Compte rendu - Indexation multimédia **FAURE Adrien**
2015 - 2016

Contents

1 Description

Ce rapport présente mon projet d'indexation multimédia réalisé en master 2 Génie Informatique 2015 - 2016.

2 Présentation

La réalisation du projet suis fidèlement les étapes décrites dans le sujet. Le projet se présente sous la forme d'un dossier contenant tout les scripts et sous dossiers nécessaire. La racine du projet présentes trois scripts essentiel ainsi que quelques dossiers clés :

- `install.py` : Scripts d'installation. Créera les sous dossier nécessaire ainsi qu'un fichier de configuration relatif à l'environnement. Ce script compile également les sources annexes (lib-svm et le code des histogrammes de couleurs).

- `manage.py` : Point d'entré du programme, il permet d'exécuter un plan d'exécution défini pour une tache données (réaliser la chaîne complète des sift par exemple).
- `webapp.py` : Application web qui met à disposition deux routes http permettant l'évaluation d'un image.
- `rapport` : Ce dossier contient les sources latex du projet ainsi que les éléments issu des calcules (les meilleurs résultats obtenue et leurs paramétrages etc).
- `scripts` : Les scripts du projet. Les scripts représente les différentes étapes décrites dans le sujets. Génération des histogramme, trec_eval, calcule des k-means etc).
- `plans` : Ce dossier contient les différents plans d'exécution utilisé pour le projet.
- `predict_files` : Ce dossier contient les modèles nécessaire pour l'évaluation d'une image. Ce dossier n'est pas présent dans l'archive rendu car il représente trop de données. Cependant il est disponible sur la version git en ligne.
- `lib` et `dep` : Ces dossiers contiennent les logiciels/utilitaires annexes (djpeg,colorDescriptor ,libsvm et trec_eval).
- `webapp` : Ce dossier contient les sources de l'IHM web en javascript et html.

3 Présentation technique

3.1 Langage utilisé

Pour réaliser le projet, j'ai choisis le langage de programmation Python. La première raison est que c'est un langage que je ne connaissais pas et que je souhaitais découvrir, par chance ce sujet de tp était réalisable en python. De plus comme pour le langage R, python est un lange réputé dans le data-mining et machine learning. La dernière raison qui motive l'utilisation de python est que je voulais essayer l'appelle de fonction native C dans un code python. Le script `histogram.py` met en place ce mécanisme avec le calcul des histogrammes de couleurs.

3.2 Déroulement des opérations

Comme je l'ai succinctement décris dans la première partie du rapport, le script `manage.py` situé à la racine du projet est le point d'entrée des calcules. Il permet, via des fichiers de paramétrage, de lancer les étapes nécessaires à l'analyse des concepts. Le scripts `manage.py` présentes les options suivantes :

- `-f` : Lancement du plan d'exécution spécifié en paramètre.
- `-d` : Lancement de tous les plans d'exécution situé dans le dossier spécifié en paramètre.
- `-i` : Permet de spécifier un plan d'initialisation à lancer avant de lancer les plans spécifier avec les options `-f` et `-d`

`Manage.py` présente donc un moyens de lancer les évaluations, depuis le téléchargement des images jusqu'à l'évaluation final avec trec_eval. Il est utile de préciser que chaque exécution peu prendre du temps (plusieurs heurs , voir un jour complet avec beaucoup de plans). Afin de pouvoir suivre les étapes tout les scripts laisse une trace dans un fichier de log situé dans le dossier `log` créé lors de l'installation du projet. Ces fichiers sont importants car sans eux, il devient difficile voir impossible de débugger le programme.

Cette organisation représente une base solide pour l'analyse des images à travers les étapes imposé. En particulier, une fois qu'un plans d'exécution est bien définie pour une évaluation, par exemple pour l'évaluation des sift, il suffit de le répliquer avec autant de paramétrages différents que souhaité. L'option `-d` de `manage.py` exécutera chacun des plans du dossier qui les contient à la suite.

Chacun des scripts de calcul est construit de façon à recevoir une liste de concepts. Ce qui, on le verra plus tard permet de paralléliser les taches, car en général il n'y à pas de conflits entre les exécutions de chaque concept.

4 Parallélisation et optimisation

4.1 Localisation des images

Le projet à été réalisé dans une optique d'optimisation et de réutilisation des temps de calculs. Par exemple afin de calculer les histogrammes de couleurs de chacune des photographie le scripts `histogram.py` télécharge une fois pour toutes les photographies mis à disposition. Si l'algorithme d'histogrammes est appelé une seconde fois le script vérifiera la présence de chacune des photographies avant d'exécuter l'analyse des couleurs. La même démarche à été adopté avec les sift. Ceci permet de pouvoir optimiser les temps de transferts réseau pour chaque exécutions.

