Introduction au Text Processing

Traitement et analyse de données textuelles

Ce dont on va parler

- Enjeux et applications du text processing
- Pre-processing de la donnée
- Les principaux algorithmes de text processing
- Hands' On !

Text Processing

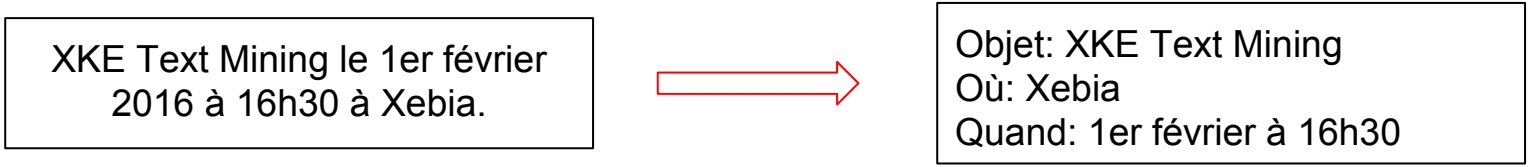
Enjeux et Principales Applications

Introduction au Natural Language Processing

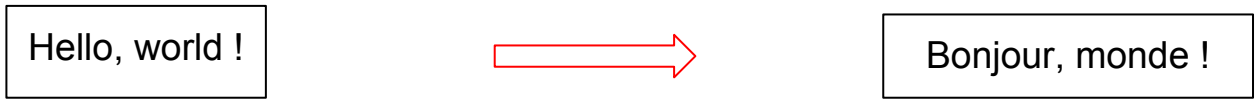
- Sous-catégorie liée à l'Intelligence Artificielle
- Découverte et Extraction **automatique** d'information **porteuse de sens** à partir de données textuelles
- Types d'information
  - Numérique
    - Les principaux algorithmes (ML ou autre) prennent comme entrée des données numériques
    - Besoin de caractériser numériquement des textes
  - Textuelle
    - Résumé, traduction, sémantique, etc.

Principales applications

- Information Extraction



- Machine Translation



Principales applications

❖ Text Summarization

Vous avez aimé l'Open XKE, vous adorerez la XebiCon ! Rendez-vous le 4 novembre prochain à l'Eurosites George V, à Paris, pour parler du futur du numérique. En voici un avant-goût : Les participants pourront assister à une keynote d'exception sur le thème Les transformations digitales, les enjeux d'un grand groupe par Yves Caseau, le Directeur de la Digital Agency d' Axa. Il s'ensuivra 27 conférences triées sur le volet, 10 témoignages client et 40 experts à vos disposition lors de l' Expert café.

Xebicon le 4 novembre à Paris avec une keynote sur les transformations digitales, des conférences et des experts café.

❖ Sentiment Analysis

J'adore la nouvelle Apple Watch, elle est trop belle !

"La nouvelle Apple Watch, un produit révolutionnaire" .. qu'est-ce qu'il faut pas entendre..

Principales applications

- Analyse sémantique

Aujourd'hui	est	une	belle	journée
Nom	Verbe	Article	Adjectif	Nom

- Segmentation

Text Processing

Preprocessing de la donnée

Preprocessing de la donnée textuelle

- Nécessité de traiter et nettoyer la donnée avant de l’ exploiter
  - Ponctuation
  - Caractères spéciaux
  - Fautes d’orthographe
  - Stop-Words
  - Splitting

- Objectif: Représenter chaque texte en une liste de mots nettoyés et filtrés

Quelques actions

- **Tokenizing**
  - Transformation d'une phrase/texte en une liste de mots
- **Filtering**
  - Filtrage des stopwords et de la ponctuation
- **Stemming**
  - Transformer un mot en sa racine
    - chiens -> chien
    - étaient -> être

Text Processing

Quelques algorithmes

Transformation de la donnée en caractéristiques numériques

Text Processing

TF-IDF

Term Frequency - Bag of Words

- Mapping d'une séquence de termes à leurs fréquences dans un document
  - Plus un terme est fréquent dans le document, plus il a de chance d'être révélateur de l'information contenue dans celui-ci
  - A l'exception des "stop words"

```
Row(Name=u'Braund, Mr. Owen Harris',
    name_TF=SparseVector(20, {1: 1.0, 3: 1.0, 9: 1.0, 14: 1.0})),
Row(Name=u'Cumings, Mrs. John Bradley (Florence Briggs Thayer)',
    name_TF=SparseVector(20, {2: 1.0, 4: 2.0, 8: 1.0, 11: 1.0, 13: 1.0, 16: 1.0})),
Row(Name=u'Heikkinen, Miss. Laina',
    name_TF=SparseVector(20, {7: 1.0, 10: 1.0, 18: 1.0}))
```

Inverse Document Frequency

- Mapping d'une séquence de termes à l'inverse de leur fréquence au sein du corpus de documents
  - Si un terme apparaît dans la plupart des documents du corpus, il y a peu de chance qu'il soit utile pour les distinguer et les classer

```
Row(Name=u'Braund, Mr. Owen Harris',
    name_TF_IDF=SparseVector(20, {1: 1.7456, 3: 1.9924, 9: 0.4988, 14: 2.0904})),
Row(Name=u'Cumings, Mrs. John Bradley (Florence Briggs Thayer)',
    name_TF_IDF=SparseVector(20, {2: 2.0904, 4: 2.533, 8: 1.6849, 11: 1.4977, 13: 1.9532, 16: 1.7148})),
Row(Name=u'Heikkinen, Miss. Laina',
    name_TF_IDF=SparseVector(20, {7: 1.9532, 10: 1.1281, 18: 1.9792}))
```

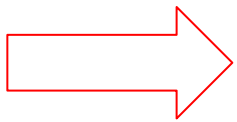
Text Processing

Word2Vec

Word2Vec

- Word2Vec est un **réseau de neurones** adapté aux données textuelles.
  - Input: Un corpus de textes splittés en listes de tokens
  - Ouput: Un dictionnaire contenant tous les mots du corpus avec un vecteur de taille fixe associé

```
u'proposant', u'partie', u'code', u'affecter',
u'exemple', u'suivant', u'demande', u'est',
u'amount', u'amount', u'format', u'souhaite',
u'affecter', u'variable', u'mode', u'seection',
u'valable', u'refactoring', u'cr\xe9ation',
u'une', u'methode', u'raccourci', u'alt',
u'description', u'refactorez', u'code',
u'facilement', u'garder', u'lisible', u'cette',
u'fonctionnalite', u'rend'
```

