Master 2 Pro « Systèmes d'Information Sécurisés »

Année Universitaire 2009-2010

M2P SIS - UE INF22

Annotations de données multimodales, le cas des vidéos

Projet Encadré par Hervé GLOTIN

Réalisé par :

Guillaume FOOT,

Omar LARAKI,

Omar TAIF,

Yassin BIBI,

Ahmed SILI,

Julien NIZZOLI,

Aurélien COUVERT,

Thibaut DAILLY,

Frédéric BILLO,

Terence MASSONI,

Walid DRIDI,

Eric FILIPPINI

Table des matières

1 Introduction sur le projet IRIM/TRECID

Le projet IRIM (Indexation et Recherche d'Information Multimédia) a pour objectif d'encourager et d'aider les équipes de recherche françaises à participer aux campagnes d'évaluation TRECVID, en particulier en favorisant les collaborations entre les équipes et les participations conjointes. Description détaillée sur le site :

http://mrim.imag.fr/irim/

Responsable :

Georges Quenot (LIG)

On cherche les modèle génériques capable de retourner un topic afin de reconnaître ensuite ce même topic dans un groupe d'images non annotées.

Dans notre cas:

Sur une vidéo on sélectionne des sous-plans > un sous plan est un Topic (groupe d'images).

Dans un topic on sélectionne une image > on étiquette manuellement en donnant un sens reel (ex avion, school, hand, two people, texte, etc.)

Cette analyse devrait permettre d'identifier un type d'image qui correspond à une réalité.

Pour ce faire plusieurs axes de modélisation sont envisagés et distribués en features.

Chaque feature a été annotée manuellement.

L'ensemble des topics étudiés est regroupé dans un fichier Keylist.txt

nota accès au projet IRIM:

http://mrim.imag.fr/irim/

login :xxxx

Pwd: xxxx

http://mrim.imag.fr/irim/Wiki

Log: **xxx**

pwd:xxx

2 TRECVID (TREC Video Retrieval Evaluation)

Le TREC (lien web: http://trec.nist.gov) série de conférences parrainée par le National Institute of Standards and Technology (NIST) (lien web: http://www..nist.gov) Avec le soutien de d'autres agences du gouvernement américain. Le but de la série de conférences est d'encourager la recherche en recherche d'information en fournissant une grande collection de tests, une uniformisation des procédures de notation, et un forum pour organismes intéressés à comparer leurs résultats. En 2001 et 2002 la série TREC a parrainé une vidéo "piste" consacrée à la recherche de la segmentation automatique, l'indexation et la récupération basés sur le contenu de vidéo numérique. Commençant en 2003, cette piste est devenue une société indépendante d'évaluation (TRECVID)

L'objectif principal de l'évaluation TREC Video Retrieval (TRECVID) est de favoriser les progrès dans l'analyse fondée sur le contenu et l'extraction à partir de vidéos numériques, basée sur des paramètres d'évaluation.

TRECVID est une institution qui tente de modéliser le monde réel en s'appuyant sur des situations ou des

aspects importants qui le caractérise.

En 2006 TRECVID a achevé le deuxième cycle de deux ans consacré aux segmentation automatique, l'indexation et la récupération basés sur le contenu de Digital Video - nouvelles diffusés en anglais, arabe et chinois. Il a également complété deux années d'études pilotes sur l'exploitation des inédits vidéo ("rushes"). Quelque 70 groupes de recherche ont été fournis avec le TRECVID 2005-2006 nouvelles vidéos de diffusion et de nombreuses ressources créées par NIST et de la communauté TRECVID sont disponibles pour poursuivre la recherche sur ces données indépendamment de TRECVID.

Voir la rubrique « Les données historiques » sur le site web de TRECVID.

En 2007 TRECVID a commencé à explorer de nouvelles données (culturelle, le magazine des nouvelles, documentaire, et la programmation de l'éducation) et un contrôle additionnel nouvelle tâche -- Vidéos Rushes résumé.

En 2008, ce travail s'est poursuivi sauf que la limite de la détection a été augmentée de la détection de copies et de la détection des événements de surveillances .

TRECVID 2009 : essais des systèmes sur les tâches suivantes:

- surveillance événement de détection
- haute fonction de niveau d'extraction
- Recherche (interactifs, à commande manuelle assistée et / ou entièrement automatique)
- basé sur le contenu de détection de copie

L'objectif de ce projet est d'améliorer de manière importante ces méthodes et de s'approcher de l'état de l'art dans le domaine. Pour cela, il faut d'une part les optimiser en prenant en compte tous les facteurs importants et leur ajouter un certain nombre d'innovations comme:

•l'utilisation de concepts de niveau intermédiaire,

•la combinaison de méthodes génériques et spécifiques,

• l'apprentissage actif pour l'amélioration de la quantité et qualité de l'annotation servant à l'entraînement des systèmes.

Un des facteurs limitant est la puissance de calcul nécessaire. Il faut en effet entraîner et évaluer les systèmes sur plusieurs centaines de concepts et sur plusieurs dizaines de milliers d'images ou de plans vidéo. Il faut en outre faire cela en étudiant de multiples combinaisons de caractéristiques de bas et moyen niveau, de méthodes de classification et de méthodes de fusion.

2.1 Objectifs M2psis2009

Le point de départ une étude d'identification des concepts imposés par le NIST sur la base d'images étiquetées annotées à la main 'KEYFRAME', pour cela on dispose d'une panoplie de critères 'FEATURE' se basant sur : Audio , Texture,Couleur,Forme, et le traitement des points d'intérêts.

En Pratique , vue qu'on ne dispose pas des fichiers nécessaires à l'étiquetage et à la conversion des données de TrecVid 2008 (erreur sur le site IRIM), il a fallu se rabattre sur les seules données à notre disposition qui sont exploitables autrement dit la base de données 2007:

Train2007 : Base utilisée pour entraîner le modèle
Test2007 : Base utilisée pour tester le modèle

Remarques importantes :

Une classification rigoureuse nécessite l'utilisation de 3 bases d'images:

- Une première destinées à entrainer le modèle
- Une deuxième pour l'optimisation : concrètement la base de test à utiliser de nombreuse fois avce des paramètres différents afin de construire un modèle « idéal ».
- Et enfin une base de Test contenant des données totalement étrangères à la phase d'entrainement et d'optimisation . Sur ces données on applique le meilleur modèle avec les réglages les plus pertinents C'est le score obtenu sur cette base qui doit valider ou non un modèle

On pourrait croire que deux DATASETS suffisent dans le cadre d'une classification, un pour entraîner un modèle et l'autre pour le tester. Mais nous cherchons ici à concevoir des modèles **GENERIQUES** valables en toutes circonstances et pas uniquement sur le corpus documentaire utilisé lors des Tests ! En se limitant à deux DataSets le risque d'**overfitting** est important : des résultats trop optimistes par rapport à une situation réelle en raison de l'hyperadaptation du modèle aux données à tester..Toutefois faute d'avoir les fichiers nécessaires à l'étiquetage et à la conversion des données de TrecVid 2008 nous ne disposons que deux groupes utilisables : Dev2007et Test2007. Il aurait été envisageable de découper un des datasets en deux afin de créer un troisième groupe mais en raison d'un manque de temps cela n'a pas été effectué.

Toujours par manque de temps, au lieu de traiter les 20 topics prévus à l'origine, on se restreint à travailler sur les **3 topics les plus représentés dans la liste d'images étudiées et qui sont:**

- Hand,
- Street
- Two_people.

3 Liste des descripteurs (ou « features »)

Nous listons tout d'abord les grandes lignes des features qui ont été calculées au sein du consortium IRIM.

Lien web → http://mrim.imag.fr/irim/wiki/doku.php

LIP6-ETIS:

•Features : couleur et texture.

• Spécificité RETIN: optimisé par k-means à plusieurs niveaux.

• Filtres de Gabor 15-30 composantes par pixel puis clustring.

