Joel Grus

Data science par la pratique

Fondamentaux avec Python

EYROLLES

Préparez-vous aux métiers du futur !

Data science par la pratique

UN OUVRAGE DE RÉFÉRENCE POUR LES (FUTURS) DATA SCIENTISTS

Les bibliothèques, les frameworks, les modules et les boîtes à outils sont parfaits pour faire de la data science. Ils sont aussi un bon moyen de plonger dans la discipline sans comprendre la data science. Dans cet ouvrage, vous apprendrez comment fonctionnent les outils et algorithmes les plus fondamentaux de la data science, en les réalisant à partir de zéro.

Si vous êtes fort en maths et que vous connaissez la programmation, l'auteur, Joel Grus, vous aidera à vous familiariser avec les maths et les statistiques qui sont au cœur de la data science et à acquérir les compétences informatiques indispensables pour démarrer comme data scientist. La profusion des données d'aujourd'hui contient les réponses aux questions que personne n'a encore pensé à poser. Ce livre vous enseigne comment obtenir ces réponses.

- Suivez un cours accéléré de Python
- Apprenez les fondamentaux de l'algèbre linéaire, des statistiques et des probabilités, et comprenez comment et quand les utiliser en data science
- Collectez, explorez, nettoyez, bricolez et manipulez les données
- Plongez dans les bases de l'apprentissage automatique
- Implémentez des modèles comme les k plus proches voisins, le Bayes naïf, les régressions linéaire ou logistique, les arbres de décision, les réseaux neuronaux et le clustering
- Explorez les systèmes de recommandation, le traitement du langage naturel, l'analyse de réseau, MapReduce et les bases de données

À qui s'adresse cet ouvrage ?

- Aux développeurs, statisticiens, étudiants et chefs de projet ayant à résoudre des problèmes de data science.
- Aux data scientists, mais aussi à toute personne curieuse d'avoir une vue d'ensemble de l'état de l'art de ce métier du futur.

Joel Grus est ingénieur logiciel chez Google. Auparavant data scientist dans plusieurs start-up, il vit aujourd'hui à Seattle et participe régulièrement à des réunions de data scientists. Il blogue occasionnellement sur joelgrus.com et tweete toute la journée via le compte @joelgrus.

Sommaire

www.editions-eyrolles.com

Joel Grus

Data science par la pratique

EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

Traduction autorisée de l'ouvrage en langue anglaise intitulé
Data Science from Scratch
de Joel Grus, ISBN : 9781491901427,
publié par O'Reilly Media.

Avant-propos

« Expert en data science » a été désigné comme « métier le plus sexy du XXIe siècle » par une journaliste qui n'avait sans doute jamais visité une caserne de pompiers. Mais il est vrai que la data science est un sujet brûlant en pleine croissance et il n'y a pas à chercher bien loin pour trouver des analystes surexcités qui prévoient que dans les dix prochaines années nous aurons besoin de milliards d'experts en plus de ceux qui existent déjà.

Mais qu'est-ce que la data science ou science des données ? Comment former des data scientists si nous ne savons pas répondre à cette question. D'après un diagramme de Venn devenu célèbre chez les spécialistes, la data science se situe à l'intersection des domaines suivants :

- compétences pointues en informatique ;
- connaissance des mathématiques et des statistiques ;
- expertise fonctionnelle.

Bien que ma première intention ait été d'écrire un livre qui couvrirait les trois, j'ai rapidement réalisé que rien que le traitement approfondi de « l'expertise fonctionnelle » nécessiterait des dizaines de milliers de pages. À ce stade, j'ai donc décidé de me concentrer sur les deux premiers points. Mon objectif est d'abord de vous aider à développer les compétences informatiques nécessaires pour démarrer en data science, et aussi de vous familiariser avec les mathématiques et les statistiques qui sont au cœur de la data science.

C'est une aspiration ambitieuse pour un livre, car la meilleure façon d'apprendre l'informatique reste la pratique. En lisant cet ouvrage, vous apprendrez à bien comprendre ma méthode de programmation qui ne sera pas forcément la meilleure pour vous. Vous apprendrez à bien comprendre quelques-uns de mes outils qui ne seront pas forcément les meilleurs dans votre propre cas. Vous apprendrez à bien comprendre comment j'aborde les problèmes de données, ce qui ne sera pas forcément la meilleure approche pour vous. Mon intention (et mon secret espoir) est que mes exemples vous inspirent pour mettre au point *votre* propre méthode. Tout le code et toutes les données de ce livre sont disponibles sur GitHub pour vous aider à démarrer.

La meilleure façon d'apprendre les mathématiques, c'est de faire des mathématiques. Ceci n'est pas un livre de mathématiques et la plupart du temps nous ne ferons pas « des maths ».

Cependant, vous ne pouvez pas réellement pratiquer la data science sans comprendre un peu les probabilités, les statistiques et l'algèbre linéaire. Cela signifie que toutes les fois que ce sera nécessaire, nous plongerons dans les équations, l'intuition mathématique, les axiomes et les versions simplifiées des grandes idées mathématiques. J'espère que vous n'aurez pas peur de plonger avec moi.