4.2 Factorisation des taches

4.2.1 Plan d'initialisation

L'option `-i` du script `manage.py` déclenchant l'exécution d'un plan d'initialisation avant de d'exécuter le ou les plans principaux permet ainsi de factoriser certaines taches propre à un ensemble de plans.

4.2.2 Couleur

Pour l'évaluation des couleurs, seulement deux histogramme de couleurs sont nécessaires pour toutes les évaluation (train et val). On peut ainsi lancer le script `manage.py` avec la commande :
`./manage.py -d dir_plans_color -i generate_color_histogramme.ini .`
`Manage.py` exécutera alors une seul fois le plan de génération des histogrammes de couleur pour tout les plans d'évaluations des images pour les histogrammes de couleurs.

4.2.3 Sift

Pour l'évaluation des sift on génère un fois pour tout les plans les histogrammes des clusters au format svm. Dans le cas des sift cette option permet de gagner encore plus de temps. En effet, mes évaluations font varier le nombres de centres des k-means. Le plan

d'initialisation doit ainsi générer les cluster k-means sur plusieurs dimensions (256, 512, 1024 ainsi que 2048) ainsi que les mappings et histogrammes svm. La génération d'un cluster de k-means pour 2048 centres dure environ une heure (avec un échantillonnage des sift à 1 sur 25). De plus la génération des mappings prend un certains temps. Une demi heurs pour les mappings des clusters à 256 centres et deux heures pour les mappings des cluster à 2048 centres. Il est quasiment impensable de devoir recalculer ces informations pour chaques variations de gamma et des poids svm.

4.3 Parallélisation des taches

4.3.1 Popen

Le langage de script Python présente un api inintéressante pour l'exécution de processus en background : Le module Popen de subprocess. Cette bibliothèque python permet de lancer et contrôler des scripts ou des commande native (bash par exemple) à partir du scripts principal. En plus de pouvoir lancer des processus en background, Popen permet de récupérer leurs sorties d'erreurs ainsi que le code de retour du scripts. Ce qui permet de gérer avec p rescision leurs exécutions et ainsi d'alimenter les fichiers de logs.

4.3.2 Taches parallélisées

Lors de l'exécution d'un plan, chacune des étapes est conçu pour être exécutée une fois pour chaque concept. En opposition à une implémentation mono-concept, le script de génération des modèles générera les modèle de chacun des concepts donné en paramètre avant de passer à l'évaluation avec svm-predict. Une implantation mono concept exécuterais tout la chaîne d'évaluation pour un concept avant de passer au suivant. Cette information est capitale pour comprendre les taches possiblement parallélisable relative à mon implantation. Le nombre de processus désiré est en général une options préciser au script `manage.py` (-nb-threads). Afin de ne pas figer la machine, le scripts exécutera `nb-threads` processus maximum, quand un processus est terminer un nouveau peut démarrer jusqu'à la fin du programme. Le projet permet de paralléliser les tâches suivantes :

- La génération des mappings `sift -> cluster`, le traitement d'une image est parallélisé.
- La génération des modèles (couleur et sift), le scripts lancera en concurrence le calcul du modèle pour un concept.
- La génération des fichiers de concepts pour les couleurs.

D'autre taches peuvent être parallélisé, cependant cela ne représente par un grand intérêt si l'exécution pour les 20 concepts est inférieur à 10 minutes. En effet un svm-train pour un concept sift prend une dizaine de minute, et il y en à une vingtaines calculer c'est donc un gain de temps significatif. Cependant la fusion des .out lors de la fusion tardive ne prends pas plus de quelques minutes (voir moins d'une minute) pour les vingts concepts. Une parallélisation n'est donc pas nécessaire en rapport gains de temps d'exécution contre gains de temps en programmation.

4.4 Gestion des risques

La gestion des risque dans un projet comme celui ci est très important. Le premier facteur à contrôler est le temps. En effet l'exécution d'un plan d'évaluation des sift pour un paramétrage donné prend environ une heure. Il est donc important de commencer en avance le projet afin de pouvoir optimiser ses résultats et ainsi permettre l'évaluation pour beaucoup de paramétrages.