• Dictionnaires 100 à 200.

Eurecom:

• Couleur, texture, découpage par grille, quantification.

•Détecteurs, fusion.

•Corrélations entre concepts.

LABRI:

•Détection qualification de mouvement de caméra.

• Activité résiduelle, mouvement résiduel.

• Extraction des objets et multiobjets dont visages.

• Couleur / non couleur.

• Clustering variés.

• Visages par OpenCV et tracking.

CEA-LIST:

•Traitement linguistique du hollandais.

•Classifieur multiclasses par boosting (shared boosting amélioré).

•Features : histogrammes divers couleur, texture, ...

•Bags of SIFT.

•Détecteurs de concepts indoor/outdoor, ...

IRIT:

•Bags of SIFT + couleur locale.

•Segmentation en évènement sonores.

•Descripteurs audio à préciser.

LISTIC:

•Features couleur/texture.

•Détection des points d'intérêt spatiotemporels.

•Fusion texte/image.

•Fusion, interaction entre features.

GIPSA:

•- Couleur, texture, mouvement.

•- Séquencement pour évènements sportifs.

•- Attention visuelle.

•- Modèles de croyances transférables.

•Détection de junk frames.

LIF:

•- Descripteurs classiques couleur, texture, SIFT, ... sur grille.

•- Nombreuses méthodes de fusion.

•Descripteurs de niveau intermédiaire.

LIG:

•- LIF+,

•- Sélection d'images clé,

•- Flot optique dense,

•- Fusion par rang,

•Méthode d'optimisation de la fusion.

LSIS :

•- Descripteurs non conventionnels : Profile Entropy Features (PEF) décrit dans les papiers Glotin CIVR08 et ICICP09. Variante des descripteurs de Fourier (DF)

•- Descripteurs classiques : HSV, Gabor, EDGE

3.1 Détail des caractéristiques

LIG_random<k> (1 dim) : random vectors with different seeds, for baseline and dispersion analysis, k = 1-4.

LIG_random<k>0 (10 dims): random vectors with different seeds, for baseline and dispersion analysis, k = 1-4.

CEALIST_global_tlep (576 dims) : texture (local edge pattern [Che 2003]) + couleur (histo RGB 64 dims

CEALIST_global_cime (64 dims) : histogramme de 4-connexite pour 64 couleurs RGB
CEALIST_global_projection (400 dims) : forme decrite par projection horiz. et vert. des pixels en niveau gris

CEALIST_global_probe (164 dims) : co-ocurence dans l'espace couleur

CEALIST_global_cciv (345 dims) : histo couleur sur les pixels de même couleur dans régions > 5% image

CEALIST_global_pigment (125 dims) : histo couleur RGB

CEALIST_local : Descripteur: bag of SIFT

•Dimension : 5000 float

•Vocabulaire construit à partir de l'ensemble de la base

•Points d'intérêt : Harris Laplace

•Descripteur : SIFT

•Creation vocabulaire : K-Means

•Taille du vocabulaire : 5000

CEALIST_scribe : Contient 4 descripteurs “conceptuels”. Les valeurs sont entières (mais écrites en float selon la convention!)

•Dimension 1 : 1=photo 2=photo N&B 3=photo N&B colorisee 4=peinture 5=clipart 6=carte 0=indetermine

•Dimension 2 : 1=indoor 2=outdoor 0=indetermine

•Dimension 3 : 1=day 2=night 0=indetermine

•Dimension 4 : 1=urban 2=nature 0=indetermine

ETIS_global_<attr>[<type hist>]<taille dict> : Descripteurs histogrammes calculés pour différents attributs visuels et dictionnaires. <attr> = lab : couleurs L*a*b*, qw : norme coeffs ondelettes quaternioniques, 3 échelles. <type hist> = (rien) : histogramme calculé sur toute l'image, bic : 2 histogrammes pixels intérieur/bordure, m1x3 : 3 histogrammes sur 3 bandes verticales, m2x2 : 4 histogrammes sur les 4 quart de l'image. <taille dict> = nombre de mot dans le dictionnaire utilisé, parfois différent de la taille finale de l'histogramme. Par exemple pour un histogramme m1x3 avec un dictionnaire de 32 mots, l'histogramme a 3×32 = 96 valeurs.

Eurecom_surf : descripteur SURF (genre de SIFT) avec un vocabulaire de 500 mots.

GIPSA_AudioCS : nouveau calcul du centre spectral avec un échantillonnage différent

GIPSA_audio_intensity : intensité audio ramenée au pourcentage. Estime l'intensité du signal audio autour de la keyframe. La valeur est exprimée en pour cent sur l'ensemble de la vidéo.

GIPSA_AudioSpectro : profil spectral en 30 bandes sur une échelle Mel

GIPSA_AudioSpectroN : profil spectral normalisé en 30 bandes sur une échelle Mel (supprime les variations d'intensité entre les différents shots)

GIPSA_spectral_centroid : moyenne du centre spectral du signal audio. L'estimateur est obtenu en calculant le cente spectral sur plusieurs frames audio pour l'ensemble du shot, puis en prenant la valeur moyenne.

GIPSA_spectral_centroid_var : écart-type du centre spectral du signal audio. L'estimateur est obtenu en calculant le centre spectral sur plusieurs frames audio pour l'ensemble du shot, puis en prenant l'écart-type. Indicateur de la variabilité en fréquence du signal audio.

GIPSA_faces_<n> : nombre de visages sur la keyframe. <n> est un paramètre de sensibilité, une valeur élevée indique une meilleure précision et un moins bon rappel.

GIPSA_motion : quantité de mouvement résiduel. Estime la quantité de mouvement résiduel basé sur l'estimation de l'outil Motion2D. La valeur est calculée sur 7 images autour de la keyframe.

IRIT_BoW-SIFT : Descripteurs Bag-of-Words (TF-IDF / 250 mots) SIFT.

IRIT_BoW-Color : Descripteurs Bag-of-Words (TF-IDF / 250 mots) Couleur locale (histogramme de teinte sur la même régionn que le SIFT correspondant)

IRIT_MFCC-Avg-Avg : MFCC (16 dimensions) –> Moyenne sur segmentation en zones acoustiques homogènes –> Moyenne sur les plans TRECVid correspondants

IRIT_MFCC-Var-Avg : MFCC (16 dimensions) –> Variance sur segmentation en zones acoustiques homogènes –> Moyenne sur les plans TRECVid correspondants

IRIT_MFCC-Var-Min : MFCC (16 dimensions) –> Variance sur segmentation en zones acoustiques homogènes –> Minimum sur les plans TRECVid correspondants

IRIT_MFCC-Var-Max : MFCC (16 dimensions) –> Variance sur segmentation en zones acoustiques homogènes –> Maximum sur les plans TRECVid correspondants

IRIT_YIN : Concaténation (4 dimensions) de descripteurs issus du YIN

- Pourcentage (en durée) de zone pitchée sur segmentation en zones acoustiques homogènes
- Moyenne sur les plans TRECVid (1D)
- Moyenne du pitch sur les zones pitchées sur segmentation en zones acoustiques homogènes
- Maximum/Minimum/Moyenne sur les plans TRECVid (3D)

LABRI_BlackAndWhite_Mean : contient la détection des shots en Noir et blanc/niveaux de gris/cepia...

Une valeur par shot, dans[0.0; 1.0] : proche de 0.0 signifie une majorité de frames en couleur, proche de 1.0 signifie une majorité de frames en noir et blanc. Sur nos essais, ce détecteur semble fournir des Rappel et Precision autour de 95%.

LABRI_facesOCV_Mean : contient la détection des visages par OpenCV [Nous n'avons pas encore appliquer notre filtrage temporel] Une valeur par shot, qui est la moyenne du nombre de visages par frame dans le shot. Sur nos essais, ce détecteur semble fournir un Rappel autour de 42% et un Précision autour de 73%.