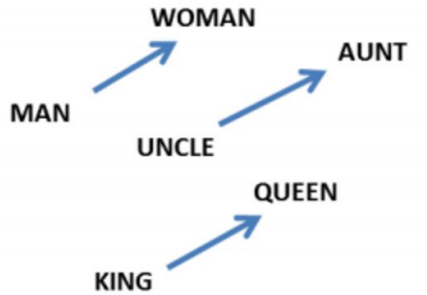


```
+--------+--------------------+
|    word|              vector|
+--------+--------------------+
|     dns|[0.09927792102098...|
| speaker|[-0.1568965017795...|
|  assert|[-0.3278719782829...|
|    mise|[0.48140543699264...|
|  plugin|[0.35941249132156...|
| retenir|[0.14124287664890...|
|    afin|[0.14102584123611...|
|   alpha|[-0.2422264367341...|
|demandes|[-0.5457851290702...|
| cyrille|[0.03088080696761...|
+--------+--------------------+
```

Word2Vec

- **Objectif**: Synonymes avec des coordonnées vectorielles proches, à partir de leurs apparutions passées

Association de mots



From *Mikolov et al.* (2013a)

V(King) - V(Man) + V(Woman) ≈ V(Queen)

Synonymes de “xebia”

```
+---------+------------------+
|     word|        similarity|
+---------+------------------+
|     chez|1.4104489993173668|
|    étage| 1.404181847473126|
|boulevard|1.3421606836751119|
|   meetup|1.3311177576813011|
|   locaux|1.3225492311830636|
|    paris|1.3178619326599148|
|  xebians|1.2589668499166167|
|haussmann|1.2585627428152393|
|directeur| 1.245307132326901|
| emmanuel|1.2434998308862675|
+---------+------------------+
```

Word2Vec - 2 méthodes d'entraînement

Continuous Bag of Words

Maximise la probabilité conditionnelle du mot étant donné son contexte

P(W(i) | W(i-2), W(i-1), W(i+1), W(i+2))

Skip-Gram

Maximise la probabilité conditionnelle du contexte étant donné le mot

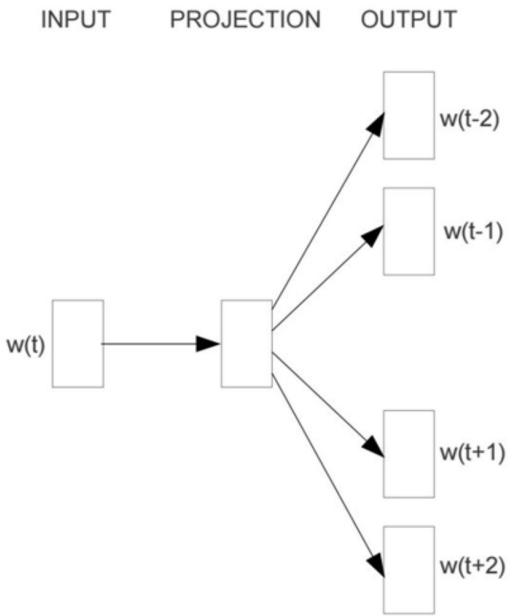
P(W(i-2), W(i-1), W(i+1), W(i+2) | W(i))

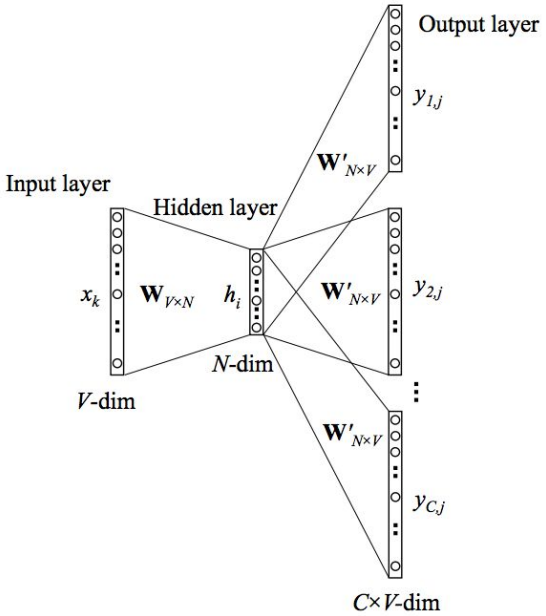
"xebia accueille le prochain **???** du scala user group jeudi soir"

meetup: 0.743
barbecue: 0.232
match: 0.025

Word2Vec - Skip Gram

- Classification supervisée
- Les vecteurs finaux correspondent aux coefficients appris dans la couche cachée du réseau de neurones

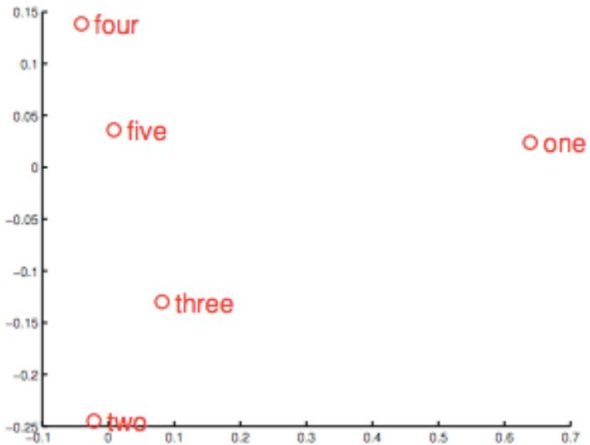


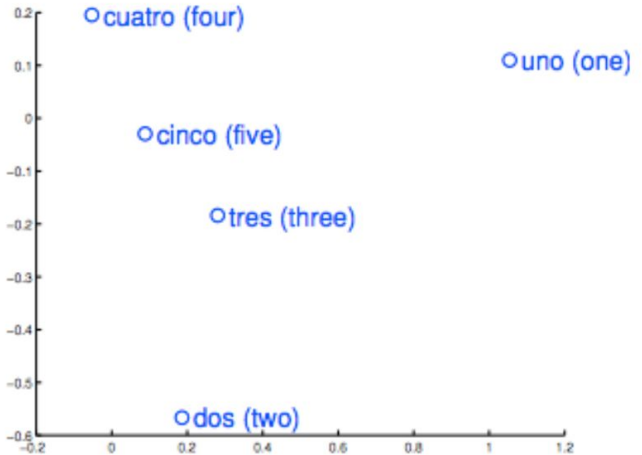


Word2Vec - Avantages

- Les techniques classiques on de nombreux défauts
  - Vecteurs sparses
  - Pas d'utilisation de la relation sémantique entre mots
- Word2Vec repose sur l'intuition que **la signification d'un mot est représentée par son contexte**