Tout au long de notre périple j'espère aussi vous faire toucher du doigt que jouer avec les données est amusant parce que, en fait, jouer avec les données est amusant ! (En particulier comparé à d'autres activités comme remplir sa déclaration de revenus ou travailler comme mineur.)

Partir de rien

Il existe pléthore de bibliothèques, frameworks, modules et boîtes à outils dédiés aux data sciences pour mettre en place les algorithmes et les techniques de data science les plus courants (et aussi les moins courants d'ailleurs). Si vous devenez data scientist, vous pénétrerez dans l'intimité de NumPy, scikit-learn, pandas, et toute une panoplie d'autres bibliothèques. Parfaites pour la data science, elles sont aussi une bonne manière de commencer en data science sans réellement comprendre la data science.

Dans cet ouvrage, nous partirons de zéro pour appréhender la data science. Nous construirons des outils et nous réaliserons des algorithmes à la main afin de mieux les comprendre. J'ai beaucoup travaillé sur la mise en œuvre des exemples pour qu'ils soient clairs, bien commentés et lisibles. La plupart du temps, les outils que nous construirons seront parfaits pour comprendre mais totalement inutilisables dans la réalité. Ils seront adaptés à de petits jeux de données mais ne résisteront pas à ceux dimensionnés à l'échelle du Web.

Tout au long du livre, je vous indiquerai les bibliothèques à employer pour appliquer ces techniques à plus grande échelle, mais nous ne nous en servirons pas ici.

Le débat fait toujours rage pour décider quel est le meilleur langage pour apprendre la data science. Beaucoup pensent que c'est le langage de programmation statistique R. (Disons-le tout net, ils ont tort.) Quelques-uns suggèrent Java ou Scala. Cependant, à mon avis, le choix de Python saute aux yeux.

Python possède plusieurs fonctionnalités qui le rendent particulièrement bien adapté pour apprendre (et pratiquer) la data science :

- il est gratuit ;
- il est assez facile à coder (et aussi à comprendre) ;
- il dispose de nombreuses bibliothèques en lien avec la data science.

J'hésite à appeler Python mon langage préféré. Il existe d'autres langages que je trouve plus agréables, mieux conçus ou simplement plus amusants à utiliser. Et pourtant, presque toutes les fois que je commence un projet de data science, je reviens finalement à Python. Chaque fois que j'ai besoin de faire un prototype rapide mais fonctionnel, je reviens à Python. Et chaque fois que je veux démontrer des concepts de data science de manière claire et limpide, je finis par utiliser Python. Ce livre fait donc appel à Python.

L'objectif de cet ouvrage n'est pas de vous apprendre Python. (Même s'il est certain qu'en lisant ce livre vous apprendrez aussi un peu Python.) Je vais vous accompagner dans un cours accéléré pendant un chapitre pour souligner les fonctionnalités les plus importantes pour nous, mais si vous ignorez tout de la programmation Python (ou de la programmation tout court), vous aurez intérêt à compléter ce livre par un manuel du genre « Python pour débutants ».

Le reste de notre introduction à la data science suivra la même voie : entrer dans les détails quand il est crucial ou particulièrement instructif d'entrer dans les détails, et à d'autres moments laisser de côté les détails pour que vous puissiez trouver vos propres solutions (ou consulter Wikipédia).

Je forme des data scientists depuis des années. Même si tous ne sont pas devenus des *data gourous* capables de changer le monde, je les ai tous quittés meilleurs data scientists que lors de notre première rencontre. Et j'en suis venu à croire que quiconque possède quelques aptitudes aux mathématiques et à la programmation a toutes les compétences nécessaires pour faire de la data science. Tout ce qu'il faut c'est un esprit avide de découverte, de la motivation, et ce livre… D'où ce livre.

Utilisation des exemples de code

Du matériel supplémentaire (exemples de code, exercices…) est disponible, en anglais, en téléchargement à l'adresse https://github.com/joelgrus/data-science-from-scratch.

Ce livre a pour objet de vous aider dans votre travail. En règle générale, vous pouvez utiliser librement le code présenté dans ces pages dans vos programmes et votre documentation. Il n'est donc pas nécessaire de nous contacter pour obtenir un accord, à moins que vous ne souhaitiez reproduire une portion significative du code. Par exemple, l'écriture d'un programme qui reprend plusieurs fragments de code tirés de ce livre ne nécessite pas d'autorisation, mais la vente ou la distribution d'un CD d'exemples tirés de cet ouvrage en exige une. Par ailleurs, si vous répondez à une question en citant ce livre et ses exemples de codes, nul besoin de demander une autorisation, mais pour incorporer une quantité significative de code tiré de ce livre dans la documentation de vos produits, il en faut une.

L'auteur apprécie d'être reconnu en tant que tel mais sans en faire une condition impérative. Cette reconnaissance prend généralement la forme suivante : titre, auteur, éditeur et numéro ISBN. Par exemple cet ouvrage, initialement publié en anglais par O'Reilly sous le titre *Data Science from Scratch* (Copyright 2015 Joel Grus, 978-1-4919-0142-7) a été traduit en français pour les éditions Eyrolles par Dominique Durand-Fleischer.

