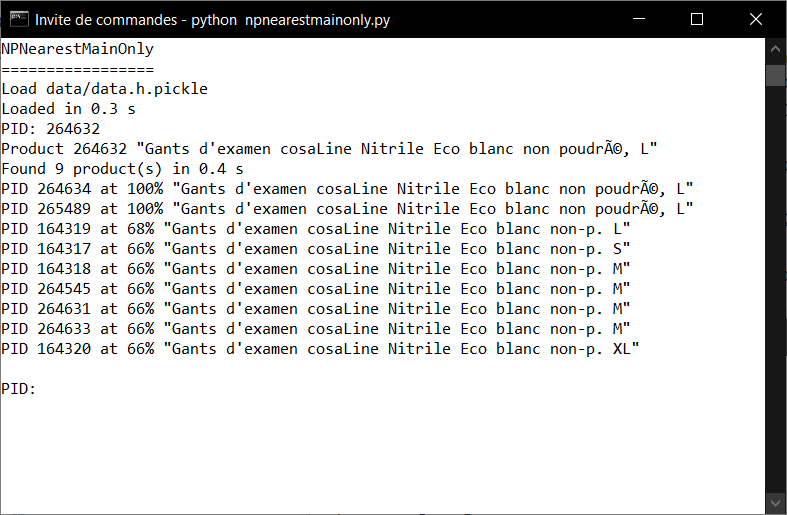


D’après les valeurs tout diffère et pourtant, les caractéristiques 977 et 978 sont presque les mêmes, cas à étudier de prêt "Gants d'examen cosaLine Nitrile Eco blanc non poudrÃ©, L" vs "Gants d'examen cosaLine Nitrile Eco blanc non-p. L"

**Option -mainonly**

Pour contrer le cas précédent j’ai créé l’option --mainonly qui test uniquement la caractéristique principale

Utilisation : exécuter npnearestmainonly.exe --mainonly



Dans ce cas il détecte d’avantage 164319 mais sans tenir compte des autres caractéristiques

**En conclusion NPNearest et NPCompare détectent bien les produits similaires et évitent les faux positifs et faux négatifs.**

Il est important de noter que NPNearest n’est pas lié au catalogue testé, il s’adapte automatiquement à n’importe quel catalogue après entrainement.

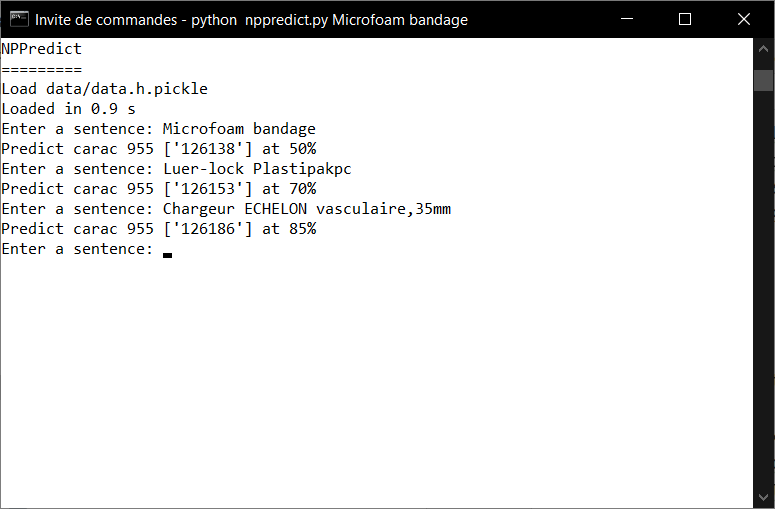
**Prévision et catégorisation**

J’ai tenté rapidement de faire un petit essai sur la prévision d’une caractéristique. Ceci pourrait également fonctionner pour un produit lié, une famille ou une sous famille.

Le but est à partir d’un libellé de prédire la valeur d’une caractéristique.

Dans nppredict.exe j’ai essayé de prédire la valeur de la caractéristique 955 en fonction du libellé du produit (sa référence)

Utilisation : démarrer nppredict.exe en ligne de commande



Dans cet exemple il prédit pour Chargeur ECHELON vasculaire,35mm que la carac 955 devrait être 126186 avec 85% de certitude.

Après vérification des 3 exemples ils sont justes.

J’ai très peu testé ce programme qui peut être largement amélioré car l’apprentissage a été fait trop rapidement.

Ce programme a été fait avec le DAN USE de Google auquel j’ai ajouté un réseau neuronal de type MLP (Multi Layer Perceptron)

**Efficacité**

Une prédiction par apprentissage profond nécessite environ 1 minutes pour 4000 produits et 1h pour 100000 !

J’ai donc utilisé une technique d’indexation NoSql pour améliorer les performances.

J’ai généré des fichiers fictifs avec 10000 et 100000 produits pour tester les temps de traitement.