Machine Translation





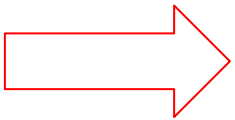
Text Processing

Latent Dirichlet Allocation

Latent Dirichlet Allocation

- Découverte automatique de **topics** contenus dans les phrases explorées

```
1 - "J'aime manger des brocolis et des bananes."
2 - "J'ai mangé une banane et un smoothie à la fraise pour le petit déjeuner."
3 - "Les chiens et les chats sont mignons."
4 - "Ma soeur a adopté un chat hier."
5 - "Regarde ce hamster mignon en train de manger un brocolli."
```



```
1 - 100% Topic A
2 - 100% Topic A
3 - 100% Topic B
4 - 100% Topic B
5 - 60% Topic A, 40% Topic B

Topic A: 30% broccoli, 15% bananes, 10% petit déjeuner
Topic B: 20% chat, 20% chien, 13% hamster, …

=> Topic A = Nourriture
=> Topic B = Animaux
```

LDA - Fonctionnement

- Documents représentés comme une **combinaison de topics**
  - Hypothèses sur la production d'un document:
    - Choix de son nb de mots N
    - Choisir une combinaison de K topics pour le document (⅓ A, ⅔ B par exemple)
    - Génération des mots
      - Choix du topic
      - Générer un mot du topic
- **Objectif**: Retrouver à partir des documents le set de topics qui ont généré ces documents de la manière la plus probable

LDA - Apprentissage

1. Assigner aléatoirement à chaque mot de chaque document un topic t
2. Pour chaque document d
   a. Pour chaque mot m du document
      i. Pour chaque topic,
      ii. P(t | d) = Prop mots appartenant à t dans le document
      iii. P(m | t) = Prop d'assignements à t qui viennent de m sur tous les documents
      iv. Réassigner un nouveau topic t à m, avec une probabilité de P(t | d) * P(m | t)
3. Réitérer

Text Mining

Hands’ On !

Hands' On

```
// Cloner le répertoire Github
git clone https://github.com/ybenoit/xke-text-mining.git

// En Python  => notebook text-mining-exercise ou text-mining-exercise-spark-user
cd /your/path/xke-text-mining/
IPYTHON_OPTS="notebook" pyspark --master local[*] --packages com.databricks:spark-csv_2.10:
1.1.0

// En Scala => notebook text-mining-exercise ou text-mining-exercise-spark-user
cd /your/path/xke-text-mining/scala/
docker build -t scala-spark-notebook .
docker run -d -p 9000:9000 -v /your/absolute/path/xke-text-mining/:/opt/docker/notebooks/
scala-spark-notebook

// A partir d'une image docker enregistrée (clé USB)
docker load < spark-notebook.tar
docker run -d -p 9000:9000 -v /your/absolute/path/xke-text-mining/:/opt/docker/notebooks/
scala-spark-notebook
```