Si vous pensez que votre usage des exemples de code ne correspond pas à l'usage équitable ni à la demande d'autorisation évoqués ci-dessus, veuillez nous contacter à l'adresse : ahabian@eyrolles.com.

Remerciements

Avant tout, je tiens à remercier Mike Loukides qui a accueilli favorablement ma proposition pour ce livre (et pour son insistance à me convaincre de le réduire à une taille raisonnable). Il lui aurait été si simple de dire « Qui est ce type qui m'envoie sans arrêt des extraits de chapitres et comment puis-je m'en débarrasser ? » Je suis heureux qu'il ne l'ait pas fait. Je voudrais remercier également mon éditrice, Marie Beaugureau, qui m'a guidé tout au long du processus de publication et qui a hissé ce livre à un état bien meilleur que tout ce que j'aurais pu faire moi-même.

Je n'aurais pas pu écrire ce livre si je n'avais pas appris la data science, et je n'aurais sans doute jamais étudié la data science sans l'influence de Dave Hsu, Igor Tatarinov, John Rauser, et le reste du gang Farecast (à une époque si lointaine qu'on ne parlait pas encore de data science !). Les organisateurs de Coursera méritent mes remerciements eux aussi.

Je suis reconnaissant aux lecteurs et aux réviseurs de ma version bêta. Jay Fundling a trouvé des erreurs à la pelle et a mis le doigt sur des explications obscures, et le livre est bien meilleur (et plus correct) grâce à lui. Debashis Ghosh est un héros qui a vérifié le bien-fondé de toutes mes statistiques. Andrew Musselman a suggéré de gommer le côté « ceux qui préfèrent R à Python n'ont aucun sens moral » de ce livre, et je suis convaincu que c'était finalement un excellent conseil. Trey Causey, Ryan Matthew Balfanz, Loris Mularoni, Nuria Pujol, Rob Jefferson, Mary Pat Campbell, Zach Geary, et Wendy Grus m'ont tous donné des conseils infiniment avisés. S'il subsiste quelques erreurs, j'en assume l'entière responsabilité.

Je dois beaucoup à la communauté Twitter #datascience qui m'a fait découvrir une foule de nouveaux concepts, m'a mis en contact avec des personnes extraordinaires et m'a fait prendre conscience de mes imperfections au point que je les ai quittées et que j'ai écrit un livre pour compenser. Des remerciements spéciaux à Trey Causey (encore lui) qui m'a rappelé au passage d'inclure un chapitre sur l'algèbre linéaire, et à Sean J. Taylor, qui a pointé du doigt (en passant) quelques lacunes sérieuses dans le chapitre « Travail sur les données ».

Par-dessus tout je dois un immense merci à Ganga et à Madeline. La seule chose plus difficile que d'écrire un livre, c'est de vivre avec quelqu'un qui écrit un livre. Je n'aurais jamais pu achever cet ouvrage sans leur soutien.

Table des matières

1

Introduction

« Des données ! Des données ! Donnez-moi des données ! » s'écria-t-il avec impatience.
« Je ne peux pas faire de briques sans argile ! »
- Arthur Conan Doyle

L'origine des données

Nous vivons dans un monde noyé par les données. Les sites web suivent à la trace chaque clic de chaque utilisateur. Tous les jours, votre smartphone enregistre votre localisation et votre vitesse à la seconde près. Des « accros de la donnée » portent des podomètres gonflés aux stéroïdes pour enregistrer en permanence leur rythme cardiaque, leurs gestes, ce qu'ils mangent et comment ils dorment. Les voitures intelligentes collectent les habitudes de conduite, les maisons intelligentes collectent les habitudes de vie et les marqueteurs intelligents collectent les habitudes d'achat. Internet lui-même n'est qu'un énorme diagramme de connaissances qui contient (entre autres) une gigantesque encyclopédie de références croisées ; des bases de données spécialisées sur le cinéma, la musique, les résultats sportifs, les flippers de bistrot, les mèmes et les cocktails ; et trop de statistiques officielles (certaines d'entre elles presque vraies !) publiées par trop de gouvernements pour alimenter la réflexion.

Enfouies dans ces données se cachent les réponses aux innombrables questions que personne n'a jamais pensé à poser. Nous allons apprendre à les trouver grâce à ce livre.

Qu'est-ce que la data science ?

Connaissez-vous la blague pour définir un data scientist (autrement dit, un expert en data science) ? C'est quelqu'un qui s'y connaît mieux en statistiques qu'un informaticien et mieux en informatique qu'un statisticien – je n'ai pas dit que c'était drôle… Dans les faits, certains data scientists sont, pour des raisons pratiques, des statisticiens, alors que d'autres se confondent avec des ingénieurs en informatique. Certains sont des experts de l'apprentissage automatique, et d'autres seraient incapables de programmer l'itinéraire de sortie du jardin d'enfants. Certains sont titulaires de doctorats et collectionnent les publications tandis que d'autres n'ont jamais lu une thèse universitaire de leur vie (honte à eux quand même). En résumé, quelle que soit la définition que vous donnez de la data science, vous trouverez des spécialistes auxquels elle ne s'applique absolument pas !

