Travaux Pratiques
Recherche d'Information

Alexis Lechervy, Frédéric Jurie d'après un sujet de Riwal Lefort

1 Introduction

L'objectif de ce TP est de mettre en place un système de recherche d'image par le contenu basé sur une mesure de similarité. Nous souhaitons pouvoir chercher les images de la base les plus similaires à une requête donnée. Le résultat attendu est un classement de la base par similarité à la requête. Le système comporte deux entrées : l'image requête et la base de données. La sortie du système est la base de données rangée par ordre croissant de dissimilarité. On retrouve en premier les images les plus similaire et en fin de classement les images les moins semblable à la requête. La figure 1 donne un aperçu du système voulu.

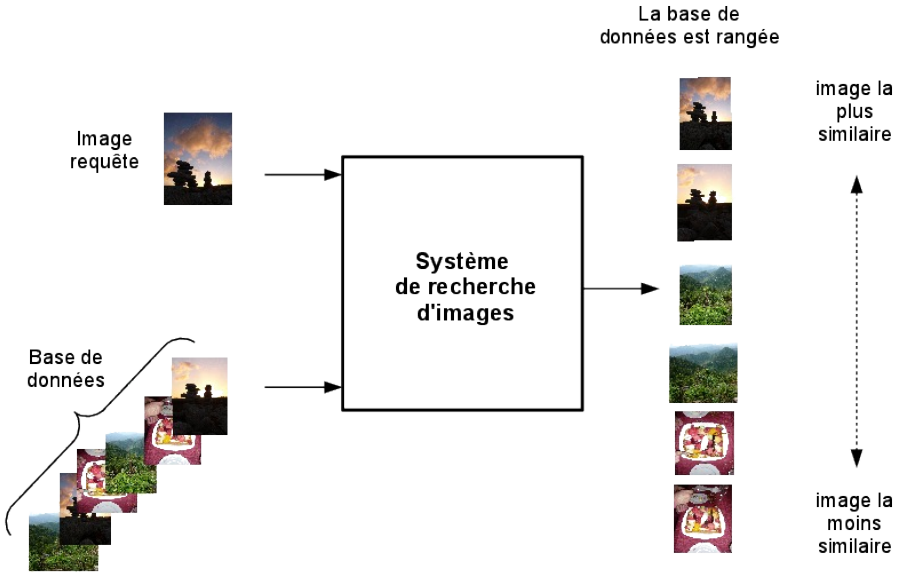

FIGURE **1. Le systŁme de recherche d'images range les images d'une base de données, en fonction de leur similarité à l'image requête.**

Ce TP est à réaliser en python 3. Vous pouvez utiliser les bibliothèques standard de python comme

numpy, scipy, matplotlib... ainsi que la bibliothèque d'apprentissage automatique scikit-learn (`http://scikit-learn.org`).

Nous allons détailler dans les parties suivantes les différents éléments nécessaires à la compréhension et la réalisation du sujet. La partie 2 vous indiquera le travail à effectuer, tandis que la partie 3 vous précisera les éléments que vous aurez à rendre.

1.1 La base de donnée

Nous travaillerons sur la base de données *INRIA Holidays* (`https://lear.inrialpes.fr/~jegou/data.php`). Pour plus d'information sur cette base, vous pouvez vous référer à l'article [3].

Cette base contient 1491 images au total et 500 classes composées de 3 ou 4 images. Les 500 images requêtes (une pour chaque classe) correspondent à la première image de chaque classe.

Nous vous avons mis sous moodle différents descripteurs (voir les parties suivantes) pour travailler sur cette base, ainsi que les points SIFT qui ont permis de les créer.

1.2 La mesure de disimilarité

Pour comparer les images de la base à une image requête, nous allons utiliser la distance euclidienne (vous pouvez vous contenter du carré des distances) entre les descripteurs des images et celui de l'image requête. L'image la plus similaire à la requête sera alors l'image dont la distance est minimale. Le classement de la base par rapport à la requête se fera donc par ordre croissant des distances à la requête.

De manière plus formelle, soit l'image requête I qui est représentée par un vecteur $x \in \mathbb{R}^d$. Soit $\{I_1, \ldots, I_n\}$ les images de la base de données qui sont représentées respectivement par les vecteurs $\{x_1, \ldots, x_n\}$, où $x_n \in \mathbb{R}^d$. Soit D_i la distance euclidienne entre le vecteur x et le vecteur x_i. Nous voulons ranger la base de données par rapport à l'image requête. Cela équivaut à ranger l'ensemble des distances par ordre croissant. L'algorithme de classement des données contient donc deux étapes : le calcul des distances $\{D_i^2\}_{i=1}^n$ et le classement de ces valeurs par ordre croissant.