LABRI_varianceResidualMovement_Mean : contient la variance du mouvement résiduel. Deux valeurs par shot, en x et y; qui sont la moyenne sur chaque coordonnée des variances des frames dans le shot. Cet outil

s'appuie sur les vecteurs de mouvements du flux MPEG et est donc très dépendant de la qualité de celui-ci.

LEAR_bow_sift_<taille dict> : Vecteurs Bag Of SIFT Words pour k = 500, 1000, 2000, 4000, 8000 et 16000 générés par le LEAR (Hervé Jégou) ; voir la présentation de Matthijs Douze à la réunion du 4 juin 2008 : http://mrim.imag.fr/irim/active2008/reunion-2008-06-04/irim_080604_douze.pdf

LIF_Global : Descripteur de couleur et texture global à l'image. La couleur est représentée par un histogramme 3D dans l'espace RGB, où l'espace de couleur est discrétisé de façon à obtenir un histogramme de 4 x 4 x 4 dimensions. La texture est extraite via un banc de 40 filtres de Gabor sur 8 orientations et 5 échelles. Finalement, ces descripteurs sont normalisés pour former un vecteur de 104 dimensions.

LIF_MMCOL_2x2 : Descripteur de couleur contenant les deux premiers moments statistiques des compasantes R, G et B extraits dans une grille de 2 x 2 blocs. Ce descripteur forme un vecteur de 36 dimensions.

LIF_MMCOL_4x3 : Idem, mais grille de 4 x 3 ⇒ 108 dimensions.

LIF_MMCOL_8x6 : Idem, mais grille de 8 x 6 ⇒ 432 dimensions.

LIF_HISTCOL_2x2 : Descripteur de couleur décrit par un histogramme 3D dans l'espace RGB, extraits dans une grille de 2 x 2 blocs. Ce descripteur forme un vecteur de 108 dimensions.

LIF_HISTCOL_4x3 : Idem, mais grille de 4 x 3 ⇒ 324 dimensions.

LIF_HISTCOL_8x6 : Idem, mais grille de 4 x 3 ⇒ 1296 dimensions.

LIF_GABOR_2x2 : Descripteur de texture de 40 filtres de Gabor sur 8 orientations et 5 échelles, extraits dans une grille de 2 x 2 blocs. Ce descripteur forme un vecteur de 160 dimensions.

LIF_GABOR_4x3 : Idem, mais grille de 4 x 3 ⇒ 480 dimensions.

LIF_GABOR_8x6 : Idem, mais grille de 8 x 6 ⇒ 1920 dimensions.

LIF_HISTEDGE_2x2 : Histogramme d'orientations extrait d'une grille de 2 x 2 blocs. Chaque dimension correspond à la somme des magnitudes d'une orientation. Ce descripteur considère 50 orientations et forme un vecteur de 200 dimensions.

LIF_HISTEDGE_4x3 : Idem, mais grille de 4 x 3 ⇒ 600 dimensions.

LIF_HISTEDGE_8x6 : Idem, mais grille de 4 x 3 ⇒ 2400 dimensions.

LIF_LBP_2x2 : Descripteur “Local Binary Pattern”, extrait dans une grille de 2 x 2 blocs. Le descripteur LBP modifie la valeur d'un pixel selon les valeurs des pixels voisins pour capter des motifs de texture. Il forme un vecteur de 1024 dimensions.

LIF_SEMANTIC : Ce descripteur vise à modéliser les relations sémantiques entre les concepts par une approche sac-de-mots. Ce descripteur nécessite une phase d'apprentissage sur les blocs d'image. Pour cela, nous considérons chaque blocs d'image comme positifs relativement à un concept lorsque l'image est annotée positivement. Cette hypothèse est certe très forte mais peut être raisonnable pour certains concepts. Nous entraînons des modèles par concept au niveau bloc sur une partie de l'ensemble d'apprentissage puis classons l'ensemble des blocs, qui abouti à NB_BLOCS x NB_CONCEPTS scores de classification par image. Le descripteur sémantique est représenté par un histogramme de NB_CONCEPTS (=20 concepts TRECVID'08) dimensions où chaque dimension contient la somme des scores d'un concept sur tous les blocs.

LIF_Percepts : Descripteur intermédiaire construit à partir des scores de prédictions de 15 concepts locaux (ciel, verdure, mer, ...) obtenus sur une grille de 20 x 13 blocs et appris sur les corpus trec2003 et trec2005. Le vecteur résultant est de 3900 dimensions.

LIG_h3d64 : normalized RGB Histogram 4 x 4 x 4 ⇒ 64 dimensions.

LIG_gab40 : normalized Gabor transform, 8 orientations x 5 échelles ⇒ 40 dimensions.

LIG_hg104 : early fusion (concatenation) of LIG_h3d64 and LIG_gab40 ⇒ 104 dimensions.

LIG_opp_sift_har : bag of word, opponent sift with Harris-Laplace detector, generated using Koen Van de Sande's software ⇒ 4000 dimensions (384 dimensions par point détecté avant clustering ; clustering sur 535117 points issus de 1000 images choisies au hasard).

XLIM-SIC_qmvt_L<n> : chaque fichier contient un float par sous-plan et les noms font référence à la quantité de mouvement primaire (L1) ou secondaire (L2).

LSIS_PEF_150 : Profile Entropy Features (PEF) version 150 dimensions décrit dans les papiers Glotin CIVR08 et ICICP09
LSIS_PEF_45 : Profile Entropy Features (PEF) version 45 dimensions décrit dans les papiers Glotin CIVR08 et ICICP09
LSIS.DF : Variante des descripteurs de Fourier
LSIS.HSV : Histogramme HSV (7x3x3)
LSIS.EDGE : Histogramme de dimension 72 de la représentation spatial des différents types de contours
LSIS.GABOR : Histogramme des filtres de Gabor , 4 x 3 x 5 dimensions.

Outils :

Pour faire cette identification, on devait utiliser openCV qui est basé sur LIBSVM, mais pour éviter un travail trop lourd, on a préféré basculer sur SVM qui est basé aussi sur LIBSVM.

4 Topics

Chaque année, TRECVID dévoile 20 concepts, nommés Topics sur lesquels les machines devront être entraînées. Cette année, par manque de temps et de moyens, nous avons décidé de limiter notre étude à 3 topics. Ce choix n'a pas été fait aléatoirement, nous avons choisis les 3 topics qui avaient le plus d'images positives dans la base de shots. C'est à dire que nous avons travaillé sur des topics qui avaient de 800 à 1600 photos classées comme "bonnes" sur la base des 20000.

- HAND : (Shots sur lesquels apparaît clairement une main) On peut d'ores et déjà imaginer que la FORME est un critère appréciable pour détecter une main ainsi que sa texture, même si le port d'un gant peut vite nous prendre à défaut.

- TWO PEOPLE : (Shots sur lesquels apparaissent clairement deux personnes) Le topic TWO PEOPLE est sensiblement lié à HAND dans la recherche par similarité. En effet, la FORME peut être un bon argument de recherche, un visage est clairement défini par deux yeux, une bouche, et une personne par son corps, 2 bras, 2 jambes...

- STREET : (Shots sur lesquels apparait une rue)Incontestablement la FORME est mise en avant aussi par des lignes droites, parallèles, bien délimitées mais il ne faut pas oublier les COULEURS qui restent non négligeables lorsque l'on peut définir une route. On imagine différentes nuance de gris sur le bas d'une photo (goudron par exemple) qui s'éclaircit en dégradé vers le haut du shot (ciel, soleil...).

Bien sur, il existe d'autres topics comme par exemple, une classe, un bateau, un avion qui auraient été aussi intéressants à étudier. Seulement dans notre cas précis (images de l'année 2007), la base de shots disponibles n'était pas probante. En effet, par exemple pour le topic AIRPLANE, il n'y avait que 11 images positives dans la base sur les 20000 analysées. Dans ce cas là, les résultats auraient été faussés et d'une interprétation douteuse. En effet, le modèle aurait eu tendance à classer l'image comme non élément du topics étant donné

le faible taux de reconnaissance d'un avion (11 / 20 000).