Voici les performances en prédictions des produits proches sur mon PC perso (i7 récent + GPU + 16Go de RAM) :

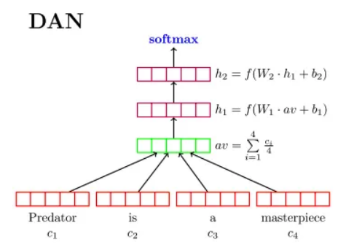
* Pour 3904 produits et 15 caractéristiques : 0.3s
* Pour 10000 produits et 5 caractéristiques : 1.4s
* Pour 100000 produits et 3 caractéristiques : 7.1s

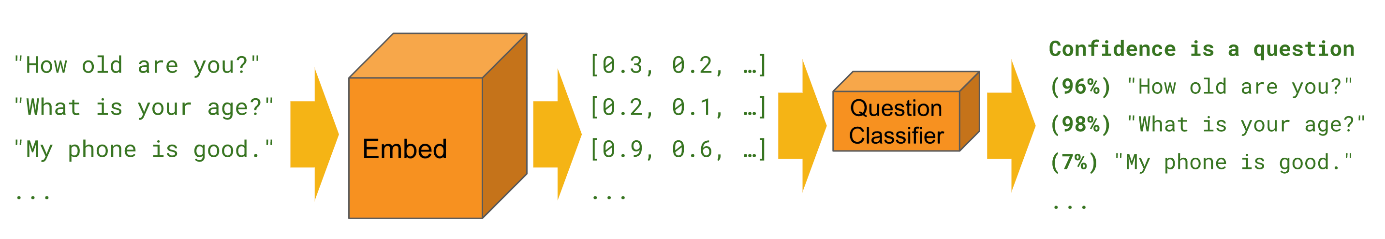
Le temps d’apprentissage est le suivant :

* Pour 3904 produits et 15 caractéristiques : 12s et 50Mo de RAM
* Pour 10000 produits et 5 caractéristiques : 57s et 230Mo de RAM
* Pour 100000 produits et 3 caractéristiques : 6 minutes et 1.4Go de RAM

**Détail d’utilisation du modèle DAN USE**

* Conversion du fichier en UTF8
* Clonage du modèle DAN USE 4
* Adaptation des poids par transfert learning
* Création de l’indexation des caractéristiques en JSON et Pickle
* Hachage des caractéristiques par prédiction du modèle avec création d’un vecteur de 512 flottants 64 bits
* Stockage des hachages en index
* Recherche des plus proches voisins par application d’un produit vectoriel interne sur le hachage avec similarité cosine
* Modification et normalisation des poids





Détail d’utilisation du modèles Gestalt+Cyril

* Réutilisation de l’index du précédent modèle
* Application du modèle Gestalt sur les valeurs par une opération matricielle et une fonction de normalisation uniquement sur les caractéristiques dont le score Use est > 0.5
* Modification des poids par un algorithme créé par mes soins
* Normalisation des poids

**Industrialisation et intégration dans NextPage**

Ceci est un POC, mais il est très facile d’industrialiser le code

Pour ceci il suffit d’encapsuler l’apprentissage et la prévision soit dans un service REST multi-process.

NP pourra facilement l’interroger par requêtes REST + JSON, soit par un wrapper .NET, un petit transfert de compétence doit avoir lieu pour installer et utiliser le service

Coût : 2j à 3j

Un transfert de compétence ou une formation Deep Learning peut avoir lieu sur une durée de 2 à 5 jours suivant le niveau des développeurs

**Opportunités futures**

* Catégorisation : catégoriser un produit automatiquement dans une famille et/ou sous famille
* Prédiction : prédire un produit lié ou une caractéristique
* Détection des incohérences : Détecter une caractéristique avec une valeur incohérente
* Reporting : Afficher tous les produits similaires entre eux à plus de 90%
* Templating : trouver automatiquement pour une famille un template de texte pour une caractéristique
* Templating bis : Trouver tous les produits ne respectant pas un template de texte pour une catégorie
* Rapprochement des images : Trouver les produits les plus proches par image
* Catégorisation par image : Trouver automatiquement la famille et/ou sous famille d’un produit par son image
* Prévision par image : Trouver les produits, les familles les plus proche d’une image quelconque, par exemple prise en photo depuis un téléphone

Ces fonctionnalités peuvent se faire en POC entre 2 et 4 jours chacune.

Pour les modèles utilisant des images, la qualité de prédiction est difficile à prévoir si les images sont très proches les unes des autres, par exemple dans un catalogue de freins.

**Références**

TensorFlow : <https://www.tensorflow.org/?hl=fr>

Deep Average Network : <https://mlexplained.com/2018/05/11/paper-dissected-deep-unordered-composition-rivals-syntactic-methods-for-text-classification-explained>

Google USE : <https://cloud.google.com/solutions/machine-learning/analyzing-text-semantic-similarity-using-tensorflow-and-cloud-dataflow?hl=fr>

Modèle Google USE : <https://tfhub.dev/google/universal-sentence-encoder/4>

Gestalt : <https://en.wikipedia.org/wiki/Gestalt_Pattern_Matching>

Gestalt librairie : <https://docs.python.org/3.8/library/difflib.html>