Lorsque vous coderez cette distance, vous essaierez de proposer une solution avec le moins de boucle for possible en privilégiant le calcul matriciel et vectoriel.

Vous n'hésiterez pas à utiliser numpy et scipy, notamment pour la fonction de tri.

1.3 La représentation des images

Nous allons dans ce TP comparer deux types de représentations des images issus de la littérature :

— La représentation par sac de mots ou *Bag-of-Word* (BoW [7]).

— La représentation par *Vector of Locally Aggregated Descriptors* (VLAD [4]).

1.3.1 BOW

Les sac de mots ou Bag-of-Word (BoW) est une méthode de représentation des images proposée par [7]. Elle s'inspire d'une méthode du même nom utilisée pour la représentation de texte. Un lecteur intéressé par les méthodes utilisées pour le texte pourra se référer par exemple, au livre [1] pour compléter les explications qui vont suivre.

Dans le cas d'un texte, un document est une suite de mots. L'ensemble des mots employés dans le corpus de textes analysé peut être regroupé au sein d'un dictionnaire dépendant de la langue utilisée.

La représentation par sac de mot d'un document consiste à représenter le document par un vecteur des fréquences d'apparitions des mots du dictionnaire dans le texte. Un document est donc représenté par un vecteur de la même taille que le dictionnaire, dont la composante i indique le nombre d'occurrences du $i^{\text{ème}}$ mot du dictionnaire dans le document. Pour une implémentation pratique dans le cas du texte, vous pouvez vous référer au tutoriel de sckit-learn sur ce sujet (`http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html`).

Dans le cas d'une image, nous auront une approche similaire en utilisant le concept de "mot visuel". Pour une image I donnée, il est possible d'extraire un ensemble $\{s_1, \ldots, s_M\}$ de points SIFT (Scale Invariant Feature Transform [5]) . Ces points SIFT seront vu de manière similaire aux mots qui compose un texte. Comme pour un mot d'un texte que l'on associe à un mot d'un dictionnaire, on associera chaque SIFT à un "mot visuel" d'un "dictionnaire de SIFT".

En pratique, un "dictionnaire de SIFT" est un clustering modélisant la distribution de tous les points SIFT du corpus d'image étudié. On peut utiliser pour ce clustering un algorithme de K-means à K clusters. Associer un SIFT à un "mot visuel" consiste donc à l'associer au centroid qui lui est le plus proche. Pour plus de détails sur ces méthodes, référez-vous à votre cours d'Apprentissage Automatique du premier semestre (partie Apprentissage non supervisé).

On notera que des SIFT différents peuvent être associés au même cluster/mot. On retrouve la même idée sur le texte où par exemple, un nom et son pluriel sont associés au même mot du dictionnaire tout en ayant une graphie différente.

On représente ensuite les images par la fréquence de présence de chaque cluster dans l'image. Autrement dit, cette représentation de l'image encode la distribution des points SIFT dans les K clusters pour l'image.

Le fichier bow.txt qui vous est fourni, contient pour chaque image le nombre de SIFT associé à chacun des K clusters. Il suffit de normaliser L1 les vecteurs pour avoir les fréquences d'apparition.

1.3.2 VLAD

Les *Vector of Locally Aggregated Descriptors* (VLAD) sont des descripteurs introduit par [4]. Ils peuvent être vu comme une simplification des *noyaux de Fisher* [2] et peuvent être rapproché des *Fisher Vectors* [6].

Comme nous l'avons vu dans la partie précédente, les BoW encode la fréquence d'apparition dans l'image de mots d'un dictionnaire. Chaque descripteur local (dans notre cas des SIFT) est associé au mot visuel le plus proche. L'idée des VLAT n'est pas de s'intéresser à la fréquence des mots du dictionnaire mais à la somme des déviations des mots de l'image aux valeurs des mots du dictionnaire qui leur sont associés. Contrairement à la représentation des BOW, qui ne permet pas de caractériser la distribution des points SIFT au sein d'un cluster, le codage de la méthode VLAD intègre cette information.

Soit K le nombre de mots visuels du dictionnaire. Soit F la dimension d'un mot. L'image I est représentée par le vecteur $v \in \mathbb{R}^{K \times F}$, dont la composante indicée par $\{i, j\}$ s'exprime ainsi :

$$v^{(i,j)} = \sum_{x \text{ les SIFT associé au cluster } c_i} x^{(j)} - c_i^{(j)} \tag{1}$$

$x^{(j)}$ étant la $j^{\text{ème}}$ composante d'un SIFT associé au cluster c_i.

On recommande généralement de normaliser L2 les VLAD.

