Catégorisation de produits pour le e-commerce

Nicolas Gaude

Introduction —

cdiscount opère une offre “marketplace” qui référence des produits en provenance de commerçants partenaires. Pour assurer une visibilité maximale de ces produits sur son site, cdiscount doit assurer une catégorisation homogène de ces produits aux origines variées.

Approche —

Devant le grand nombre de catégories, la richesse de l'espace des mots et les forts déséquilibres en nombre de produits par catégorie, nous avons structuré notre proposition autour d'une approche bagging / staging via un Ensemble de pyramides de Régressions Logistiques : «staged-logistic-regressions ensemble»

Critères —

Nous souhaitons souligner dans ce préambule qu'avant la performance, nos deux critères personnels ont été la qualité de généralisation et la capacité d'industrialisation de la proposition.

«En particulier les algorithmes utilisés ne requièrent pas d'apprentissage semi-supervisé sur le dataset de test “test.csv” ce qui assure la généralisation du modèle dans le cadre d'une mise en production : le vocabulaire du dataset de test n'a donc pas été intégré durant la phase d'apprentissage et, de même, la distribution estimée des catégories du dataset de test n'a pas été réinjectée dans le modèle ce qui aurait pu toutefois artificiellement améliorer la performance dans ce challenge au détriment de la généralisation. e.g. Lorsqu'un partenaire externe spécialisé «JARDIN-PISCINE» viendra déverser son catalogue dans le «marketplace» nous souhaitons que le modèle présente la même qualité de catégorisation que pour un commerçant plutôt versé «INFORMATIQUE». Ceci alors que le dataset de test a une distribution de catégories très particulière, à dessein de complétude du test plus que de réalisme.

Dans le même souci de généralisation, nous n'avons pas non plus intégré les informations associées au “Prix” qui nécessiteraient selon nous une mise à jour du modèle en cas de changement parité monétaire important avec un pays partenaire e.g. $/€, ou en cas de rareté exceptionnelle d'un produit e.g. composants informatiques, jouets période noël.

C'est donc sur la seule base des champs «Marque», «Description» et «Libelle» présents dans le dataset «train.csv» et sur la hiérarchie des catégories décrite dans rayon.csv que cette proposition a été construite.»

I. Algorithme

I problématique

La difficulté du challenge proposé se décompose en 4 problèmes majeurs que notre proposition va adresser.

En premier lieu, le déséquilibre des «Categorie3» (x1 vs x2M) représentées dans le dataset d'apprentissage est un écueil pour bon nombre de classifieurs. Il faudra choisir une méthode d’échantillonnage adaptée permettant de redresser «a priori» les biais du dataset d'apprentissage : **Sampling**

En second, l'apparente diversité des descriptions de produits (6M mots uniques pour 15M de produits) à catégoriser donne à penser que l'espace des features ne sera pas facilement réductible. Conserver cette richesse sera l'enjeu de la vectorisation : **Vectorisation**

En troisième, le nombre de catégorie qu'il faut prédire est très élevé (5789 Categorie3) ce qui nous contraint d'emblée à une approche multi-classe bien calibrée pour lever les ambiguïtés inter-classe et garder un modèle de taille raisonnable en mémoire. Le classifieur retenu devra pouvoir combiner à la fois un espace de features et un espace de classe de grandes dimensions : **Classification**

Enfin, en quatrième, la qualité de catégorisation du dataset d'apprentissage étant elle-même sujette à caution (5% de Produit_Cdiscount seulement), il faudra introduire une forme adéquate de régularisation et d'assemblage des prédictions pour ne pas restituer ce bruit de catégorisation : **Ensemble**

II sampling

Le choix d'un échantillonnage adapté est de loin le facteur qui impacte le plus la performance du modèle.

Dans le dataset d'apprentissage, la médiane du nombre d'échantillons par catégorie est de ~150. mais 10 % des catégories sont illustrées par moins de 6 échantillons tandis qu'à l'opposé 10 % des catégories sont illustrées par plus de 3K échantillons. Mention spéciale aux 2M de coques de téléphone.