La gestions des bug est également principale. En effet, durant mon projet j'ai été confronté à deux bugs importants. Comme un plan d'évaluation représente beaucoup d'enchaînements de scripts est de calcul il n'est pas toujours évident de déceler un bug. De plus il est important d'avoir une connaissance approfondit de chacun des outils que nous utilisons. Dans mon cas lors de l'évaluation par trec_eval je téléchargeais les fichiers `.rel` avec l'utilitaire `wget`. Cependant à la première utilisation j'ai téléchargé un fichier erroné, wget par défaut ne remplace pas les fichiers existant mais crée un fichier nouveau `nomfichier.rel.1`. Ce comportement ma coûté une semaine avec de très mauvais résultats.

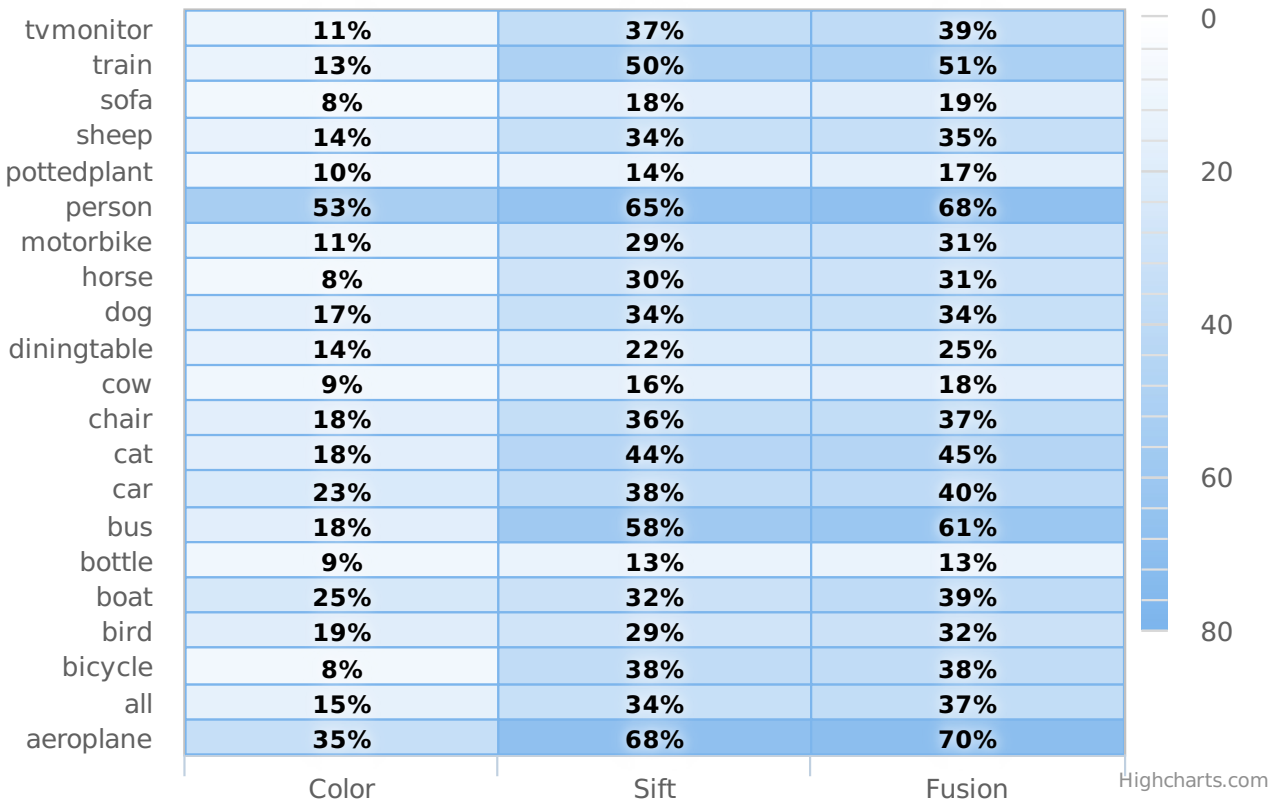
Une deuxième source de bug entraînant des résultats proches de zéro est la quantités importante de fichiers. En effet, lors du téléchargement des images, le réseau s'est arrêté entraînant ainsi le téléchargement incomplet d'un fichier sift. Le fichier tronqué entraînais ainsi une erreur dans les mappings de clusters sift menant finalement un décalage de 1 pour chaque images dans les histogrammes.