4.1 PréTraitement des Données TRECVID

Il a fallu convertir les données du IRIM CF (8.1 Annexe_1_Pré Traitement des Données TRECVID)

5 Machines de classification automatiques

5.1 classification Aléatoire

Un bon point de comparaison pour nos modèles est de se référer à un modèle aléatoire. Intuitivement on s'attend à une courbe «plate» dont l'ordonnée (la précision) est au niveau de la proportion d'images positives pour le concept cherché dans le DATASET. En générant des fichiers de résultats aléatoires avant de leur appliquer la fonction Octave calculant l'Average Precision on peut effectivement conjecturer cela. Les courbes obtenues avaient l'allure attendue (un exemple parmi beaucoup d'autres ci dessus) et l'Average Precision moyenne pour 200 itérations du procédé s'approchaient systématiquement de la proportion de documents positifs pour un topic donné.

5.1.1 Classifieur majoritaire (aléatoire)

Autre méthode:non employée.

5.2 Cosinus

Machine naïve de classification qui se contente de faire le produit scalaire des deux vecteurs passés en paramètres (ici le fichier train obtenu à partir du fichier Torch en lui enlevant les étiquètes et la première ligne, et le fichier test obtenue de la même manière à partir du fichir test du même topic et de la même feature), et en le divisant par le produit des normes des deux vecteurs ce qui donne un scalaire.

5.3 Torch

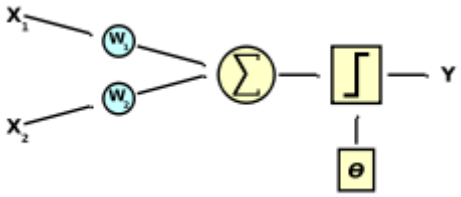
La machine MLP (Multi Layer Perceptron) développée par TORCH est un réseau de neurones artificiels.

Site Web :http://www torch.ch/

Le perceptron peut être vu comme le type de réseau de neurones le plus simple. Un réseau de neurones artificiel est un modèle de calcul dont la conception est très schématiquement inspirée du fonctionnement de vrais neurones (humains ou non). Les réseaux de neurones sont généralement optimisés par des méthodes d’apprentissage de type statistique, si bien qu’ils sont placés d’une part dans la famille des applications statistiques, qu’ils enrichissent avec un ensemble de paradigmes permettant de générer de vastes espaces fonctionnels, souples et partiellement structurés, et d’autre part dans la famille des méthodes de l’intelligence artificielle qu’ils enrichissent en permettant de prendre des décisions s’appuyant davantage sur la perception que sur le raisonnement logique formel. Un perceptron est un classifieur linéaire. Ce type de réseau neuronal ne contient aucun cycle (en anglais feedforward neural network). Le perceptron a été inventé en 1957 par Franck Rosenblatt au Cornell Aeronautical Laboratory, inspiré par la théorie cognitive de Friedrich Hayek et celle de Donald Hebb.

Dans sa version simplifiée, le perceptron est mono-couche et n'a qu'une seule sortie à laquelle toutes les entrées sont connectées. Les entrées et la sortie sont booléennes. Le potentiel post-synaptique $\sum W_i e_i$ où Wi est le poids de l'entrée ei. La fonction d'activation est la fonction de Heaviside (la fonction signe est parfois utilisée) $H(x) = \begin{cases} 0 & \text{si} \quad x < 0 \\ 1 & \text{si} \quad x \geq 0 \end{cases}$, avec $x = \sum W_i e_i - b$. Ici, b définit le seuil à dépasser pour que la sortie soit à 1. On trouve fréquemment une variante de ce modèle de base dans laquelle la sortie prend les valeurs -1 et 1 au lieu de 0 et 1.

Exemple de Perceptron avec 2 entrées et une fonction d'activation à seuil :



Les réseaux de neurones, en tant que système capable d'apprendre, mettent en œuvre le principe de l'induction, c'est-à-dire l'apprentissage par l'expérience.

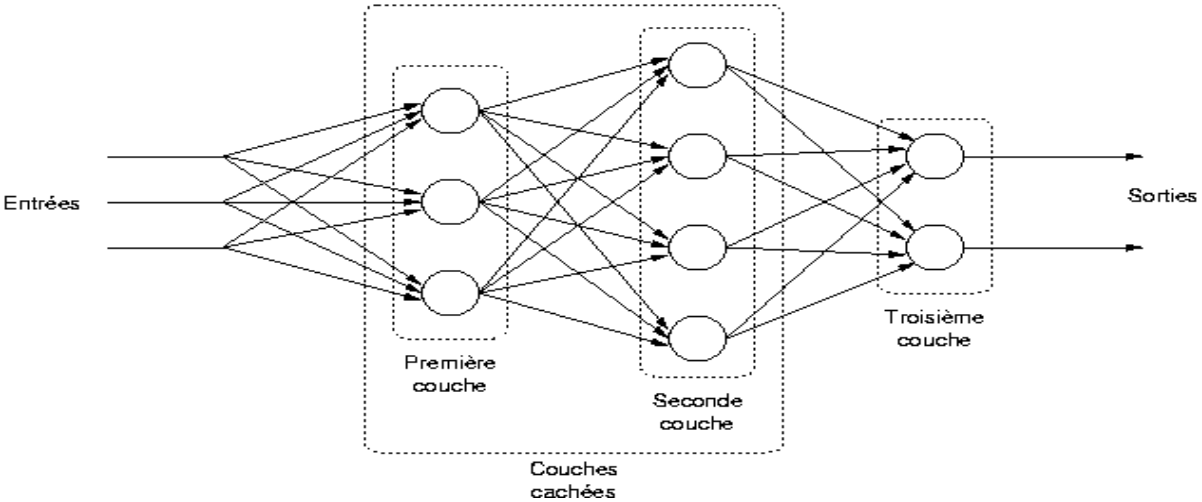
Par confrontation avec des situations ponctuelles, ils infèrent un système de décision intégré. Il s'agit ici de la seule couche cachée de notre machine.Grâce à leur capacité de classification et de généralisation, ils sont généralement utilisés dans des problèmes de nature statistique.

Le perceptron est organisé en plusieurs couches. La première couche est reliée aux entrées, puis ensuite chaque couche est reliée à la couche précédente. C'est la dernière couche qui produit les sorties du MLP. Les sorties des autres couches ne sont pas visibles à l'extérieur du réseau, et elles sont appelées pour cette raison couches cachées.

Dans notre cas, nous travaillons sur des machines déterministes. L'objectif étant de savoir si une image traîtée appartient ou non à un topic donné, notre machine aura un nombre de classe en entrée équivalent à la dimension du feature traîté.

A l'inverse, le nombre de classe de sortie reste fixe à 2 quelque soit le topic, quelque soit le feature car notre machine est déterministe et la seule information qui nous interesse est un jugement statistique de l'appartenance ou non au topic. Il y aura donc une classe 'VRAI' avec la probabilité que le shot appartienne au topic et une classe 'FAUX' avec les probabilités qu'elle n'appartienne pas au topic.

De manière plus générale, un Perceptron multi-couches peut faire appel à un nombre beaucoup plus élevé de couches cachées. Cette architecture particulière ne nous a pas servie directement dans notre démarche si ce n'est dans l'apprentissage du fonctionnement du MLP. En voici un bref exemple :



5.4 Machines SVM (support vector machine)

Sont un ensemble connexe d'apprentissage supervisé méthodes utilisées pour classement et régression. Une machine à vecteurs de support construit un hyperplan ou un ensemble d'hyperplans dans un haut ou un espace de dimension infinie, qui peut être utilisé pour la classification, de régression ou d'autres tâches. Intuitivement, une bonne séparation est réalisée par l'hyperplan qui a le plus grand écart à la formation les points de données le plus proche d'une classe (ce qu'on appelle la marge fonctionnelle), car en général, plus la marge est faible, meilleure est la généralisation d'erreur du classificateur. SVMs appartiennent à une famille généralisée classificateurs linéaires.