Un concept important est que ce déséquilibre est la signature de la réalité du catalogue disponible. Par ailleurs, il ne préfigure en rien la distribution des catégories qui devront être prédites ensuite, dans le dataset de test, ou lors de l'ingestion d'un nouveau catalogue de produits en provenance d'un commerçant partenaire : «on ajoute au catalogue généralement ce qui nous manque justement en catalogue».

Le choix que nous avons fait est alors de considérer comme «prior» une distribution uniforme des catégories. Comme soulevé en introduction, il aurait été, à court terme, plus performant d'apprendre dans un premier temps la distribution du dataset de test afin d’entraîner spécifiquement notre modèle à résoudre le problème du dataset de test. Mais cela aurait été préjudiciable à ses qualités ultérieures de généralisation.

L'échantillonnage du dataset d'apprentissage est effectué donc de façon totalement équilibrée autour d'une valeur pivot choisie entre la moyenne (2000) et la médiane du nombre (150) d'échantillons par categorie3 e.g. **Stratified Sampling**.

Dans le cas simple du sous-échantillonnage, nous sélectionnons sans remise le nombre désiré d’échantillons. Dans le cas du sur-échantillonnage, nous avons finalement opté pour un simple tirage avec remise.

Note : nous avons implémenté des techniques plus avancées de sur-échantillonnage comme SMOTE et ADASYN mais les résultats en validation se sont révélés peu probants en regard de l'incrément de complexité.

III Vectorisation

Afin de limiter les dimensions de l'espace des features tout en conservant les informations essentielles nous avons classiquement pré-traité les champs «Libelle» et «Description» en ôtant notamment ~600 mots-courants (stopwords) de la langue française, les balises HTML, les caractères numériques et avons racinisé les mots (stemming). Le texte associé à un produit est par la suite défini comme la concaténation de la triple répétition de la «Marque» suivie de la double répétition du «Libelle» et enfin de la «Description». Ceci permet sur le dataset d'apprentissage de réduire le nombre de mots uniques de 6M à environ 1M.

Concernant la vectorisation en elle-même, notre choix s'est porté sur une approche bag-of-word de type TF-IDF mono-gram et bi-gram (1-2-grams) qui permet de faire ressortir l'importance relative de chaque mot et couple de mots consécutifs dans un texte-produit par rapport au corpus-catalogue. Les textes associés aux produits étant davantage une liste de termes descriptifs que des phrases construites, il n'y avait pas d'intérêt à avoir des n-grams trop longs : les 1-2-grams permettent la désambiguïsation dans certains cas e.g. «film *de* protection» vs «film comédie». Les poids associés aux 1-2-gram ont été choisis comme le rapport des fréquences logarithmiques e.g. log(1+TF)/log(1+DF) afin de conserver la notion d'importance d'un mot par rapport au corpus et également de rendre compte du fait que le corpus-catalogue peut être d'une nature différente de celle du corpus-test. En validation, cette TF-IDF nous a fourni des résultats sensiblement meilleurs qu'un simple bag-of-word binaire.

Note : sur ce challenge les approches bag-of-word ont battu invariablement les représentations plus «deep» telles que word-vector et paragraph-vector ce qui retranscrit selon nous l'intuition que les textes associés aux produits sont avant tout des «listes à la Prévert» non réductibles à des concepts topic-produit.

IV Classification

Convaincus que les vecteurs obtenus possédaient de bonnes propriétés linéaires nous avons choisi comme classifieur de base une régression logistique OAA. L'avantage à disposer d' un tel classifieur probabiliste, plutôt qu'un SVM ou un KNN par exemple, est qu'il est alors possible de partitionner les catégories-classes et d'entraîner spécifiquement un classifieur sur chaque partition pour ensuite combiner les étages de classification selon une règle de composition bayésienne e.g **Staged Logistic Regression.**