Mais ce n'est pas ce qui va nous empêcher d'essayer. Nous dirons qu'un data scientist est une personne capable d'extraire du sens d'un ramassis de données inextricables. Le monde d'aujourd'hui est rempli de gens qui tentent de donner du sens aux données.

Par exemple, le site de rencontre OkCupid pose à ses membres des tonnes de questions pour trouver le partenaire qui leur correspond le mieux. Mais il analyse aussi ces résultats pour imaginer des questions innocentes qui vous permettront de connaître vos chances de coucher avec elle ou lui dès le premier rendez-vous.

Facebook vous demande votre ville natale et celle où vous vivez actuellement, officiellement pour permettre à vos amis de vous trouver plus facilement. Mais il utilise ces données pour repérer des schémas de déplacement des populations et savoir où vivent les fans des différentes équipes de football.

Certaines chaînes de supermarchés suivent vos achats et vos interactions, aussi bien en ligne que dans leurs magasins. Elles utilisent ces données pour construire un modèle prédictif, comme savoir quelles clientes sont enceintes pour mieux mettre en valeur les articles de puériculture auprès d'elles.

En 2012, Obama a employé pour sa campagne électorale des douzaines de data scientists pour explorer les bases de données d'électeurs dans le but de trouver ceux qui devaient être particulièrement soignés afin d'orienter les appels de levées de fonds vers certains donateurs et de concentrer les efforts là où ils seraient le plus utiles. Il est généralement admis que tous ces efforts ont joué un rôle non négligeable dans la réélection du président, ce qui permet d'affirmer que les futures campagnes seront de plus en plus pilotées par les données, avec pour résultat une course aux armements sans fin pour la collecte et l'analyse des données.

Rassurez-vous, certains data scientists mettent de temps en temps leurs compétences au service de la bonne cause, pour rendre les gouvernements plus efficaces, aider les sans-abris ou améliorer la santé publique. Mais votre carrière ne s'en portera pas plus mal si vous préférez imaginer comment inciter les gens à cliquer sur des publicités.

L'hypothèse DataSciencester

Félicitations ! Vous venez d'être recruté pour conduire les efforts en faveur de la data science chez DataSciencester, le réseau social des experts de la data science.

Malgré son public cible, DataSciencester n'a jamais rien investi pour développer sa propre pratique de data science. (Pour être tout à fait honnête, DataSciencester n'a pas non plus investi dans l'élaboration de son produit.) Ce sera votre mission ! Tout au long de ce livre, vous allez découvrir les concepts de la data science en trouvant les solutions aux problèmes que vous rencontrez au travail. Parfois nous examinerons des données explicitement fournies par des utilisateurs, parfois nous observerons des données générées au moyen de leurs interactions avec le site, et parfois nous examinerons des données issues des expériences que nous aurons conçues.

Et comme DataSciencester a une très forte culture hostile à ce qui « ne vient pas de chez nous », nous construirons nos propres outils à partir de zéro. À la fin, vous aurez acquis une compréhension solide des bases de la data science. Et vous aurez davantage d'assurance pour mettre vos compétences au service d'une entreprise ou pour résoudre tout autre problème qui vous intéresse.

C'est parti, bienvenue et bonne chance ! (Vous pouvez venir en jeans le vendredi, et les toilettes sont à droite au fond du couloir.)

À la recherche des connecteurs clés

C'est votre premier jour chez DataSciencester et déjà le responsable Réseautage a des tonnes de questions concernant vos utilisateurs. Jusqu'à présent, il n'avait aucun interlocuteur. Il est donc tout excité de votre recrutement.

En particulier, il veut que vous identifiiez les « connecteurs clés » parmi les experts de data science. Pour ce faire, il vous remet le contenu complet de tout le réseau DataSciencester. (Dans la vraie vie, il est rare que quelqu'un vous remette ainsi les données qui vous sont nécessaires. Le chapitre 9 traite de la collecte des données.)

À quoi ressemble cette extraction de données ? C'est une liste d'utilisateurs, chacun représenté par un dict contenant l'identifiant (`id`, au format numérique), et le nom de l'utilisateur (qui, sous l'effet d'une de ces coïncidences cosmiques extraordinaires, rime avec l'identifiant en anglais) :

```
users = [
{ "id": 0, "name": "Hero" },
{ "id": 1, "name": "Dunn" },
{ "id": 2, "name": "Sue" },
{ "id": 3, "name": "Chi" },
{ "id": 4, "name": "Thor" },
{ "id": 5, "name": "Clive" },
{ "id": 6, "name": "Hicks" },
```

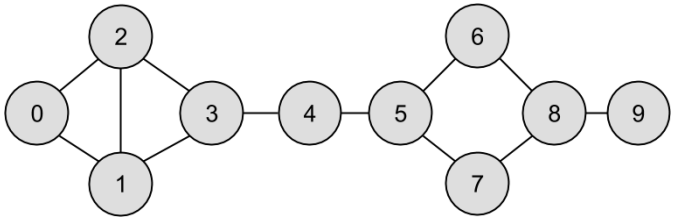
```
{ "id": 7, "name": "Devin" },
{ "id": 8, "name": "Kate" },
{ "id": 9, "name": "Klein" }
]
```

Il vous remet aussi les « relations », représentées par une liste de paires d'`id` :

```
friendships = [(0, 1), (0, 2), (1, 2), (1, 3), (2, 3), (3, 4),
(4, 5), (5, 6), (5, 7), (6, 8), (7, 8), (8, 9)]
```

Par exemple, le tuple `(0, 1)` indique que le data scientist identifié par id 0 (Hero) et le data scientist identifié par id 1 (Dunn) sont amis. La figure 1-1 illustre le réseau.