5.4.1 Libsvm

C'est la plus classique, machine utilisée dans OPENCV & SVM.Un petit guide est disponible à l'adresse suivante : http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf

5.4.2 Lssvm

Classification de grands ensembles de données avec un nouvel algorithme de SVM La méthode de régression Least Squares-Support Vector Machines (LS-SVM) est une variante des outils statistiques SVM. Ces derniers sont très populaires dans la communauté de l'apprentissage automatique étant donnée sont caractère deterministe.

Le nouvel algorithme de boosting de Least-Squares Support Vector Machine (LS-SVM) que nous présentons vise à la classification de très grands ensembles de données sur des machines standard. Les méthodes de SVM et de noyaux permettent d'obtenir de bons résultats en ce qui concerne la précision mais la tâche d'apprentissage pour de grands ensembles de données demande une grande capacité mémoire et un temps relativement long. Utilisant les atouts de Matlab (calcul matriciel entre autre) mais aussi ses inconvénients (impossible d'utiliser des grosses données et temps de calcul assez long).

Voici le liens du site officiel de cet algorithme: http://www.esat.kuleuven.be/sista/lssvmlab/

5.5 Linearsvm

*5.5.1 **Le classifieur lineaire***

Dans le domaine de Machine Learning, L'objectif de classement est d'utiliser les caractéristiques d'un objet à identifier la classe du groupe d'appartenance. Un linéaire classificateur atteint cet objectif en rendant une décision de classification basé sur la valeur d'une combinaison linéaire des caractéristiques.

Définition:

Si le vecteur caractéristique d'entrée au classificateur est un vecteur de réel, le niveau de sortie est donné par la fonction décrite ci dessous:

$$y = f(\vec{w} \cdot \vec{x}) = f\left(\sum_j w_j x_j\right),$$

ou

$\vec{w}$ est un véritable vecteur de poids et f est une fonction qui convertit les produits scalaires des deux vecteurs dans la sortie désirée. Le vecteur de poids qui est appris à partir d'un ensemble d'échantillons d'apprentissage étiquetés. Souvent f est une fonction simple qui relie toutes les valeurs supérieures à un certain seuil à la première classe et toutes les autres valeurs de la deuxième classe. A plus complexe f pourrait donner la probabilité qu'un objet appartient à une certaine classe. Pour un problème de classification à deux classes, on peut visualiser le fonctionnement d'un classificateur linéaire comme la séparation d'un

espace de grande dimension avec une entrée hyperplan:

Tous les points d'un côté de l'hyperplan sont classés comme "oui", tandis que les autres sont classés comme "non".

Un classificateur linéaire est souvent utilisé dans des situations où la vitesse de la classification est un problème, car il est souvent le plus rapide des classificateurs, surtout lorsque les points sont est clairsemés. Cependant, l'arbre de décision peut être plus rapide. Ainsi, les classificateurs linéaires sont souvent performants dans le cas d'un nombre de dimensions élevé, comme dans le cas de classement des documents, où chaque élément analysé est généralement le nombre d'occurrences d'un mot dans un document (voir document terme matrice). Dans de tels cas, le classificateur doit être bien régularisé. Cette machine est donc déterministe qui sépare les données 'oui' 'non' suivant un plan ce qui intéressant c'est de déterminer la distance des données par rapport à ce plan. C est donc cette distance par rapport au plan qui déterminera la qualité des mesures retournées par le modèle.

5.5.2 Liblinear-svm v 1.5

Ce modèle utilise différents paramètres employés lors de la phase d'entrainement. Choix, faire varier uniquement le paramètre 'c' (c=cost) ;n'ayant pas trouvé d'informations pertinentes sur les autres paramètres je me suis donc lancé dans une expérimentation assez exhaustive:

•.Etude de l'incidence du paramètre 'c' sur le résultat retourné.

Lors de premiers essais les résultats étaient tous de l'ordre de 96.6% . Donc le modèle renvoyait une probabilité de reconnaître des valeurs fausses 'non' (0) .L'utilisation d'Octave avec un calcul de rappel précision a permis d'obtenir une pertinence plus fine du modèle.

5.6 Iksvm:

C'est une machine basée sur libsvm et faisant des calculs binaires sur les vecteurs permettant un gain de temps remarquable lors de l'apprentissage instantané d'un feature de 250 dimensions.(Voir annexe compiler et utiliser iksvm.)

Pour contourner le problème du temps, on a essayé de compiler d'utiliser une nouvelle machine basé sur svm développée en c++ avec des dépendances matlab,cette dernière fait un travail sur les vecteurs au format binaire ce qui simplifie le calcul et réduit son temps sans négliger que les resultats obtenue pour mon meilleur feature étaient meilleur que pour svm classique, on a constaté un gain entre 1% et 2%.
Vu que cette machine est assez récente des problèmes de compilations et d'adaptation à la plateforme ont été rencontrés mais on a pu y remédier et lancer le test sur le meilleur feature pour les 3 topics, en modifiant le makefile, et en créant un programme C qui transforme les fichiers TORCH au format que demande cette machine.
Malheureusement le temps passé à corriger toutes les erreurs et à faire les réglages de cette machine avant de l'utiliser ne nous a pas permis de faire plus de test, cependant elle reste plus efficace que libsvm et aussi rapide que svm-linéraire, tout en offrant un choix d'option aussi large que ceux de libsvm avec des sorties paramétrables à souhait.

6 Performances

Après quelques essais nous nous sommes vite rendus compte que les résultats des prédictions étaient sans valeur car correspondant systématiquement au pourcentage de 0 parmi les étiquettes des images (oscillant entre 91% et 97%). En effet le système prédit systématiquement la classe 0 ! Afin de juger nos résultats de manière plus pertinente nous nous sommes intéressés aux probabilités d'appartenance aux classes. Une option particulière existant sur la plupart des machines permet de produire un fichier de résultat comportant deux colonnes représentant la probabilité d'appartenance à la classe 0 et la probabilité d'appartenance à la classe 1, c'est cette dernière colonne qui nous intéresse.

On récupère la colonne contenant la probabilité d'être un 1, donc un shot pertinent pour le concept cherché et on la trie par ordre décroissant : les images considérées comme le plus proche du concept recherché seront donc tête de la liste nouvellement crée.Y faire correspondre les étiquettes réelles permet de tracer une courbe de rappel/précision.

Rappel = Retournés et Pertinents / Pertinents = Proportion de retournés dans les pertinents

Précision = Retournés et Pertinents / Retournés = Proportion de pertinents dans les retournés

L'aire sous la courbe constitue la mesure de l'efficacité de nos modèles. Deux mesures ayant un intérêt pour juger de la qualité d'un modèle en pratique ont été ajoutés . Il s'agit de la précision après voir retourné 10 documents et 100 documents : un modèle même très médiocre dans l'absolu mais qui serait très solide sur quelques dizaines d'éléments en tête de liste reste utilisable puisque l'utilisateur moyen d'un système de recherche d'images va probablement parcourir uniquement les premières pages de réponses.La courbe tracée est sauvée au format png, les 3 mesures évoquées sont inscritesdirectement sur l'image.

6.1 Average précision / Topics (%)

Ci-après un tableau récapitulatif des performances par topic et par machine.