5 Résultats

5.1 Résultats général

Il est important de noter que le résultat présenté pour `all` représente la valeur du gain apporté pour le paramétrage permettant de trouver le meilleur compromis pour tout les résultats. Il donc pas très intéressant dans notre cas. Une valeur plus pertinente de `all` est la moyenne des évaluations pour chaque concept avec leurs meilleurs paramétrages :

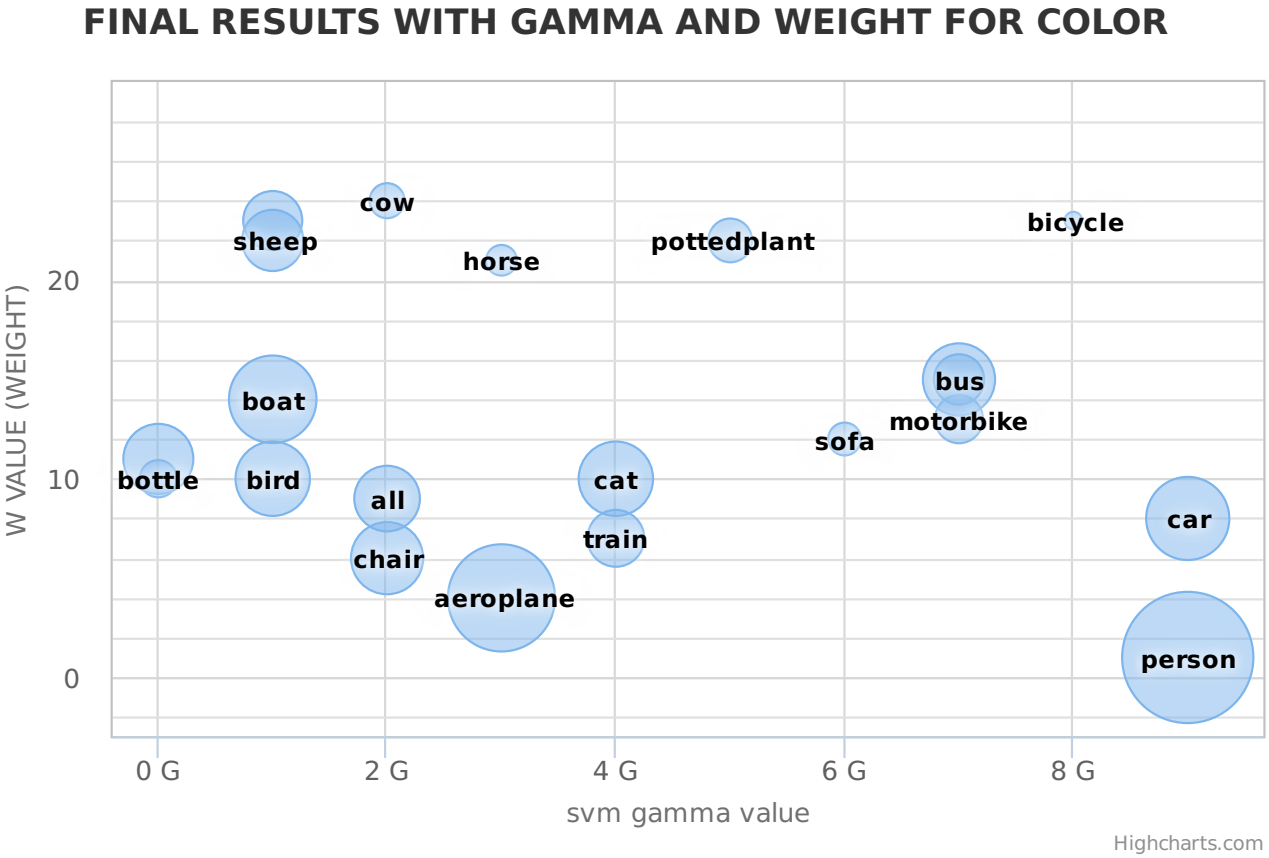
- sift : 35%;
- color : 17%;
- fusion : 37%.



Voici les résultats obtenue pour chaque concept par catégorie. On observe une net amélioration au passage de couleur à sift. Avec un amélioration de 77.5% pour `all`. Le passage de sift à fusion permet tout de même d'améliorer légèrement les résultats avec une augmentation de 8,4%. Les évaluations par histogrammes de couleur permettent donc d'apporter une précision en plus sur le résultats général grâce à la fusion tardive.

Afin d'optimiser les résultats obtenue, les évaluations sont exécuté plusieurs fois avec des paramétrages différents.