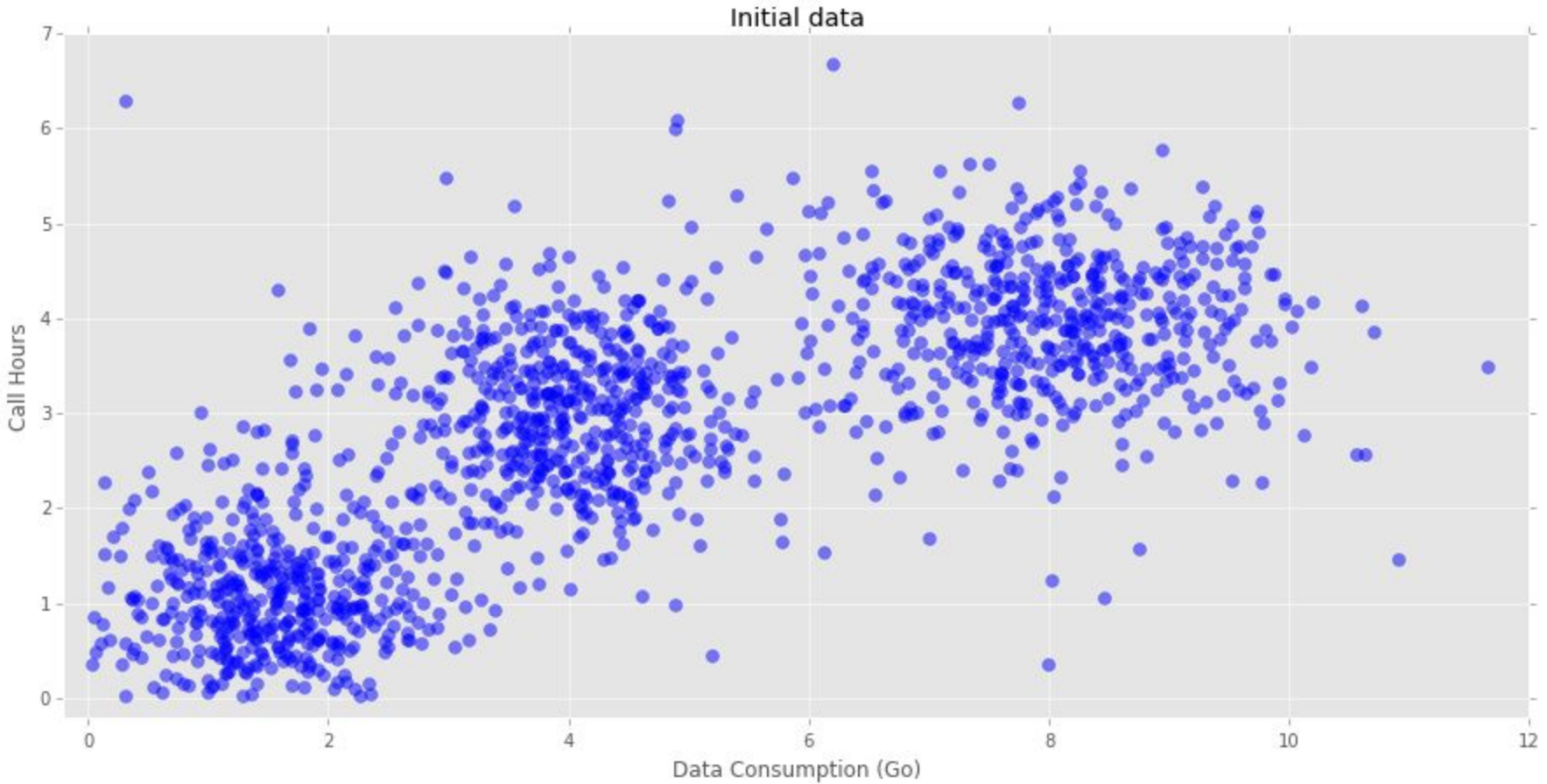
Text Mining

Annexes

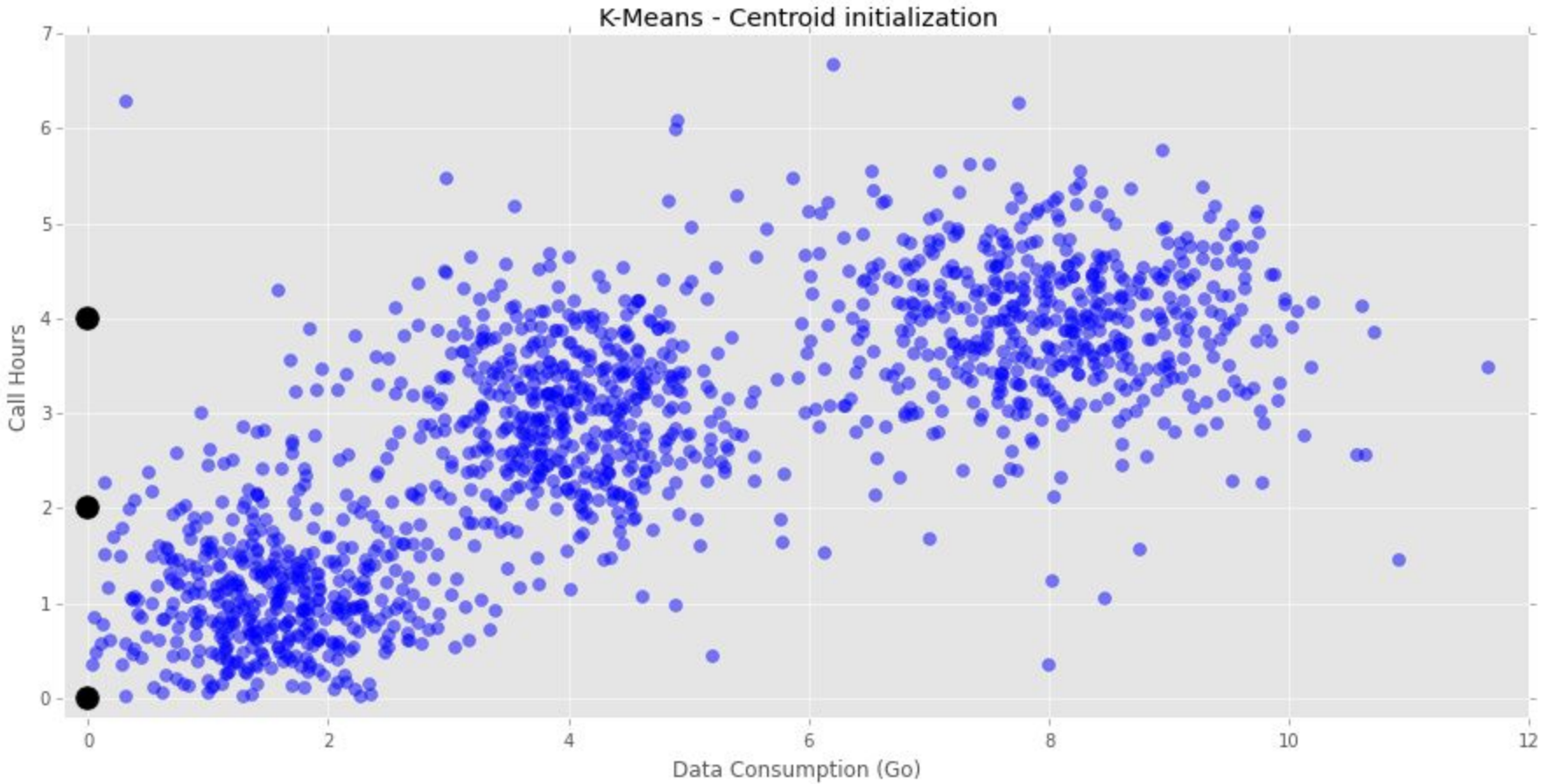
Sources

- Natural Language Processing
  - http://fr.slideshare.net/pranavgupta21/introduction-to-natural-language-processing-13847262
  - http://fr.slideshare.net/ananth/natural-language-processing-l01-introduction
- LDA
  - http://blog.echen.me/2011/08/22/introduction-to-latent-dirichlet-allocation/
  - http://fr.slideshare.net/WayneLee9/lda-oct3-2013
- Word2Vec
  - http://www-personal.umich.edu/~ronxin/pdf/w2vexp.pdf
  - http://deeplearning4j.org/word2vec.html