Figure 1–1
Le réseau DataSciencester



Comme nous représentons nos utilisateurs comme des dict, il est facile de les enrichir de données supplémentaires.

Note

Ne vous attardez pas trop sur le détail du code à ce stade. Vous aurez droit à un cours accéléré de Python au chapitre 2. Pour le moment, essayez simplement d'avoir une vue d'ensemble de la chose.

Par exemple, si nous voulons ajouter une liste d'amis à chaque utilisateur, il faut d'abord assigner à l'attribut `friends` de chaque utilisateur une liste vide :

```
for user in users:
    user["friends"] = []
```

Et ensuite, il faut remplir la liste à partir des données de l'objet `friendships` :

```
for i, j in friendships:
    # possible parce que user[i] est l'utilisateur dont l'id est i
    users[i]["friends"].append(users[j])  # ajoute j comme ami de i
    users[j]["friends"].append(users[i])  # ajoute i comme ami de j
```

Une fois que chaque dict contient sa liste d'amis, il est facile pour nous de répondre aux questions comme « Quel est le nombre moyen de connexions ? »

Cherchons d'abord le nombre total de connexions en faisant la somme le long de toutes les listes d'amis :

```
def number_of_friends(user):
    """combien d'amis l'utilisateur _user_ a-t-il ?"""
    return len(user["friends"])  # longueur de la liste  friend_ids

total_connections = sum(number_of_friends(user)
              for user in users)  # 24
```

Il suffit ensuite de diviser par le nombre total d'utilisateurs :

```
from __future__ import division  # la division entière est bancale
num_users = len(users)           # longueur de la liste des utilisateurs
avg_connections = total_connections / num_users  # 2.4
```

Il est tout aussi facile de trouver les personnes les plus connectées : ce sont les personnes qui ont le plus grand nombre d'amis.

Comme il n'y a pas beaucoup d'utilisateurs, nous pouvons les trier depuis « le plus d'amis » jusqu'à « le moins d'amis » :

```
# créer une liste (user_id, number_of_friends)
num_friends_by_id = [(user["id"], number_of_friends(user))
      for user in users]
sorted(num_friends_by_id,  # la trier
  key=lambda (user_id, num_friends): num_friends,  # suivant num_friends
  reverse=True)  # du plus grand au plus petit

# chaque paire vaut (user_id, num_friends)
# [(1, 3), (2, 3), (3, 3), (5, 3), (8, 3),
# (0, 2), (4, 2), (6, 2), (7, 2), (9, 1)]
```

Pour interpréter ce que nous venons de faire, on peut considérer que c'est une façon d'identifier les personnes centrales du réseau. En fait, ce que nous avons calculé est le degré de centralité du réseau (figure 1-2).

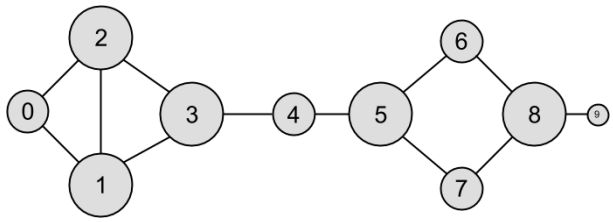


Figure 1–2
Le réseau DataSciencester dimensionné par degré

L'avantage est que c'est très facile à calculer, mais le résultat n'est pas toujours celui qui est attendu. Par exemple, dans le réseau DataSciencester, Thor (id 4) a seulement deux connexions alors que Dunn (id 1) en a trois. Et pourtant, quand on regarde le réseau, on a intui-

tivement l'impression que Thor est plus central. Dans le chapitre 21, nous examinerons les réseaux plus en détail et nous étudierons des notions de centralité plus complexes, plus ou moins en accord avec nos intuitions.

Des experts que vous connaissez peut-être

Alors que vous êtes en train de remplir votre dossier de nouvel embauché, la responsable de la Fraternisation débarque dans votre bureau. Elle veut encourager davantage de connexions entre vos adhérents. Elle vous demande donc de concevoir un système pour suggérer « des experts que vous connaissez peut-être ».