5.2 Résultats de l'évaluation par histogrammes de couleurs



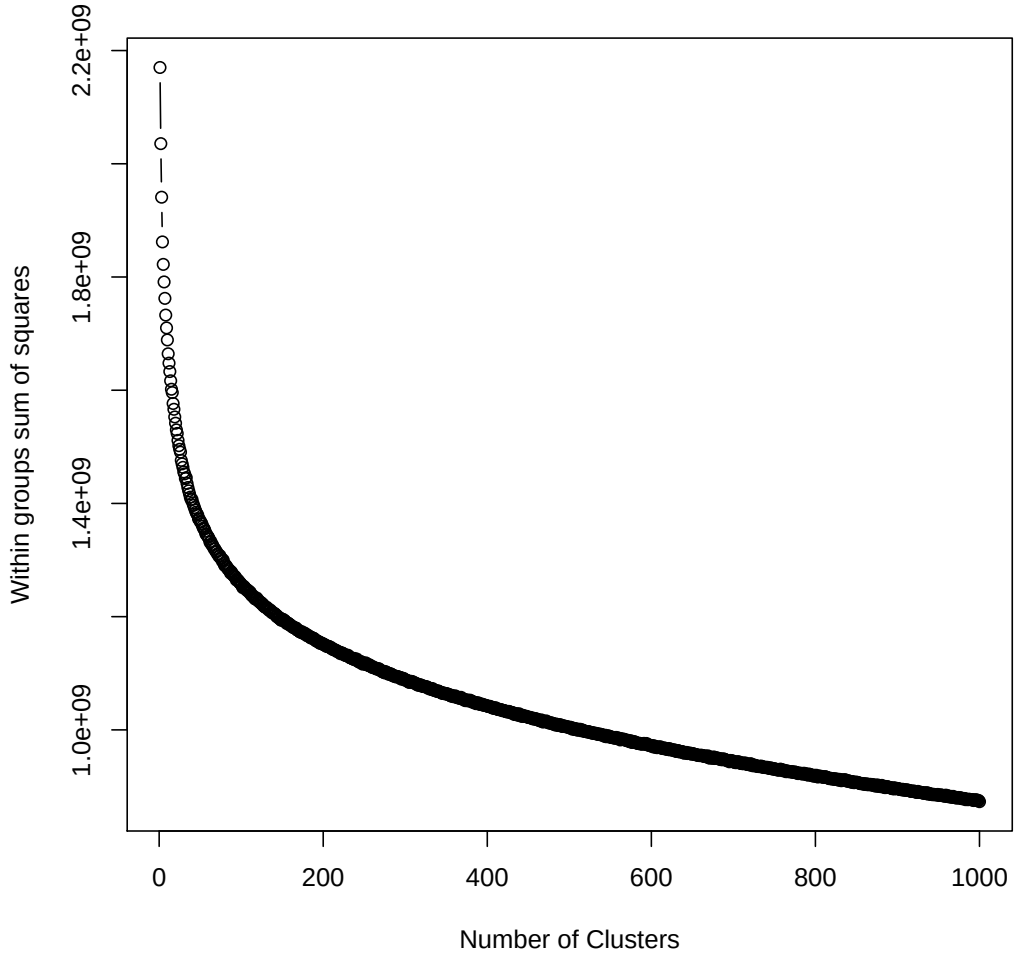
Pour l'évaluation des couleurs, les paramètres à varier sont la valeurs gamma de svm- train qui représente une marge d'erreur acceptable pour la séparation ainsi que le paramètre du poids attribué à un élément enregistré comme vrai dans la base d'image. Les valeurs de gamma sont choisis dans les intervalles suivants : $gamma \in [\![0\ ;\ 9]\!]$ et $w \in [\![0\ ;\ 25]\!]$. Le diamètre du cercle représente la valeur obtenue avec l'évaluation de trec_eval.

5.3 Résultats de l'évaluation par sift

Pour l'évaluation des sift, j'ai choisis de faire varier trois paramètres.

- Le paramètre gamma de svm;
- Le poids pour un concept (paramétrage svm);
- Le nombre de centre du k-means.

Afin d'avoir une bonne estimation du nombre de cluster à choisir lors de la clusterisation, j'ai tracé la courbe de disparition du kmeans (valeur fournit dans l'objet kmeans du package de R).