1.4 Mesure des performances

Afin d'évaluer les performances de notre système de recherche, nous avons besoin d'une mesure d'évaluation. Dans ce TP, nous utilisons le *Mean Average Precison* (MAP).

Le MAP correspond à la moyenne des scores d'*Average Precison* (AP) sur l'ensemble des requêtes. C'est une valeur comprise entre 0 et 1. Plus sa valeur est forte, plus le système est performant.

Si Q est le nombre de requête (500 sur notre base de donnée), on a :

$$MAP = \frac{1}{Q} \sum_{q=1}^{Q} AP(q). \tag{2}$$

L'*Average Precison* (AP) est l'aire sous la courbe de précision /rappel pour une requête. Si on note $p(r)$ la fonction de précision en fonction du rappel, on a :

$$AP = \int_0^1 p(r) \mathrm{d}r. \tag{3}$$

En pratique, on utilise une approximation de cette aire. On peut par exemple utiliser un échantillonnage par rectangle supérieur comme dans la figure 1.4.

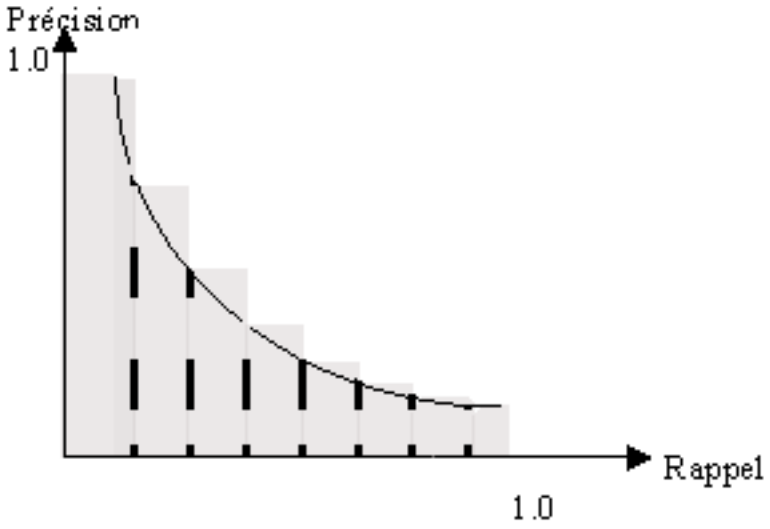

FIGURE **2. Approximation de l'intégral de la courbe de rappel/précision**

Si on note n le nombre d'exemple de la base d'apprentissage, $P(i)$ la précision au rang i, $\Delta r(i)$ la variation de rappel entre le rang $i-1$ et le rang i, on a comme approximation :

$$AP \simeq \sum_{i=1}^{n} P(i) \Delta r(i). \tag{4}$$

La précision au rang i est le nombre d'image correctement classée au rang i sur le nombre total d'image attribuée par le système à la classe étudié (en l'occurrence i ici).

Le rappel au rang i est le nombre d'image correctement attribuée à la classe étudiée au rang i sur le nombre d'image appartenant réellement à la classe étudiée.

Par conséquence on a :

$$AP \simeq \frac{1}{N_c} \sum_{j=1}^{N_c} \frac{j}{n_j}, \tag{5}$$

où N_c est le nombre d'exemple appartenant réellement à la classe étudiée et n_j est le nombre d'image nécessaire pour avoir j images de la classe étudiée parmi les premières images du classement. Autrement dit s'est le rang a partir du quel on a rencontré j images similaires à la requête.

2 Travaux à réaliser

Le travail que vous avez à réaliser est à effectuer en python 3. Une explication détaillée des concepts utilisés et une interprétation précise de vos résultats sont attendus.

2.1 Mise en place d'un systŁme de recherche par similarité

1. Commencez par charger les fichiers *bow.txt* et *vlad.txt* que vous pouvez récupérer sous moodle. En regardant les données contenues dans ces fichiers, indiquez le nombre de cluster K utilisés pour le codage BOW et le codage VLAD. La dimension des SIFT utilisés pour la création de ces fichiers est 128.
2. Réalisez le code permettant de réaliser le système de recherche de la figure 1. Votre code doit prendre en entrée les descripteurs associés aux images de la base de donné et une requête et doit retourner le classement des images de la base selon cette requête.
3. En utilisant la fonction de la question précédente calculez le MAP sur les 500 requêtes pour les deux types de représentations. Vous pouvez utiliser la fonction label_ranking_average_precision_score de sckit-learn. Quelle méthode est selon vous la plus performante ?
4. Calculez expérimentalement le temps de calcul. Quelles méthodes est la plus rapide ? Expliquez d'où vient cette différence de temps de calcul.
5. Les vecteurs qui vous sont fournis ne sont pas normalisé. Caractérisez l'effet de la normalisation (L1 et L2) sur les résultats.
6. Faites un bilan comparé des avantages et des inconvénients des BoW et des VLAD.
7. En lisant les articles cités en références ainsi que les articles que vous pourrez trouver dans la littérature (que vous prendrez le soin de citer), que pouvez-vous proposer comme idées pour améliorer votre système de recherche ?