Votre première idée est de suggérer qu'un utilisateur peut connaître les amis de ses amis. C'est facile à calculer : pour chacun des amis d'un utilisateur, il suffit de naviguer parmi les amis de cette personne et collecter les résultats :

```
def friends_of_friend_ids_bad(user):
  # "foaf" est l'abréviation de  "friend of a friend", autrement dit ami d'un ami
  return [foaf["id"]
      for friend in user["friends"]       # pour chaque ami de l'utilisateur
      for for foaf in friend["friends"]]  # obtenir chacun de leurs amis _their_ friends
```

Quand nous appelons cette fonction sur `users[0]` (Hero), on récupère :

```
[0, 2, 3, 0, 1, 3]
```

Ce qui inclut user `0` (deux fois), car Hero est ami avec ses deux amis. Le résultat inclut donc les utilisateurs `1` et `2` bien qu'ils soient déjà amis avec Hero. Et il inclut deux fois l'utilisateur `3`, car Chi peut être atteint par l'intermédiaire de deux amis différents :

```
print [friend["id"] for friend in users[0]["friends"]]  # [1, 2]
print [friend["id"] for friend in users[1]["friends"]]  # [0, 2, 3]
print [friend["id"] for friend in users[2]["friends"]]  # [0, 1, 3]
```

Savoir que des personnes sont des amis d'amis de plusieurs manières semble être une information intéressante, donc nous devrions plutôt compter les amis communs. Et nous devrions surtout utiliser une fonction supplémentaire pour exclure les personnes déjà connues de l'utilisateur :

```
from collections import Counter  # n'est pas chargé par défaut

def not_the_same(user, other_user):
  """deux utilisateurs ne sont pas le même s'ils ont des identifiants différents"""
  return user["id"] != other_user["id"]
```

```
def not_friends(user, other_user):
  """l'autre utilisateur n'est pas un ami s'il n'est pas dans  user["friends"];
  C'est-à-dire, s'il n'est pas le même que tous ceux présents dans  user["friends"]"""
  return all(not_the_same(friend, other_user)
      for friend in user["friends"])

def friends_of_friend_ids(user):
  return Counter(foaf["id"]
    for friend in user["friends"]               # pour chacun de mes amis
    for foaf in friend["friends"]               # compter *leurs* amis
    if not_the_same(user, foaf)                 # ceux qui ne sont pas moi
                  and not_friends(user, foaf))  # et ne sont pas mes amis

print friends_of_friend_ids(users[3])           # compteur({0: 2, 5: 1})
```

Ceci met en évidence que Chi (id 3) a deux amis communs avec Hero (id 0), mais seulement un avec Clive (id 5).

En tant que data scientist, vous aimeriez aussi rencontrer des utilisateurs qui partagent vos centres d'intérêt. (Un bon exemple de l'aspect « domaines de prédilection » de la data science.) Après avoir demandé autour de vous, vous avez réussi à mettre la main sur ces données, sous la forme d'une liste de paires `(user_id, interest)` :

```
interests = [
(0, "Hadoop"), (0, "Big Data"), (0, "HBase"), (0, "Java"),
(0, "Spark"), (0, "Storm"), (0, "Cassandra"),
(1, "NoSQL"), (1, "MongoDB"), (1, "Cassandra"), (1, "HBase"),
(1, "Postgres"), (2, "Python"), (2, "scikit-learn"), (2, "scipy"),
(2, "numpy"), (2, "statsmodels"), (2, "pandas"), (3, "R"), (3, "Python"),
(3, "statistics"), (3, "regression"), (3, "probability"),
(4, "machine learning"), (4, "regression"), (4, "decision trees"),
(4, "libsvm"), (5, "Python"), (5, "R"), (5, "Java"), (5, "C++"),
(5, "Haskell"), (5, "programming languages"), (6, "statistics"),
(6, "probability"), (6, "mathematics"), (6, "theory"),
(7, "machine learning"), (7, "scikit-learn"), (7, "Mahout"),
(7, "neural networks"), (8, "neural networks"), (8, "deep learning"),
(8, "Big Data"), (8, "artificial intelligence"), (9, "Hadoop"),
(9, "Java"), (9, "MapReduce"), (9, "Big Data")
]
```

Par exemple, Hero (id 0) n'a aucun ami commun avec Klein (id 9), mais ils partagent un intérêt pour l'apprentissage automatique.

Il est facile de construire une fonction qui trouve les utilisateurs partageant un sujet donné :

```
def data_scientists_who_like(target_interest):
    return [user_id
        for user_id, user_interest in interests
        if user_interest == target_interest]
```

Ça marche, mais il faut passer en revue la liste complète des centres d'intérêt à chaque recherche. Si nous avons beaucoup d'utilisateurs et de centres d'intérêt (ou si nous voulons effectuer beaucoup de recherches), il sera sans doute préférable de construire un index des centres d'intérêt par utilisateur :

```
from collections import defaultdict

# les clés sont les centres d'intérêt, les valeurs sont les listes d'identifiants user_id
# avec ce centre d'intérêt
user_ids_by_interest = defaultdict(list)

for user_id, interest in interests:
    user_ids_by_interest[interest].append(user_id)
```

et un autre à partir des utilisateurs vers les centres d'intérêt :

```
# les clés sont les user_ids, les valeurs sont les listes de centres d'intérêt
# pour cet identifiant d'utilisateur
interests_by_user_id = defaultdict(list)

for user_id, interest in interests:
  interests_by_user_id[user_id].append(interest)
```

Maintenant, il est facile de trouver qui a le plus de centres d'intérêt en commun avec un utilisateur donné :

- faire des itérations sur les centres d'intérêt de l'utilisateur ;
- pour chacun des centres d'intérêt, explorer les autres utilisateurs avec ce centre d'intérêt ;
- compter combien de fois nous voyons les autres utilisateurs.

```
def most_common_interests_with(user):
    return Counter(interested_user_id
      for interest in interests_by_user_id[user["id"]]
      for interested_user_id in user_ids_by_interest[interest]
      if interested_user_id != user["id"])
```

Nous pourrons alors utiliser ces résultats pour construire une fonctionnalité plus riche des « experts que vous pourriez connaître » basée sur une combinaison d'amis communs et de centres d'intérêt communs. Nous explorerons ce type d'application au chapitre 22.