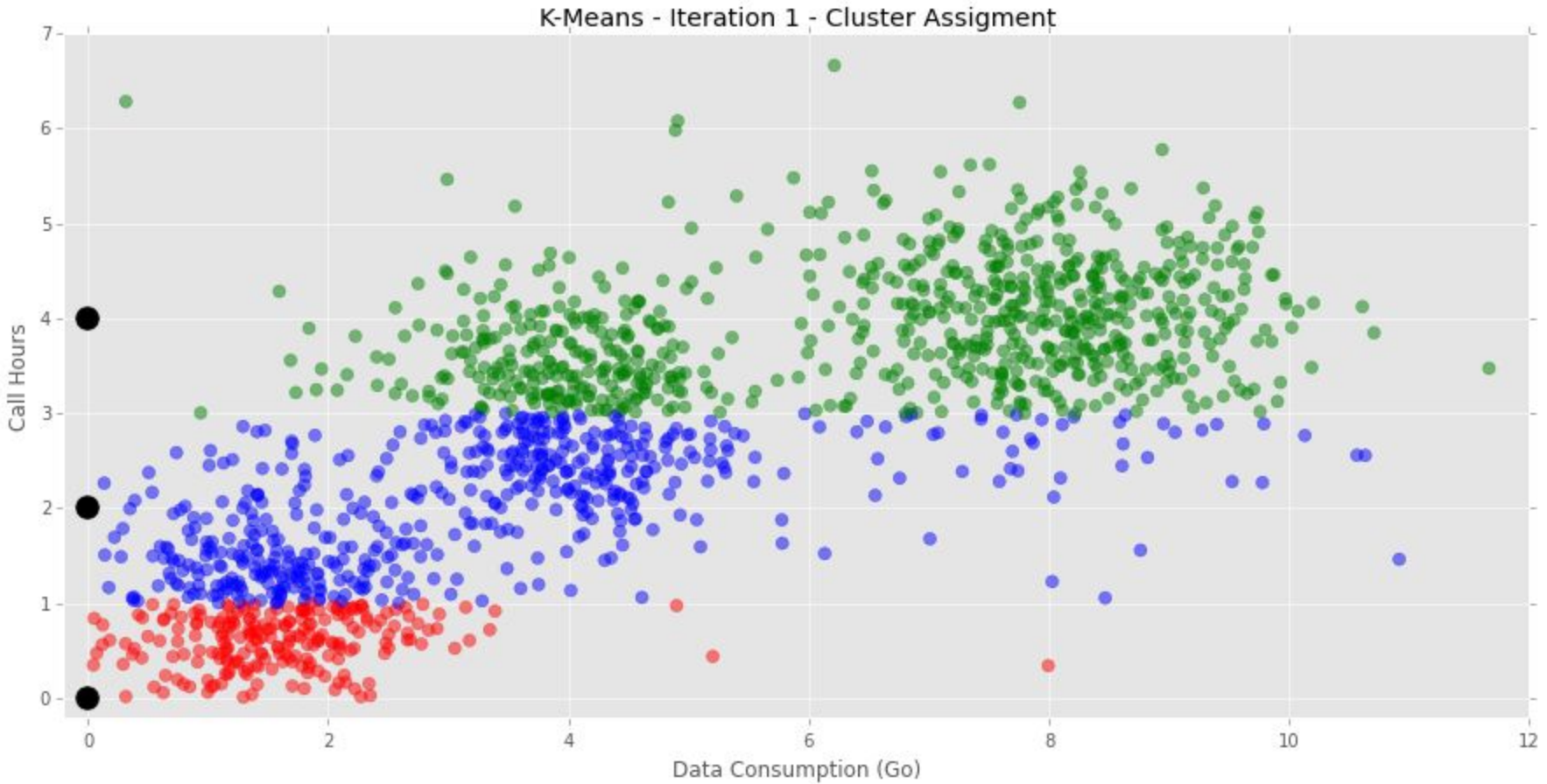
K-Means: Exemple



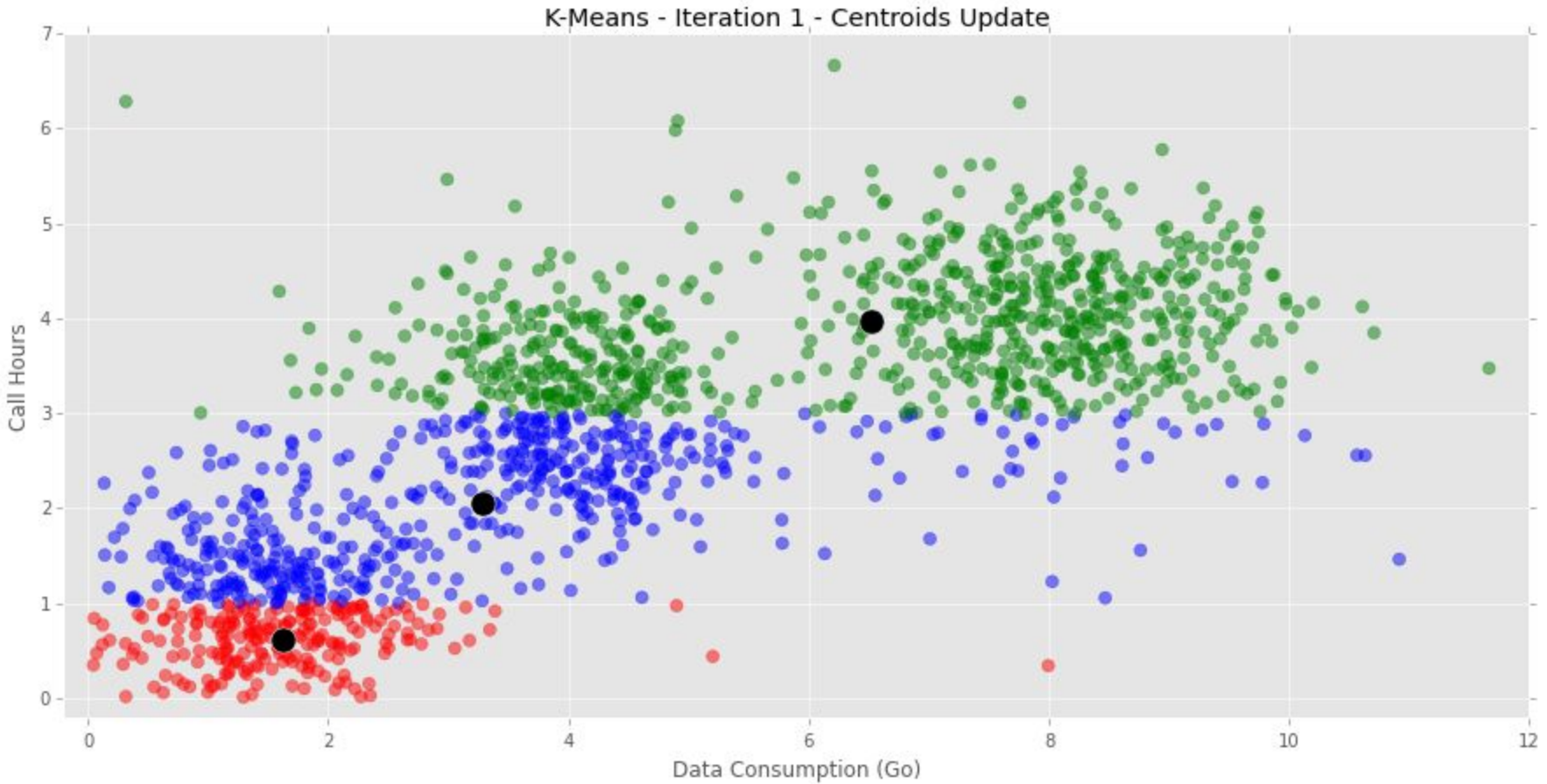
K-Means: Exemple