2.2 Mesure des performances du systŁme de recherche

Vous avez dans la partie précédente utilisée des fonctions de sckit-learn pour calculer le MAP . Dans cette partie nous allons nous intéresser à recoder ces fonctions.

1. En utilisant l'équation 5 calculez à la main l'AP pour des systèmes qui auraient donnés les réponses suivantes :

(a)

Rang	1	2	3	4	5	6
Similaire à la requête	oui	non	oui	non	non	oui

(b)

Rang	1	2	3	4	5	6
Similaire à la requête	oui	oui	oui	non	non	non

(c)

Rang	1	2	3	4	5	6	7	8
Similaire à la requête	non	oui	non	oui	oui	oui	non	non

2. Coder une fonction calculant l'AP en utilisant l'équation 5. Votre fonction doit prendre deux vecteurs y_true, d_pred. y_true est un vecteur dont les valeurs sont des 1 si l'image appartient à la même classe que la requête et 0 sinon. d_pred est le vecteur des distances à la requête. Comparez vos résultats à la fonction average_precision_score de sckit-learn.
3. Pour une requête, tracez la courbe de rappel/précision en utilisant les fonctions de sckit-learn. Comparez la valeur de l'AP que vous avez coder avec l'aire sous cette courbe.
4. Le MAP est la moyenne des Average Precision (AP) de toutes les classes. Écrivez une fonction calculant le MAP.

2.3 Calcul des descripteurs

1. Programmer les BoW. Vous pourrez utiliser les cluster donnés dans les fichiers *centroids/K.txt* résultat d'un K-means.
2. Programmer les VLAD. Vous pourrez utiliser les cluster donnés dans les fichiers *centroids/K.txt* résultat d'un K-means.
3. Comparez les performances de vos descripteurs à celles des descripteurs fournis.
4. Étudiez l'influence du nombre de cluster K sur les performances du système.

3 Rendu et évaluation

Le travail est à réaliser en binôme (ou seul). Vous devrez rendre votre travail en utilisant le système Devoir de moodle. Pour cela, vous devez déposer une archive targz dans moodle avant le 15/02/2018 à 23h55.

Dans le cas présent cette archive doit contenir au moins un fichier *noms.txt* contenant les noms des étudiants ou étudiantes composant le binôme. Chaque étudiant(e) doit déposer au minimum ce fichier, qu'il ou elle soit seul(e), ou en binôme. Les autres fichiers (liste ci-dessous) ne seront déposés qu'une seule fois par binôme.

— les sources de vos programmes. L'ensemble des scripts vous ayant permit de réaliser ce TP ;
— un fichier *README.txt* indiquant clairement les instructions pour lancer le code ;
— un rapport. Vous présenterez précisément les notions mises en jeu dans ce TP, et les solutions que vous avez mis en place pour répondre aux problématique du sujet. Vous ferez également une analyse détaillée des résultats obtenus. Le format de rédaction pourra s'inspirer des articles de recherche mis en bibliographie.
— tous autres fichiers dont la présence est nécessaire pour faire fonctionner l'ensemble de votre réalisation ou pour en comprendre le fonctionnement.

Le non-fonctionnement du code rendu sera fortement pénalisé, vous devrez donc vérifier avant tout que se que vous rendez est pleinement fonctionnel .

La qualité et le contenu du rapport ainsi que le respect des consignes représenteront une partie importante de la note. Vous devez présenter en détail les problèmes mis en avant dans le sujet, décrire la solution que vous avez mis en place et analyser qualitativement et quantitativement les résultats obtenus. Une partie références citant correctement vos sources est attendu.

Références

[1] R.A. Baeza-Yates and B.A. Ribeiro-Neto. Modern information retrieval - the concepts and technology behind search, second edition. 2011.

[2] T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*, pages 487–493, Cambridge, MA, USA, 1999. MIT Press.

[3] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In Andrew Zisserman David Forsyth, Philip Torr, editor, *European Conference on Computer Vision*, volume I of *LNCS*, pages 304–317. Springer, oct 2008.

[4] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 3304–3311, jun 2010.

[5] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.

[6] J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek. Image Classification with the Fisher Vector : Theory and Practice. *International Journal of Computer Vision*, 105(3) :222–245, December 2013.

[7] J. Sivic and A. Zisserman. Video Google : A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, October 2003.