A contrario, supposons que nous souhaitions classifier les 5789 classes de type Categorie3 depuis un espace de features de dimensions 500K (ce qui correspond plus ou moins au résultat d'un stratified sampling avec 400 échantillons par Catégorie3-classe comme décrit à l'étape précédente), alors la matrice des coefficients linéaires du modèle serait de taille 500Kx5789x64b = 24GB uniquement pour stocker le modèle. A cela devrait s'ajouter lors de l'apprentissage le gradient ou la hessienne, doublant au moins encore l'espace mémoire requis. Ainsi pour des raisons pratiques, nous avons souhaité décomposer la prédiction de la Categorie3 d'abord par une prédiction de la Categorie1 vue comme une partition de l'espace des Categorie3 grâce à la hiérarchie décrite dans «rayon.csv». Le rationnel derrière cette approche est triple : D'abord il s'agit évidemment de limiter l'empreinte mémoire du processus d'apprentissage. De plus, en partitionnant l'espace de recherche, nous allons lever des ambiguïtés syntaxiques (film de protection == informatique, film comédie == culture). Enfin nous allons également re-vectoriser plus efficacement les échantillons selon telle ou telle Categorie1 supposée.

Nous avons donc un premier étage de régression logistique qui prédit pour chaque échantillon la probabilité de telle ou telle Categorie1 :

$$P(Y=Categorie1).$$

Ensuite nous entraînons un deuxième étage de régressions logistiques, pour chacune des 52 partitions-Categorie1, qui prédit pour chaque échantillon la probabilité de telle ou telle Categorie3 sachant la Categorie1 :

$$P(Y=Categorie3|Categorie1)$$

Il est alors trivial de calculer la probabilité totale d'une catégorie3 en particulier en combinant les deux étages de prédictions et d'obtenir une prédiction basée sur le MLE de la catégorie3 :

$$P(Y=Categorie3).1=P(Y=Categorie1).P(Y=Categorie3|Categorie1)$$
$$Categorie3(Y) = argmax\{Categorie3\}\ P(Y=Categorie3)$$

Note : le MLE est bien calculé comme l'argument maximal parmi les 5789 probabilités des Categorie3. Nous avons essayé de simplifier l'approche en prenant le MLE de Categorie1 du premier étage pour n'avoir à choisir notre MLE du deuxième étage que sur les ~100 Categorie3 associé à une Catégorie1 particulière. Mais ainsi en validation le score s'est avéré sensiblement moins élevé (~-0.5%). Notre interprétation est qu'il existe des Categorie1 ambiguës, qui se recouvrent naturellement (ex SPORTS et VETEMENTS) et pour lesquelles il convient d'explorer l'ensemble des Categorie3 avant de se prononcer.

Concernant l'empreinte mémoire de cette approche : La re-vectorisation des sous ensembles d'échantillons dans chacune des Categorie1, ayant un espace de mot plus réduit, permet de réduire et ré-équilibrer naturellement les dimensions en entrée des 52 régressions logistiques du deuxième étage à environ 50K features (en sortie de TF-IDF) pour un stratified sampling type de 400 échantillons par Catégorie3. Ainsi le poids de ce modèle est de (500Kx52 + 50Kx5789)x64b = 2,5GB.

Note : ce modèle de staged-logistic-regressions permet de ne pas avoir à faire de concession sur la dimension de l'espace d'entrée (500K) ni sur celle de l'espace de sortie (5789) tout en maîtrisant l'enveloppe mémoire et le temps de calcul.

Note : sur ce type de problème de classification où la variance fine des données d'entrées aide à la catégorisation, les performances en régression logistique pure se sont avérées non seulement meilleures qu'avec une SGD/vowpal mais aussi pour un temps de calcul au final équivalent grâce au «trick» de l'imbrication des modèles qui sont entraînés en parallèle.

Note : nous avons également exploré la piste d'une

pyramide de classifieurs à 3 étages en utilisant également les Categorie2 intermédiaires mais il apparaît que la puissance de désambiguïsation est déjà en partie apportée par le premier étage Categorie1. Aussi quoique le gain en validation ait été mesuré (~+0.5%) nous n'avons pas retenu ce modèle en regard de la complexité qu'il apportait: +536 classifieurs par pyramide.