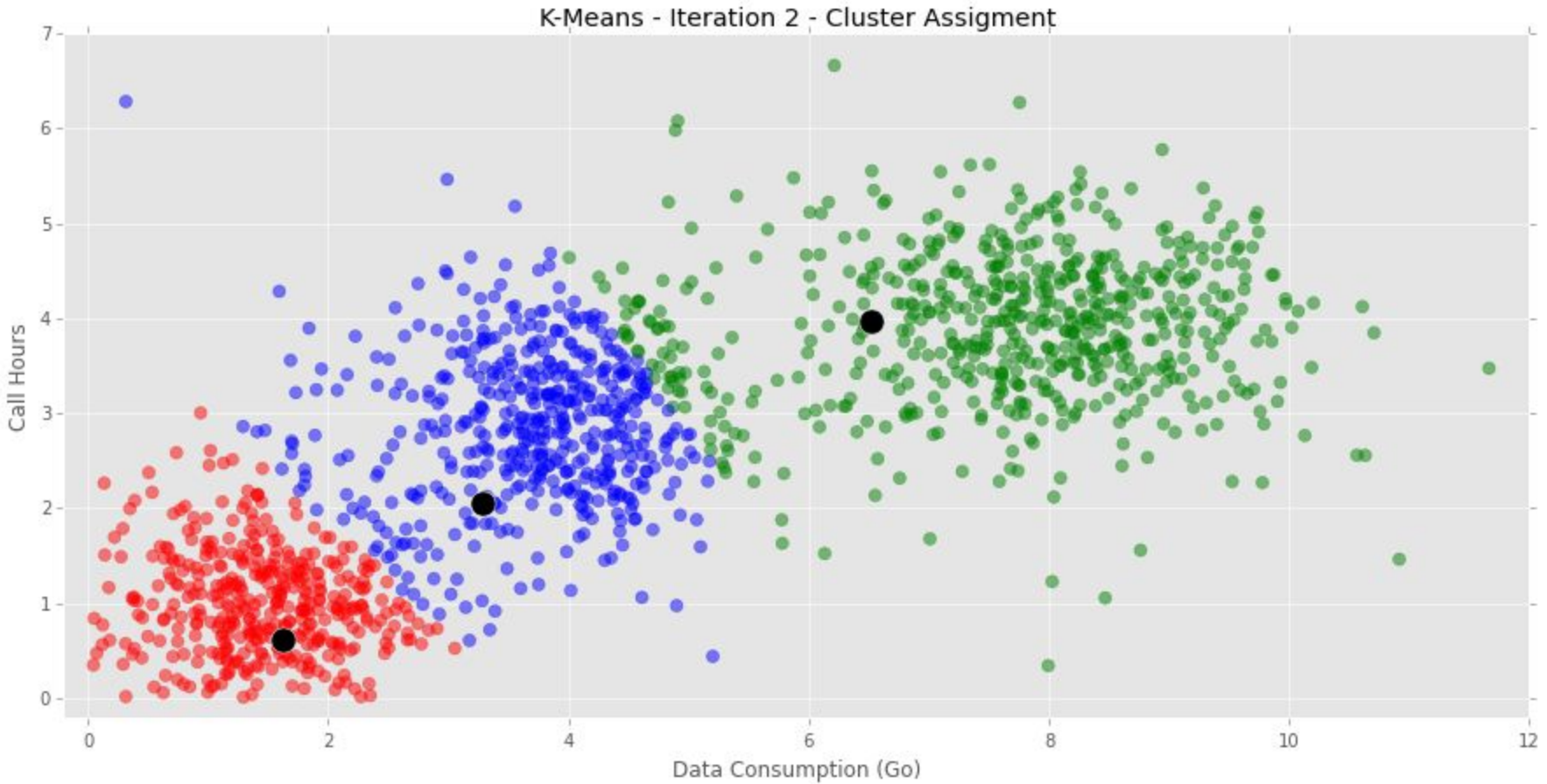
K-Means: Exemple



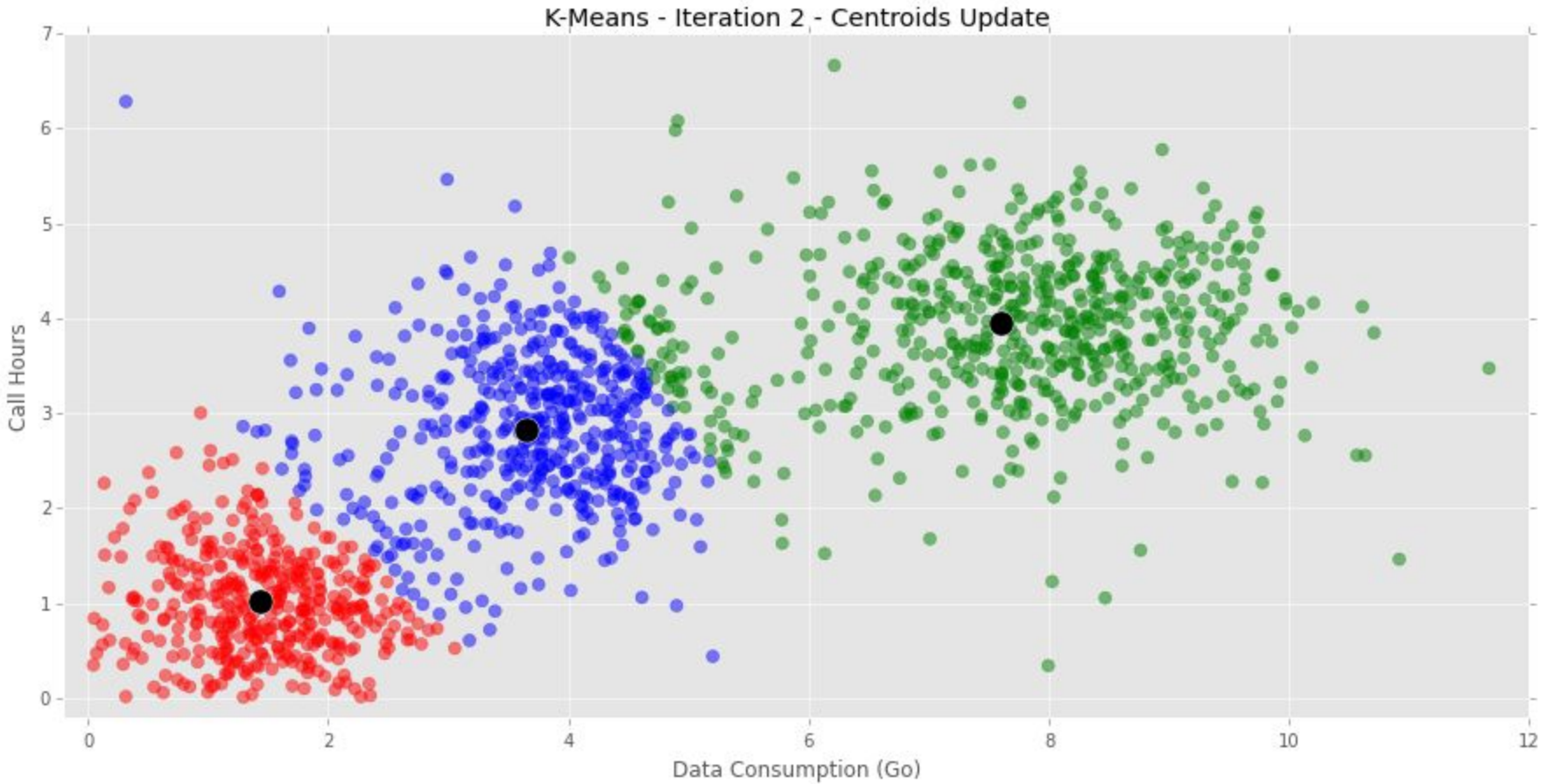
K-Means: Exemple