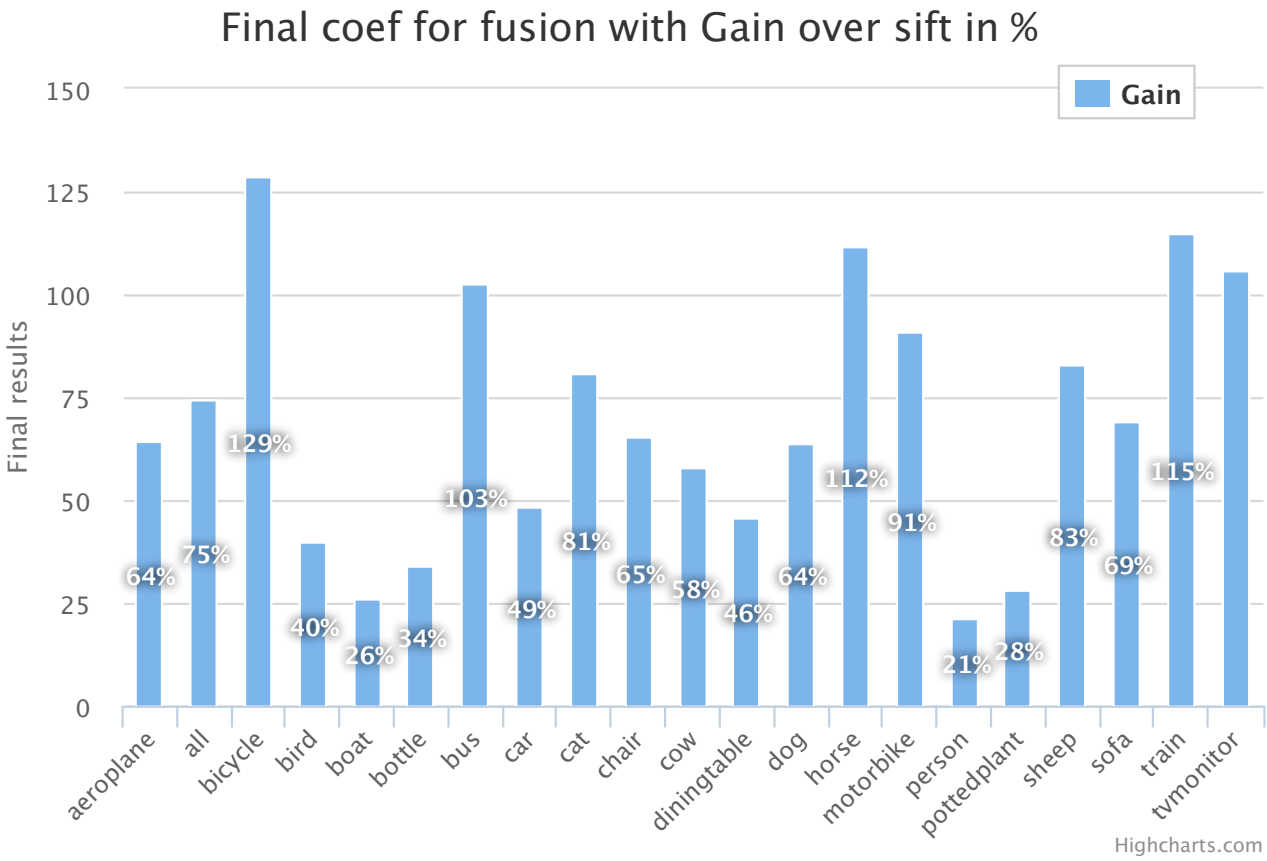
En général un "bon" nombre de cluster se situe dans la région du `coude` de cette courbe. Ainsi les clusters de 200 centres à 2000 centres représente une clusterisation de qualité. C'est pourquoi j'ai choisis de faire les évaluations pour : $nbClusters \in [\![256, 512, 1024, 2048]\!]$.

D'une façon générale, l'évaluation d'image grâce aux calculs de sift apporte de meilleurs résultats finaux. Les paramètres ont été selectionnés dans les ensembles :

- $nbClusters \in [\![256; 512; 1024; 2048]\!]$.
- $gamma \in [\![1; 25; 50; 150; 400]\!]$.
- $w \in [\![5; 20; 50]\!]$.

concept	Nombre de centres	Gamma	w	Resultats (map)
aeroplane	1024	150	20	0.6883
all	512	150	5	0.3465
bicycle	512	150	5	0.3813
bird	512	150	5	0.2910
boat	1024	150	20	0.3273
bottle	512	25	20	0.1344
bus	2048	150	20	0.5828
car	512	150	5	0.3818
cat	512	150	5	0.4418
chair	512	400	20	0.3604
cow	512	150	5	0.1692
diningtable	512	150	100	0.2252
dog	512	150	5	0.3438
horse	512	150	5	0.3091
motorbike	512	400	20	0.2983
person	512	400	20	0.6571
pottedplant	256	100	20	0.1415
sheep	2048	150	20	0.3482
sofa	256	150	20	0.1834
train	512	150	5	0.5052
tvmonitor	2048	150	20	0.3774