V Ensemble

Une fois notre première pyramide entraînée, un constat en cross-validation s'impose : la recherche des paramètres optimaux que nous avons exécutée sur chacun des 1+52 classifieurs nous oriente invariablement vers plus de variance e.g. la cross validation de la vectorisation nous amène à garder le maximum de n-gram possible (toute tentative de réduction par SVD s'est soldée par une performance en baisse) et de même lors du training des classifieurs, la régularisation L1-lasso n'a donné aucun résultat (même si nous avions l'intuition de pouvoir sélectionner des features clés) et la régularisation L2-ridge optimale (~1e-2) induit un biais limité sur la prédiction globale .

Ceci nous a amené à penser le problème de catégorisation d'une façon duale : prédire la catégorie3, c'est à la fois trouver un produit «quasi-identique» dans la base d'apprentissage (low bias) et également trouver un «demi-plan» qui peut correspondre (low variance). Il faut donc que l'algorithme soit imbattable quand le produit existe déjà dans la base d'apprentissage mais également qu'il puisse généraliser quand le produit possède une formulation textuelle «originale».

Or, dans l'étape de sampling, en équilibrant les classes nous avons perdu une grande partie de la base d'apprentissage : un stratified sampling de 400 échantillons par Categorie3 contient 2,3M de produits mais si l'on retire l'effet du sur-échantillonnage il contient en tout et pour tout 1,1M de produits uniques : soit 1/14 de la base d'apprentissage. L'intuition est donc de considérer chaque sampling et sa pyramide associée comme une instance d'un bagging, et d'ensuite accumuler dans un Ensemble ces instances afin de consolider la prédiction.

L'étape de consolidation des prédictions retenue n'est pas un simple vote de majorité mais favorise les prédictions «quasi-certaines» car elles ont probablement appris parfaitement tel ou tel produit. Au final, le faible biais des classifieurs conserve bien ces «certitudes», dont nous pouvons profiter alors dans l'étape de consolidation.

Soit P les probabilités prédites par chaque pyramide alors la probabilité prédite par l'Ensemble est l'estimateur de la moyenne et la categorie3 prédite par l'Ensemble est à nouveau le MLE de ces 5789 moyennes :

$$Pensemble(Y=Categorie3) = mean(P(Y=Categorie3))$$
$$Categorie3(Y) = argmax\{Categorie3\}\ Pensemble(Y=Categorie3)$$

La validation s'effectue simplement par l'estimation du nombre de pyramides qu'il faut entraîner sur le le dataset d'apprentissage afin qu'un maximum d'échantillons ait été «vu» par au moins une des pyramides de l'Ensemble.

Note : basiquement nous avons vu que dans le cadre d'un stratified sampling à 400 échantillons, chaque pyramide apprend alors sur environ 1/14 du dataset d'apprentissage. Nous avons fixé à 70 pyramides la taille de l'ensemble que nous souhaitions produire pour ce challenge : les simulations indiquent que l'ensemble produit aura été entraîné sur ~50% des données au total, et sur ~100% pour 98% des categorie3 (2% des categorie3 ayant une cardinalité supérieure à 28K)

II. PERFORMANCES

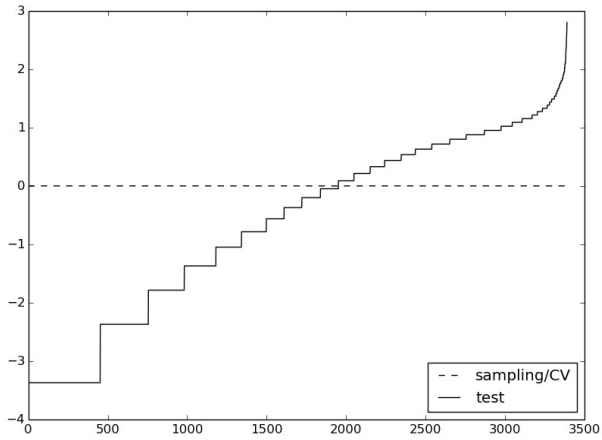
I Classifieur Simple ~66.3%

Chaque pyramide consiste en un apprentissage sur la base d'un sampling équilibré de 400 échantillons avec une coupure pour les Categorie3 ayant moins de 7 échantillons. Telle qu'implémentée, la phase de sampling nécessite le chargement en mémoire de la totalité du dataset d'apprentissage soit ~4GB et prend actuellement 7mn en mono-core.