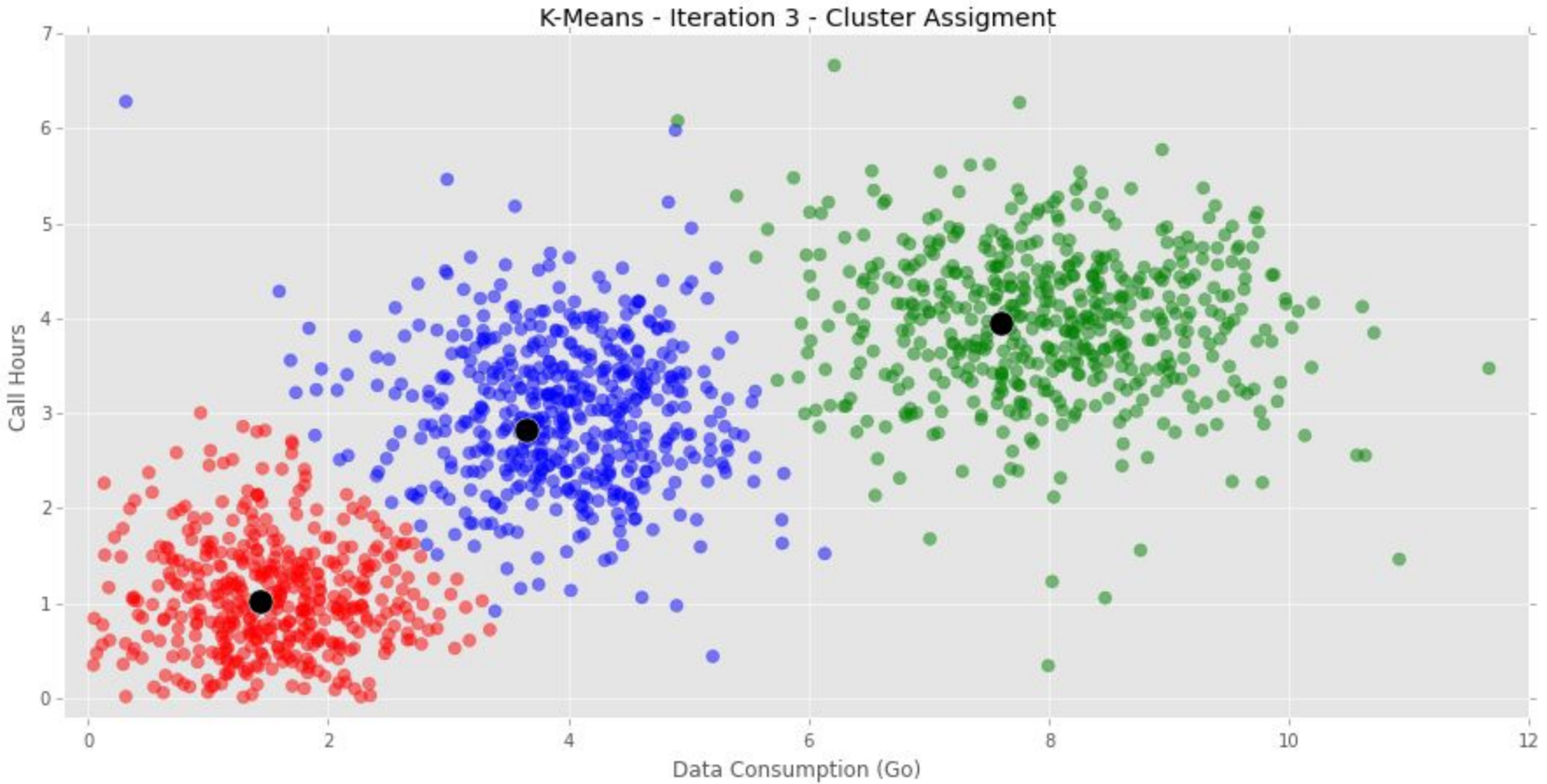
K-Means: Exemple



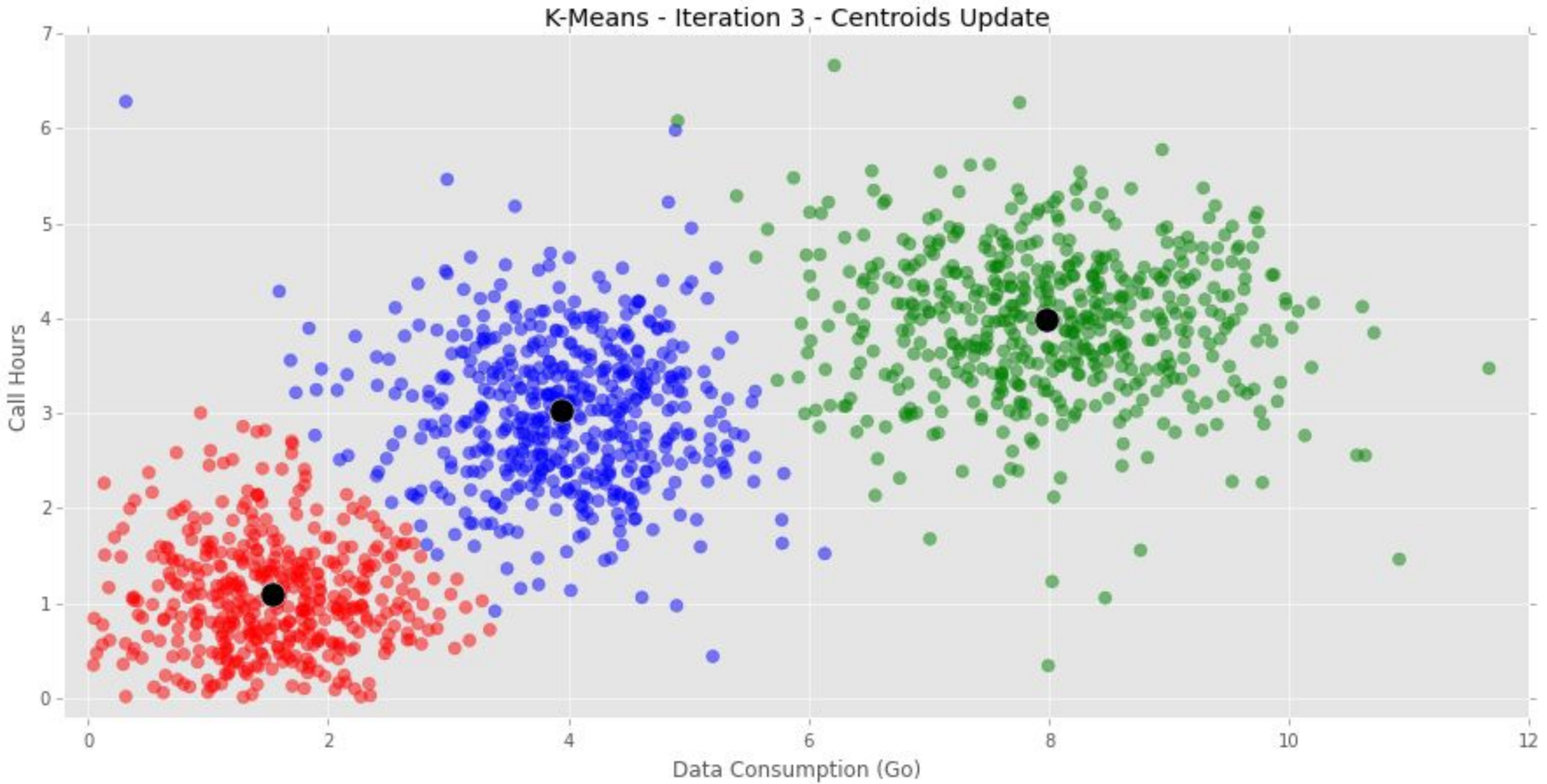
K-Means: Exemple