Afin de présenter le gain de qualité j'ai dessiné le diagramme ci-dessous. Il présente pour chaque concept le gain effectif en pourcentage apporté à la prédiction des sifts. Voici la liste des paramètres optimaux trouvé pour chaque concept :



Le Gain est calculé de la façon suivante :

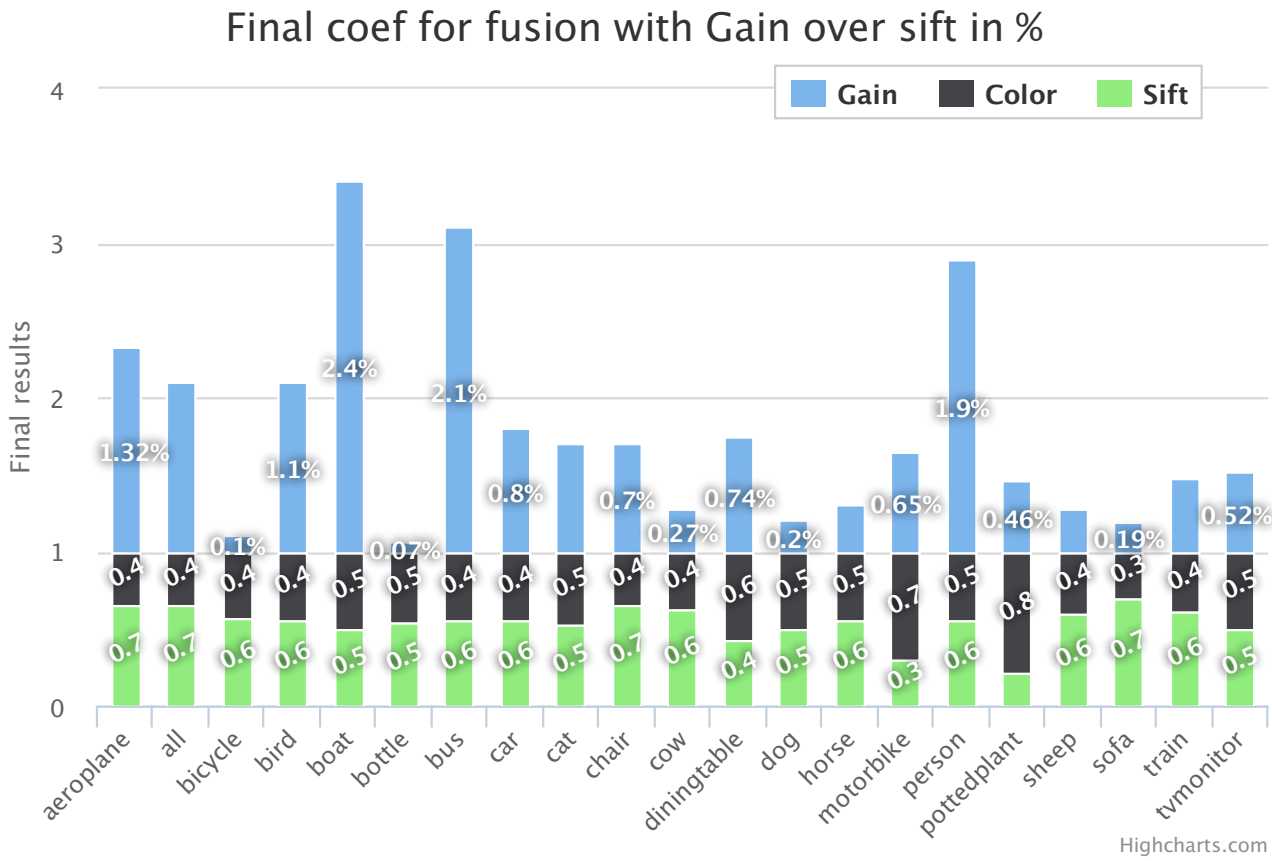
$$Gain = (|map_{sift} - map_{color}|/(map_{sift} - map_{color})/2) * 100 \quad (1)$$

La variables map est la variable correspondant au champ `map` dans le fichier résultat de trec_eval.

5.4 Résultats de l'évaluation par fusion tardive

Pour une optimisation maximal des résultats de la fusion, la fusion des fichier `.out` s'effectue depuis les fichiers issue des paramétrages ayant apporté les meilleurs résultats pour les sift et pour les couleurs. Par exemple, pour le calcul des avions, on prendra le fichier de sortie des paramétrages ;

- couleur : g = 4 et w = 3;
- sift : cluster de 1024 centres, g = 150 et w = 20.