Le dataset équilibré puis vectorisé (au global, puis par categorie3) nécessite 2GB principalement pour stocker les dictionnaires de mots associés au n-grams, cette étape prend environ 4mn. Afin de réduire cette empreinte mémoire nous avons étendu la classe TF-IDF de sklearn pour utiliser des «marisa-trie» afin de stocker plus efficacement ces dictionnaires. Cette optimisation mémoire permet de réduire l'empreinte mémoire à ~500MB mais nécessite deux passes de vectorisations soit environ 7mn.

Le training, effectué avec la classe LogisticRegression de sklearn, est parallélisé : l'étage 1 de la pyramide est entraîné en monocore en 1h45mn et nécessite 3GB en mémoire et 1GB pour stocker le modèle appris. L'étage 3 est entraîné en parallèle sur 3 cores en 1h30 pour les 52 régressions logistiques, l'espace mémoire requis n'excédant pas 8GB et la taille totale des 52 modèles stockés est d'environ 3GB .

Le temps global d'entraînement d'une pyramide est d'environ 2h30. La performance d'une pyramide en cross-validation est ~73% avec une variance de ~1,5%. En test publique, la performance mesurée est de 66,75%. En test privé, la performance mesurée est de 66,3%. Les variations obtenues en cross validation et en test sont, selon nous, imputables aux différences de distribution des Catégorie3



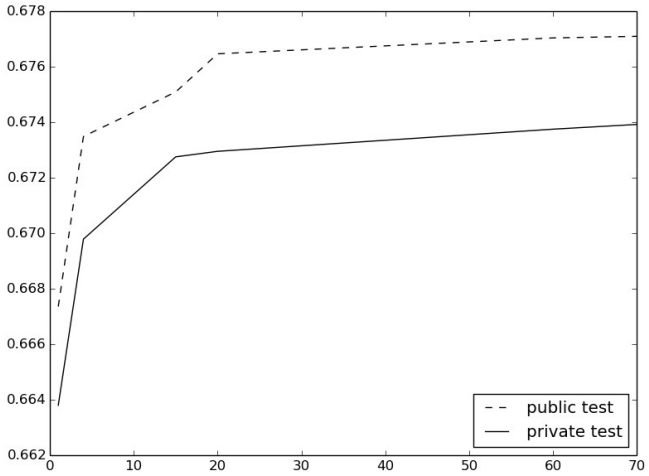
Ci-dessus : la log2-distribution du nombre d'échantillons par Categorie3 estimé dans le dataset de test comparée à la «prior»

utilisée (distribution uniforme) dans le dataset de sampling et de cross-validation.

Note : dans le cadre d'un «information-leakage», il aurait été possible d'entraîner nos classifieurs avec une «prior» distribution de Categorie3 en accord avec le dataset de test pour un gain certain de performance mais sans grande valeur en production.

II Ensemble complet ~ 67.4%

Il reste maintenant à accumuler les pyramides précédentes afin de consolider au maximum les informations présentes dans le dataset d'apprentissage et réduire la variance du classifieur unique que nous avions constatée en cross validation (~1,5%). Sur la base de 7 pyramides par jour il nous a fallu ~10j pour monter l'Ensemble souhaité. Nous avons en pratique constaté qu'un optimum performance vs temps de calcul se situait autour de 20 pyramides.