Salaires et expérience

Vous êtes prêt à sortir pour le déjeuner quand le responsable des Relations publiques vous demande si vous pouvez illustrer de manière ludique la question du salaire d'un expert en data science. Les salaires sont des données sensibles, mais il a réussi à vous procurer un jeu de données anonymes contenant le salaire de chaque utilisateur (en dollars) et son ancienneté au poste de data scientist (en années) :

```
salaries_and_tenures = [(83000, 8.7), (88000, 8.1),
                        (48000, 0.7), (76000, 6),
                        (69000, 6.5), (76000, 7.5),
                        (60000, 2.5), (83000, 10),
                        (48000, 1.9), (63000, 4.2)]
```

La première étape consiste naturellement à représenter les données sur un graphique (nous verrons comment procéder au chapitre 3). Vous pouvez découvrir le résultat obtenu sur la figure 1-3.

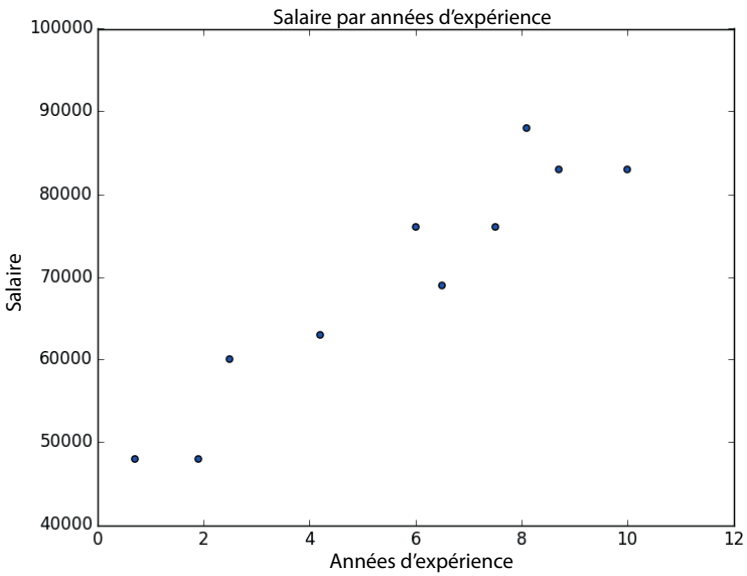


Figure 1–3
Salaire par années d'expérience

Il est évident que les personnes les plus expérimentées ont tendance à gagner davantage. Comment illustrer cette réalité ? Votre première idée est de regarder le salaire moyen par nombre d'années d'ancienneté :

```
# les clés sont les années, les valeurs sont les listes de salaires pour chaque période
# d'affectation
salary_by_tenure = defaultdict(list)

for salary, tenure in salaries_and_tenures:
  salary_by_tenure[tenure].append(salary)

# les clés sont les années, chaque valeur est le salaire moyen pour cette période
average_salary_by_tenure = {
  tenure : sum(salaries) / len(salaries)
  for tenure, salaries in salary_by_tenure.items()
}
```

Finalement, ce n'est pas très utile, car aucun des utilisateurs n'a la même ancienneté et le rapport revient à énumérer simplement le salaire de chacun.

```
{0.7: 48000.0,
1.9: 48000.0,
2.5: 60000.0,
4.2: 63000.0,
6: 76000.0,
6.5: 69000.0,
7.5: 76000.0,
8.1: 88000.0,
8.7: 83000.0,
10: 83000.0}
```

Il serait sans doute plus intéressant de répartir l'ancienneté en catégories :

```
def tenure_bucket(tenure):
    if tenure < 2:
      return "less than two"
    elif tenure < 5:
      return "between two and five"
    else:
      return "more than five"
```

ensuite de regrouper les salaires par catégories :

```
# les clés sont les catégories de postes, les valeurs sont les listes de salaires
# pour cette catégorie
salary_by_tenure_bucket = defaultdict(list)

for salary, tenure in salaries_and_tenures:
  bucket = tenure_bucket(tenure)
  salary_by_tenure_bucket[bucket].append(salary)
```

et enfin, de calculer le salaire moyen de chaque catégorie :

```
# les clés sont les catégories de poste, les valeurs sont le salaire moyen
# pour cette catégorie
average_salary_by_bucket = {
  tenure_bucket : sum(salaries) / len(salaries)
  for tenure_bucket, salaries in salary_by_tenure_bucket.iteritems()
}
```

ce qui est plus intéressant :

```
{'between two and five': 61500.0,
 'less than two': 48000.0,
 'more than five': 79166.66666666667}
```

Et voilà ce que vous annoncez : « Les data scientists ayant plus de cinq ans d'expérience gagnent 65 % de plus que ceux qui ont peu ou pas d'expérience. »

Mais nous avons choisi les catégories de manière totalement arbitraire. Nous préférerions dégager une règle sur l'effet que peut avoir une année d'ancienneté supplémentaire sur le salaire (en moyenne). En plus de nous procurer une annonce plus percutante, cela nous permettrait de faire des prévisions sur les salaires que nous ne connaissons pas. Nous développerons cette idée au chapitre 14.