Les descripteurs utilisés dans les fusion sont préfixés de deux étoiles rouges '**'

** Descripteurs utilisés dans les fusions

	HAND								
	Dims	ALEATOIRE	COSINUS	MLP	LIBSVM	LSSVM	LINEAR SVM	IKSVM	MOYENNE
LABRI_faces	8		2,41	4,5	2,94	3,62			3,45
**LIF_SEMANTIC	20		5,01	7,5					6,26
LSIS_PEF45	45		2,12	5,61	4,19				3,97
LSIS_GABOR	60		2,82		3,32	5,03			3,72
LIF_HISTCOL_2x2	108		1,19			4,12			2,66
**LSIS_PEF150	150		3,84	5,99	4,94	3,73	7,71		5,24
LIF_GABOR_2x2	160	3,78	2,5	5,84		3,68			4,01
IRIT_BoW-Color	250		3,8		4,99	3,28			4,02
**IRIT_BoW-SIFT	250		5,44	5,89	15	3,11	6,71	16,01	8,69
LIF_GABOR_4x3	480		1,08	5,71	3,44	4,01	6,56		4,16
**CEALIST_global_tlep	576		2,52	6,83			10,81		6,72
LIF_GABOR_8x6	1920		3,35	6,03	5,53				4,97
CEALIST_local	5000		1,07	5,83					3,45
moyenne sur la sélection				*5,86*	*7,79*	*3,62*	*6,99*		

	STREET								
	Dims	ALEATOIRE	COSINUS	MLP	LIBSVM	LSSVM	LINEAR SVM	IKSVM	MOYENNE
LABRI_faces	8		0,82	3,02	1,56	2,54			3,13
**LIF_SEMANTIC	20								NC
LSIS_PEF45	45		8,15	14,74	12,13				11,67
LSIS_GABOR	60		4,32		6,07	6,32			5,57
LIF_HISTCOL_2x2	108		2,12			4,13			3,13
**LSIS_PEF150	150		9,13	13,87	12,75	5,07	15,69		11,3
LIF_GABOR_2x2	160	7,71	3,8	12,17		4,46			6,81
IRIT_BoW-Color	250		1,5		2,65	1,72			1,96
**IRIT_BoW-SIFT	250		3,58	6,6	15,43	8,62	12,03	16,86	10,52
LIF_GABOR_4x3	480		7,3	9,53	8,32		3,48		7,16
**CEALIST_global_tlep	576		2,53	11,86					7,2
LIF_GABOR_8x6	1920		7,62	9,65	9,55				8,94
CEALIST_local	5000		6,02	8			8,41		7,48
moyenne sur la sélection				*11,74*	*13,44*	*6,85*	*13,86*		

	TWO_PEOPLE								
	Dims	ALEATOIRE	COSINUS	MLP	LIBSVM	LSSVM	LINEAR SVM	IKSVM	MOYENNE
LABRI_faces	8		7,27	22,61	7,16	14,9	7,69	8,18	10,02
**LIF_SEMANTIC	20		,						NC
LSIS_PEF45	45		6,02	13,09	10,04				9,72
LSIS_GABOR	60		10,22		14,12	9,88			11,41
LIF_HISTCOL_2x2	108		5,03			9,01			7,02
**LSIS_PEF150	150		9,15	11,68	13,23	9,84	14		11,58
LIF_GABOR_2x2	160	2,32	10,28	12,55		9,97			10,93
IRIT_BoW-Color	250		6,37		9,97	9,12			8,49
**IRIT_BoW-SIFT	250		10,96	10,88	23,85	8,68	12,03	25,85	15,38
LIF_GABOR_4x3	480		7,55	13,87	13,28		13,58		12,07
**CEALIST_global_tlep	576		5,02	11,09		8,76			13,58
LIF_GABOR_8x6	1920		11,76	16,6	14,89				14,42
CEALIST_local	5000		4,38	10,88					7,63
moyenne sur la sélection				*15,06*	*14,75*	*11,14*	*11,24*		

6.2 Moyenne des résultats obtenus sur Average Precision par Topics

	HAND	STREET	TWO_PEOPLE
LABRI_faces	3,45	3,13	10,02
LIF_SEMANTIC	6,26	NC	NC
LSIS_PEF45	3,97	11,67	9,72
LSIS_GABOR	5,03	5,57	11,41
LIF_HISTCOL_2x2	2,66	3,13	7,02
LSIS_PEF150	5,24	11,3	11,58
LIF_GABOR_2x2	4,01	6,81	10,93
IRIT_BoW-Color	4,02	1,96	8,49
IRIT_BoW-SIFT	8,69	10,52	15,38
LIF_GABOR_4x3	4,16	7,16	12,07
CEALIST_global_tlep	6,72	7,2	13,58
LIF_GABOR_8x6	4,97	8,94	14,42
CEALIST_local	3,45	7,48	7,63

Base d'images	
Topic	Nombre
Hand	814
Two_People	661
Street	500

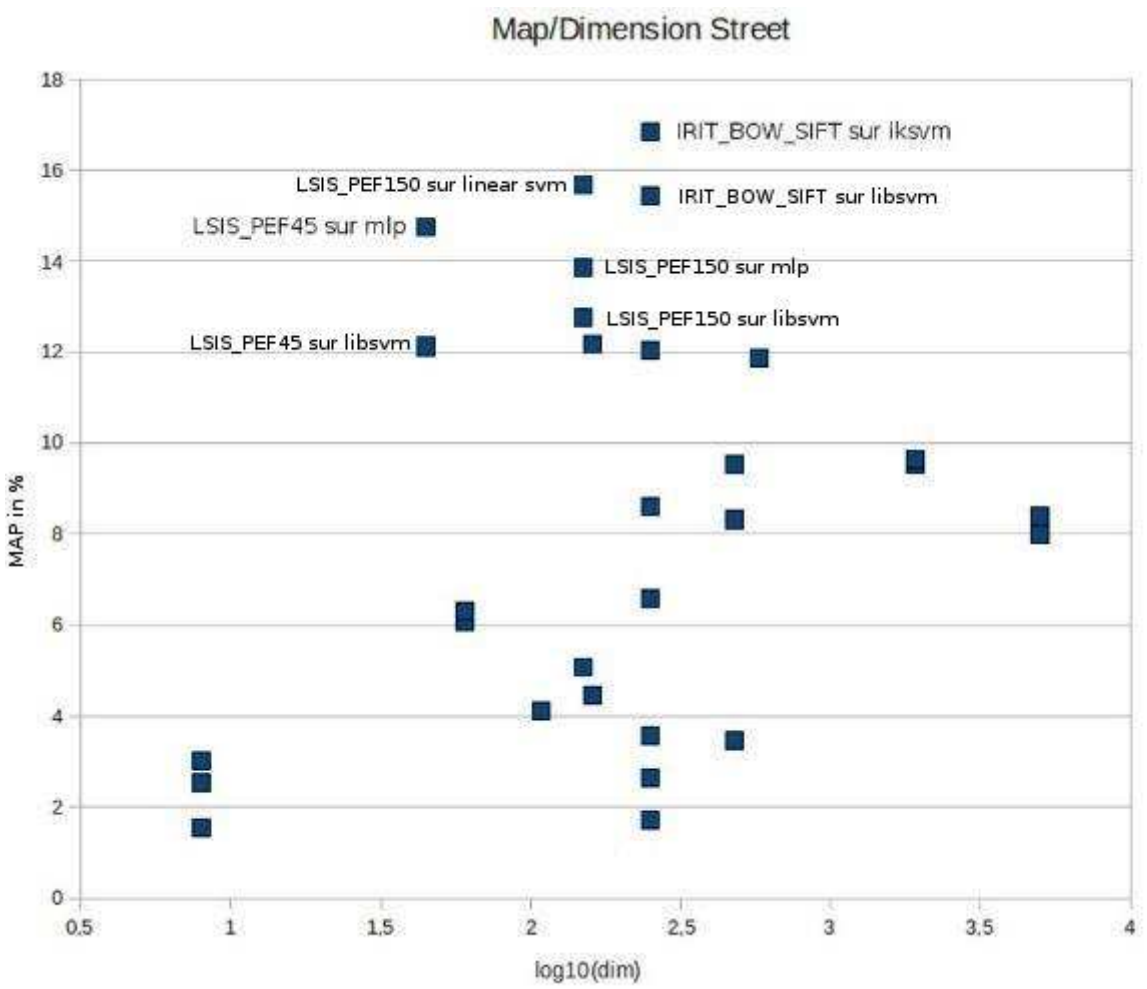
Notre étude limitée aux trois topics « baseimages » (tableau ci dessus) soit 1975 images une petite partie des données 2007 qui contiennent 21 532 images.