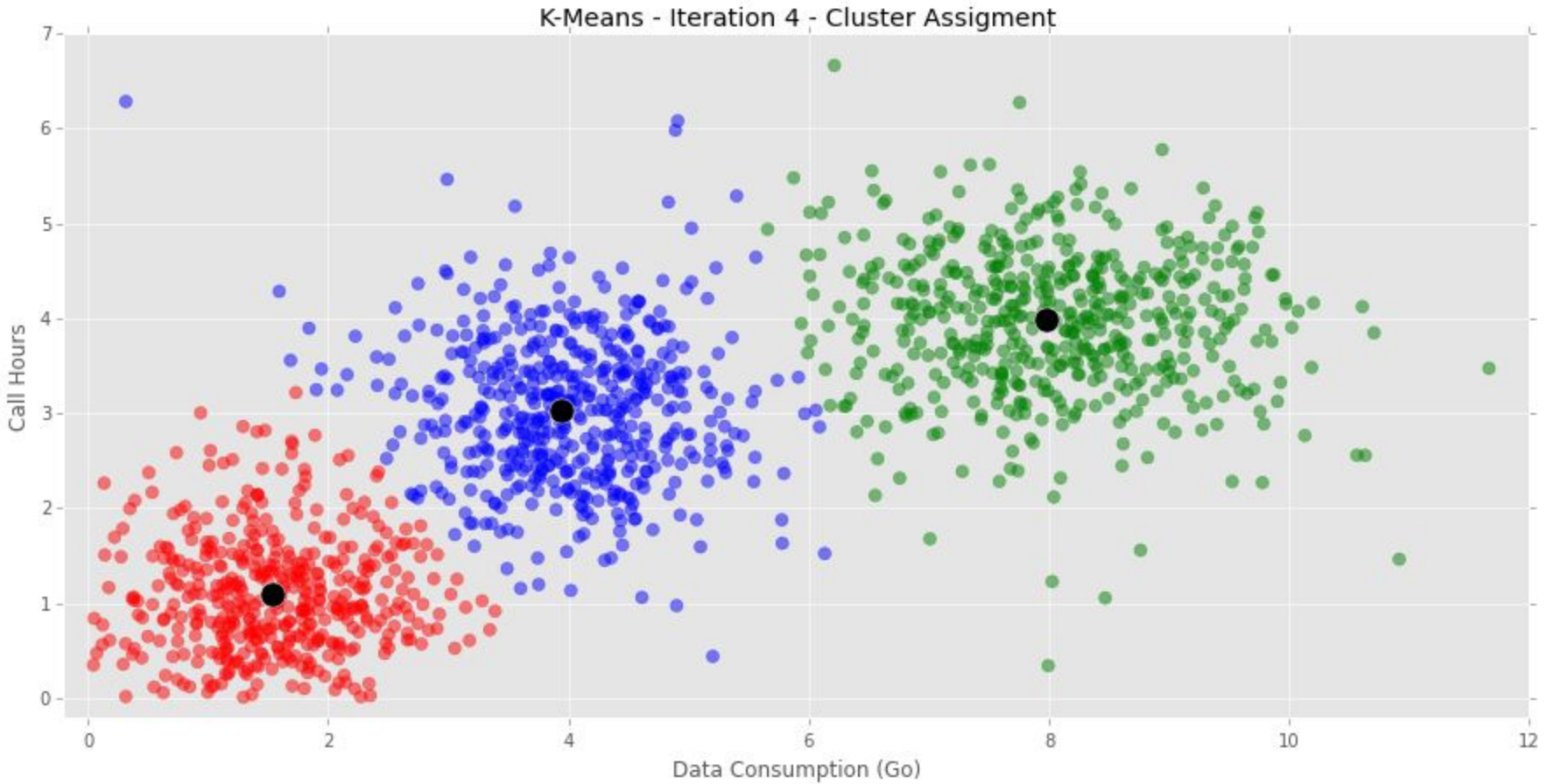
K-Means: Exemple



K-Means: Exemple



K-Means: Exemple



K-Means: Exemple