Les comptes payants

De retour à votre bureau, vous trouvez la responsable Recettes qui vous attend. Elle voudrait mieux appréhender quels utilisateurs paient pour avoir un compte et lesquels ne paient pas. (Elle connaît leurs noms, mais ce n'est pas une information très facile à exploiter.)

Vous remarquez qu'il semble exister une correspondance entre les années d'expérience et les comptes payants :

```
0.7 paid
1.9 unpaid
2.5 paid
4.2 unpaid
6   unpaid
6.5 unpaid
7.5 unpaid
8.1 unpaid
8.7 paid
10  paid
```

Les utilisateurs qui ont très peu ou beaucoup d'années d'expérience ont tendance à payer, alors que les utilisateurs avec une durée d'expérience moyenne ne paient pas.

Donc, si vous voulez modéliser la situation – même s'il n'y a vraiment pas assez de données pour établir un modèle – vous pourriez être tenté de prédire que les utilisateurs peu ou très expérimentés sont « payants » et que les utilisateurs d'expérience moyenne sont « non payants ».

```
def predict_paid_or_unpaid(years_experience):
  if years_experience < 3.0:
    return "paid"
  elif years_experience < 8.5:
    return "unpaid"
  else:
    return "paid"
```

Évidemment, nous avons complètement déduit les seuils à l'œil nu.

Avec davantage de données (et plus de maths), nous pourrions construire un modèle prédictif pour connaître la probabilité qu'un utilisateur paie en nous basant sur son nombre d'années d'expérience. Nous étudierons ce genre de problème au chapitre 16.

Les centres d'intérêt

Alors que votre premier jour touche à sa fin, la responsable Stratégie de contenu voudrait savoir quels sujets intéressent le plus les utilisateurs, afin de pouvoir planifier ses publications sur le blog en conséquence. Vous avez déjà les données brutes du projet précédent dont le but était de suggérer des experts qui pourraient être vos amis :

```
interests = [
  (0, "Hadoop"), (0, "Big Data"), (0, "HBase"), (0, "Java"),
  (0, "Spark"), (0, "Storm"), (0, "Cassandra"),
  (1, "NoSQL"), (1, "MongoDB"), (1, "Cassandra"), (1, "HBase"),
  (1, "Postgres"), (2, "Python"), (2, "scikit-learn"), (2, "scipy"),
  (2, "numpy"), (2, "statsmodels"), (2, "pandas"), (3, "R"), (3, "Python"),
  (3, "statistics"), (3, "regression"), (3, "probability"),
  (4, "machine learning"), (4, "regression"), (4, "decision trees"),
  (4, "libsvm"), (5, "Python"), (5, "R"), (5, "Java"), (5, "C++"),
  (5, "Haskell"), (5, "programming languages"), (6, "statistics"),
  (6, "probability"), (6, "mathematics"), (6, "theory"),
  (7, "machine learning"), (7, "scikit-learn"), (7, "Mahout"),
  (7, "neural networks"), (8, "neural networks"), (8, "deep learning"),
  (8, "Big Data"), (8, "artificial intelligence"), (9, "Hadoop"),
  (9, "Java"), (9, "MapReduce"), (9, "Big Data")
]
```

Une manière simple (mais pas très excitante) de trouver les sujets les plus populaires serait de compter les mots :

1. Passer chaque sujet en minuscules (car différents utilisateurs ont pu utiliser ou pas des majuscules).
2. Isoler les mots de chaque sujet.
3. Compter les résultats.

Soit sous forme de code :

```
words_and_counts = Counter(word
                           for user, interest in interests
                           for word in interest.lower().split())
```

Cela permet, par exemple, de facilement identifier les mots qui ont plus d'une occurrence :

```
for word, count in words_and_counts.most_common():
    if count > 1:
        print word, count
```

Ce qui vous donne le résultat prévu (à moins que vous ne vous attendiez à séparer en deux mots « scikit-learn », auquel cas vous n'obtiendrez pas le résultat attendu) :

```
learning 3
java 3
python 3
big 3
data 3
hbase 2
regression 2
cassandra 2
statistics 2
probability 2
hadoop 2
networks 2
machine 2
neural 2
scikit-learn 2
r 2
```

Nous examinerons des méthodes d'extraction plus élaborées au chapitre 20.

À suivre

Ce premier jour s'est plutôt bien passé ! Épuisé, vous vous glissez hors de l'immeuble avant que quelqu'un ne vienne encore vous solliciter. Reposez-vous bien, car demain, ce sera la journée d'intégration des nouveaux employés. (Oui, vous venez de vivre une journée de travail bien remplie *avant même* la journée d'intégration des nouveaux employés. Il faudra en parler au DRH !)