Au regard des résultats précédents, il ne suffit pas d'avoir des descripteurs de très grosse dimension pour avoir les meilleurs score.

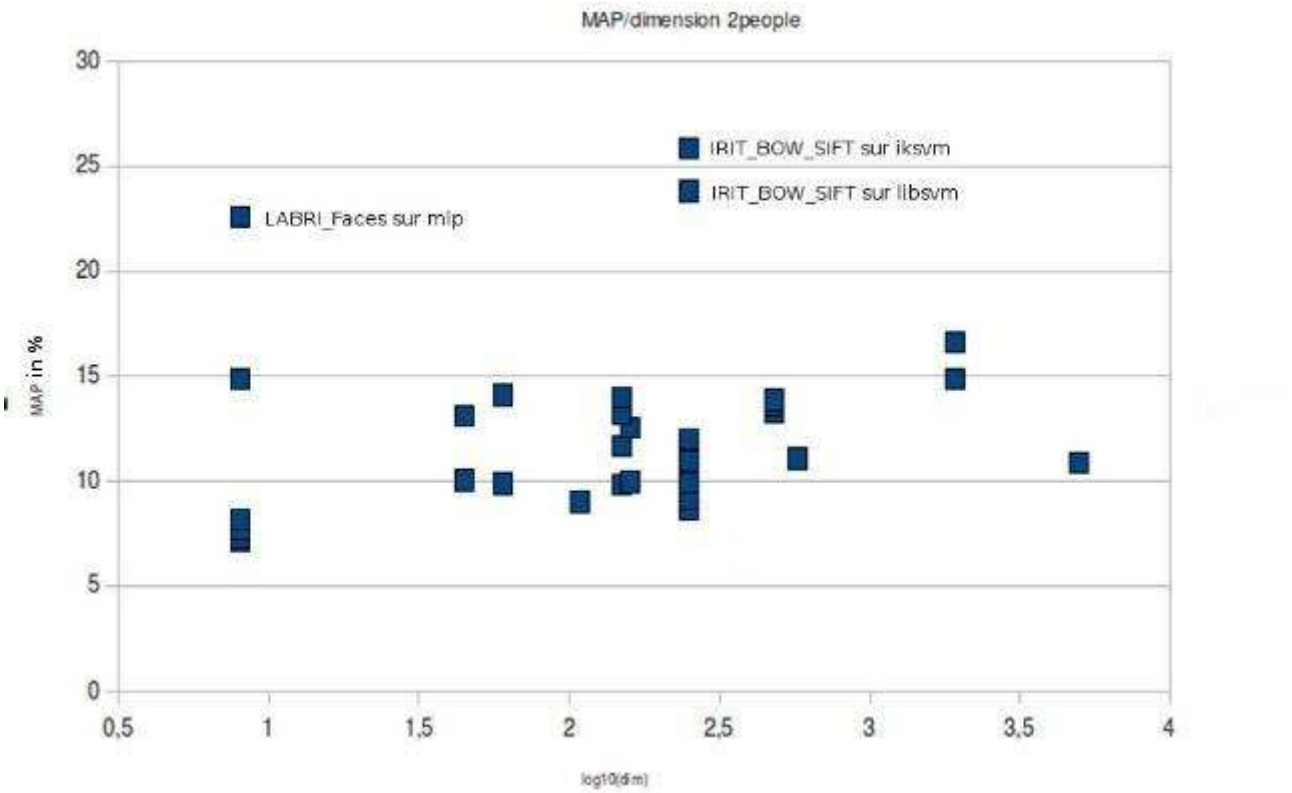
IRIT_BoW-SIFT est de dimension 250 et il est bien meilleurs que certains autres de plus grosses dimensions

Ainsi IRIT_BoW-SIFT semble être un bon candidat au vu de sa faible taille et de ses bonnes performances quelque soit les Topic utilisés (parmi les trois ci dessus) .

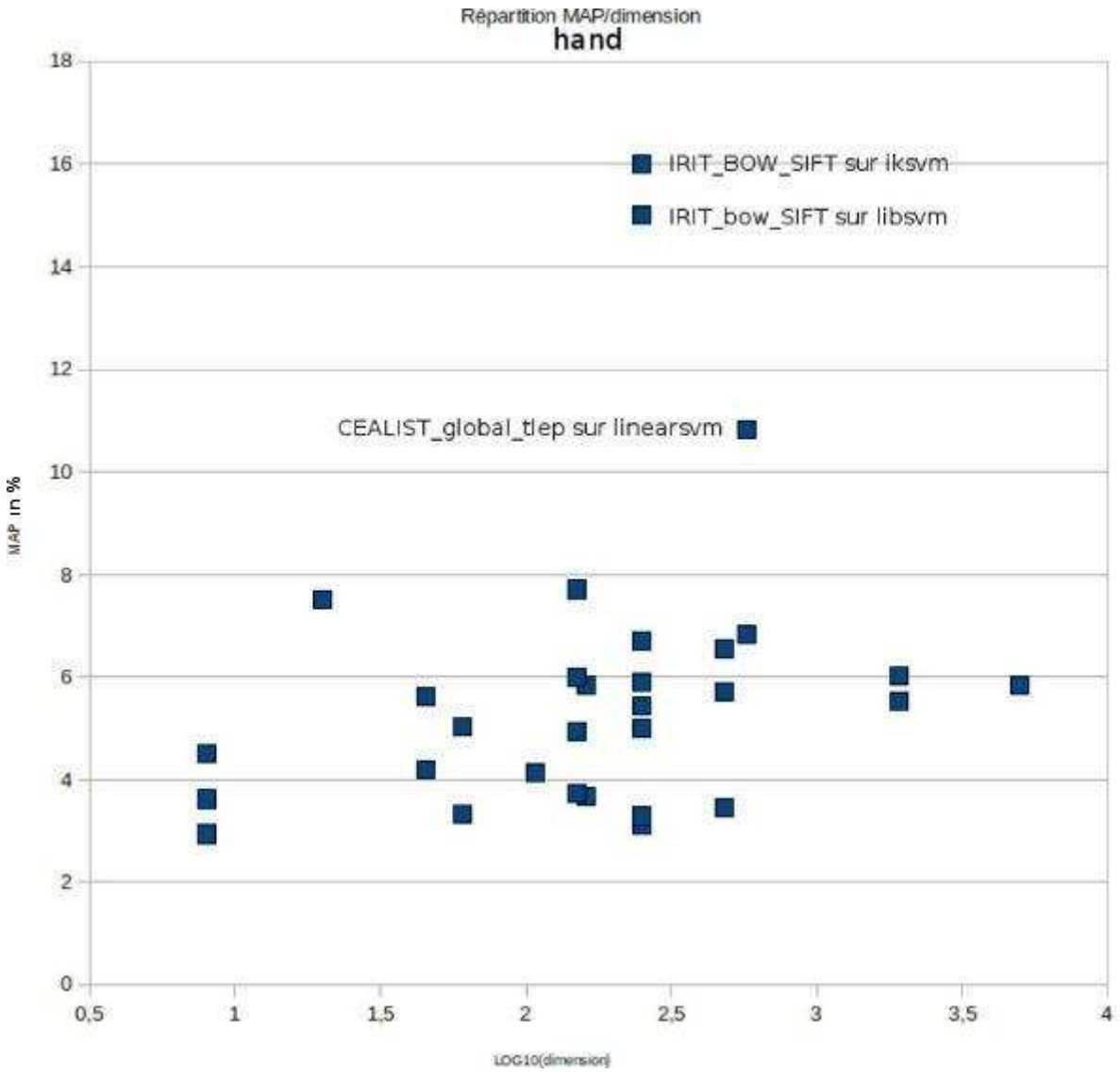
6.3 Courbe générale définie par : MeanAveragePrecision / log(Dimension)



Ci-dessus le nuage de points des résultats toutes machines confondues pour le topic Street. La machine iksvm a le meilleur résultat mais mlp obtient également des résultats correctes sur une feature de plus petite dimension.