Le schéma ci-dessus décrit le gain effectif en pourcentage apporté par la fusion par rapport au résultats de sift. La valeur "Gain" est calculé avec la formule :

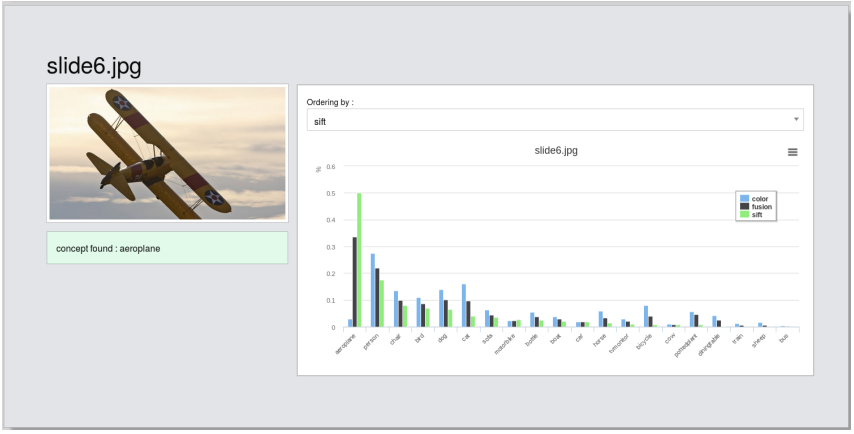
$$Gain = (|map_{sift} - map_{fusion}|/(map_{sift} - map_{fusion})/2) * 100 \quad (2)$$

Les valeurs de sift et de color représentent les poids attribués lors de la fusion des fichiers `.out`.

6 Evaluation d'une image

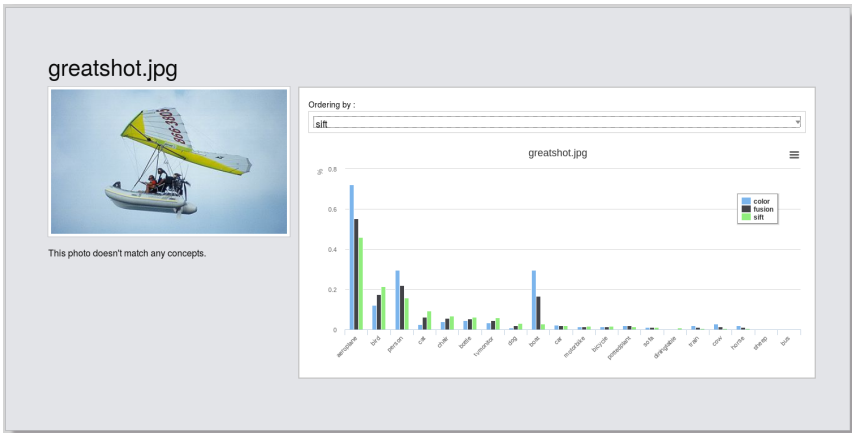
voici Les figures 1 à trois présentes quelques résultats obtenue grace à l'interface web sur l'évaluation d'une image.

(a) dining table and chair



(b) Aeroplane

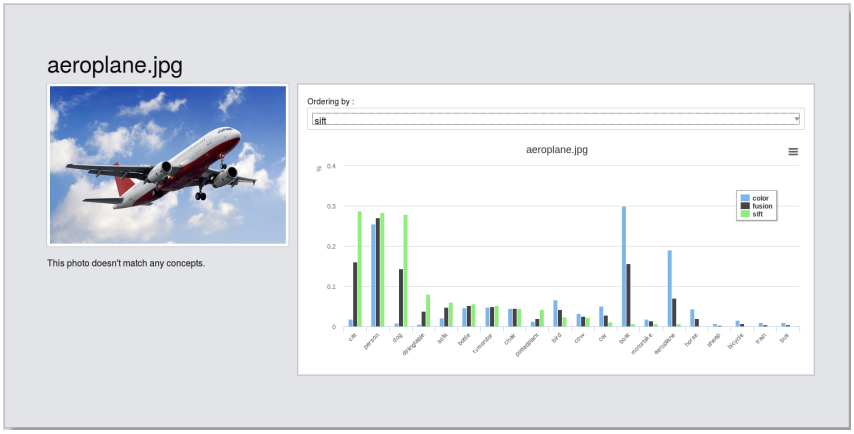
Figure 1



(a) Aeroplane

(b) Train

Figure 2



(a) Aeroplane - une Anomalie

(b) Un centaure

Figure 3