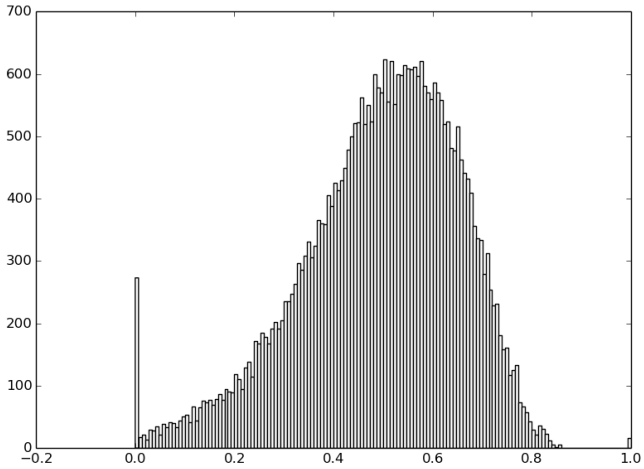
Ci dessus : la progression du score mesuré sur le test publique et test privé d'ensembles composés d'un nombre croissant de pyramides entre 1 (66,3%) et 70 (67,4%)

III Post : Duplication et Évidence ~ 68%

Enfin, une fois l'Ensemble entraîné, nous avons intégré deux mécanismes correctifs en sortie de l'Ensemble qui améliorent certains de ses effets de lissage : la recherche de produits dupliqués et un moteur de règles d'évidence.

Duplication : il s'agit d'une simple recherche des produits «quasi-identiques» entre le dataset de test et le dataset d'apprentissage. Pour l'identification des duplications nous avons utilisé un moteur de recherche de texte (http://code.google.com/p/apporo/) construit avec une indexation des seuls 900K «Produit_Cdiscount» et pour «query» les 35K produits du dataset de test. Sur la base de distances de Levenshtein «normalisées» [0,1] nous avons choisi une valeur limite en deçà de laquelle nous considérons le produit test comme identique au produit d'apprentissage. Le temps de requêtage du système est d'environ 1mn pour 1000 «queries». Sur les 2000 prédictions «strictes» fournies par ce système environ 5% ont apporté effectivement des corrections aux prédictions de l'Ensemble ; soit un gain de performance d'environ 0.3%.

Ci-dessus : la distribution des distances de levenshtein des



Produit_Cdiscount proches voisins du dataset de test. La distance de coupure a été choisie à 0.12 (les «pures» duplications sont en «0»).

Evidence : lors de nos premières analyses du dataset d'apprentissage, nous avions remarqué que les «Categorie3_Name» disponibles dans le fichier «rayon.csv» semblaient avoir été choisis comme étant la chaîne de caractères la plus longue maximisant la probabilité d'être incluse dans le «Libelle» (ou la «Description»). Nous avons alors calculé sur l'ensemble du dataset d'apprentissage et pour chaque Categorie3 les probabilités suivantes :

Ptp=P(Categorie3_Name in Libelle, Categorie3 |Categorie1)
Pfp=P(Categorie3_Name in Libelle, not Categorie3 |Categorie1)

Ptp peut être vu comme la probabilité True-Positive e.g le nom de la catégorie apparaît dans le libellé, et il s'agit bien d'un produit de cette catégorie. Au contraire Pfp peut être vu comme la probabilité False-Positive e.g le nom de la catégorie apparaît dans le libellé mais malheureusement ce n'est pas un produit de cette catégorie.

Ensuite, connaissant le score de validation de l'Ensemble (~73%), nous avons alors choisi pour règle «stricte» d'affecter la Categorie3, sachant la Categorie1, à tous les produits dont le Libellé contient Categorie3_Name, si Ptp > 5*Pfp.

Appliquée sur le dataset de test (en regard des probabilités calculées sur le dataset d'apprentissage) une cinquantaine de Categorie3_Name ont fait l'objet d'une règle d'évidence (exemple «Fontaine *à* Chocolat»). Ces règles ont prédit 300 résultats dont ~100 sont des corrections apportées aux prédictions de l'Ensemble ; soit un gain de 0.3%

Note : il est important lors du calcul de Ptp et de Pfp de redresser les probabilités par les fréquences d'apparition des Categorie3 considérées (e.g. stratified sampling uniforme).

Note : il est important également de garder la Categorie1 comme un pré-filtre pour désambiguïser certaines Categorie3_Name proches mais situées dans des Categorie1 différentes.