Ci-dessus le nuage de points des résultats toutes machines confondues pour le topic Two people. La machine iksvm obtient encore les meilleurs résultats, mais ici l'intérêt penchera plus pour mlp car il obtient de très bons résultats sur une feature de très petite dimension.



Ci-dessus le nuage de points des résultats toutes machines confondues pour le topic Hand. Les machines libsvm et iksvm se démarquent clairement et ce sont les seuls résultats intéressants, les autres ne dépassant pas les 10%.

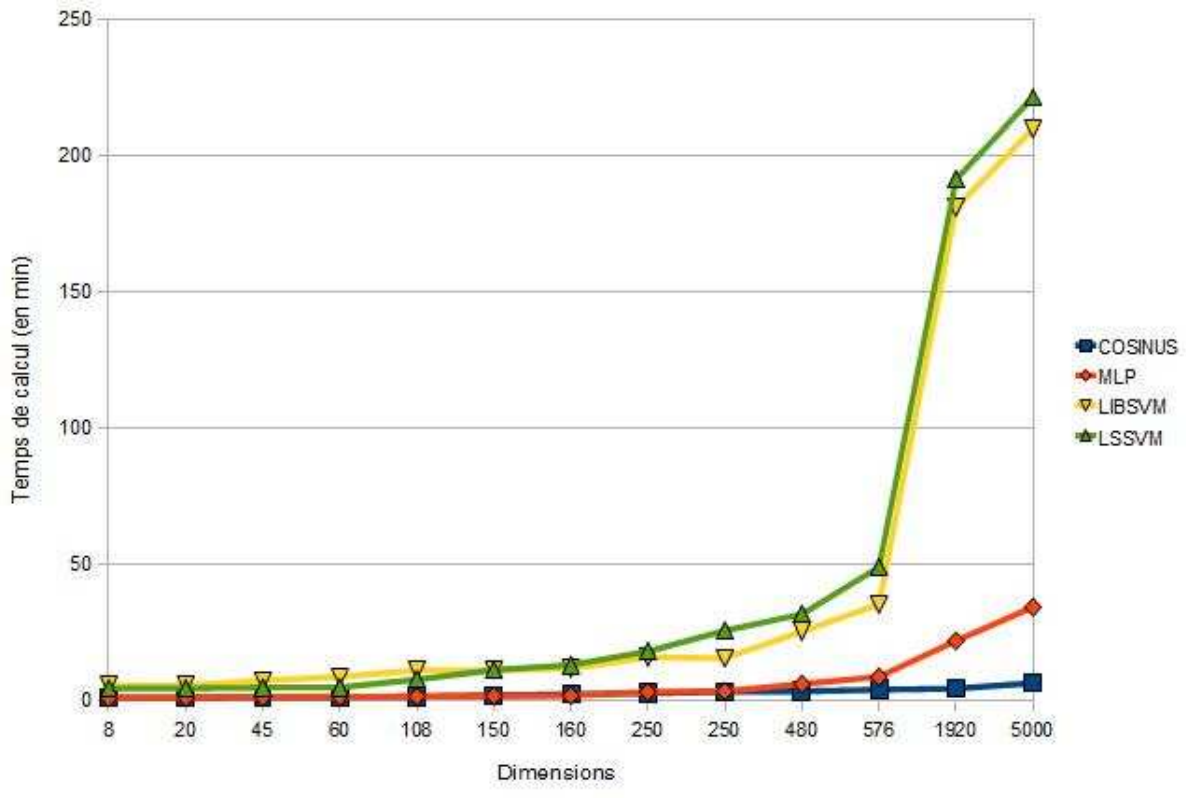
Pour conclure sur ces schémas, les machines libsvm et iksvm sont les plus performantes. Mais il faut prendre en compte également mlp qui est presque aussi efficace sur des features de plus petite dimensions.

6.4 Analyse de la moyenne des temps de calcul / machine et par feature

Le tableau ci dessous reflète la moyenne des temps de calculs lors de l'entrainement des modèles sur les données Train2007.

		Temps moyen en minutes / Machine			
FEATURES	**Dims**	**COSINUS**	**MLP**	**LIBSVM**	**LSSVM**
LABRI_faces	**8**	0,8	0,82	5,12	4,23
LIF_SEMANTIC	**20**	0,9	0,92	5,32	4,31
LSIS_PEF45	**45**	1	1,05	7,13	4,53
LSIS_GABOR	**60**	1,1	1,12	8,54	4,64
LIF_HISTCOL_2x2	**108**	1,3	1,3	10,89	7,52
LSIS_PEF150	**150**	1,7	1,48	11,01	11,02
LIF_GABOR_2x2	**160**	2,1	1,59	11,52	12,87
IRIT_BoW-Color	**250**	2,5	2,95	15,62	17,85
IRIT_BoW-SIFT	**250**	3	3,25	15,4	25,44
LIF_GABOR_4x3	**480**	3,2	5,88	25,09	31,54
CEALIST_global_tlep	**576**	3,8	8,51	35,16	48,78
LIF_GABOR_8x6	**1920**	4,2	21,73	180,87	191,25
CEALIST_local	**5000**	6,2	34,12	209,77	221,19

Le graphe ci dessous permet de visualiser le rapport entre les temps de calcul et la dimension de chaque Feature et ce pour les quatre machines utilisées (Cosinus, MLP, LIBSVM et LSSVM)



On peut classer les machines en deux catégories « rapides » et « lentes »:

- Machines « rapides » (inférieurs à 40 minutes): **LIBSVM et LSSVM**

On remarque une faible incidence de la dimension des features sur le temps d'exécution.

- Machines « lentes » : **MLP et Cosinus**

Le terme lent est utilisé afin de différencier les machines .En fait les machines «lentes » ont un comportement qui varie avec la dimension des features. On note un temps d'exécution inférieur à 40 minutes pour les features qui ont des dimensions comprises entre 8 et 576. Le temps de calcul augmente de façon exponentielle pour les features de dimension supérieures à 576 on atteint rapidement 220 minutes soit un temps 5 fois plus élevé.

Notre étude démontre que (voir chapitre 6.3) les machines classées « lentes » ou « rapides » peuvent donner malgré tout des résultats très performants.

7 Fusion de modèles

Types de fusion considérés

Nous disposons de nombreux modèles plus ou moins efficaces, l'objectif est d'en créer de nouveaux à partir de ceuxci. Pour ce faire on va se baser sur les fichiers résultats obtenus pour en construire un qui serait la fusion des autres, on utilisera 4 types de fusion :
•Maximum : le vecteur colonne construit aura pour chaque ligne une probabilité d'être un 1 équivalente à la plus forte probabilité parmi les modèles à fusionner pour la ligne courante.
•Minimum : analogue au maximum mais on aura pour chaque ligne la probabilité minimum.
•Moyenne : chaque composante du vecteur construit vaudra la moyenne des probabilités obtenues pour la ligne courante pour les modèles à fusionner.
•Produit : chaque composante est le produit des composantes à fusionner

Des scripts Octave ont été écrits pour chaque méthode. A partir d'un tableau contenant les chemins des fichiers de résultats des modèles à fusionner, ainsi que du fichier étiquette correspondant au topic étudié. Ils construisent les courbes rappel/precision des modèles de fusion classés par nombre de modèles considérés. Ces derniers étant ordonnés du plus performant au moins performant.
On construit ensuite pour chaque topic la courbe d'évolution de la MAP en fonction du nombre de modèles fusionnés.

7.1 Topic Hand

Liste des modèles fusionnés :

•IRIT_BoW-SIFT (libsvm)

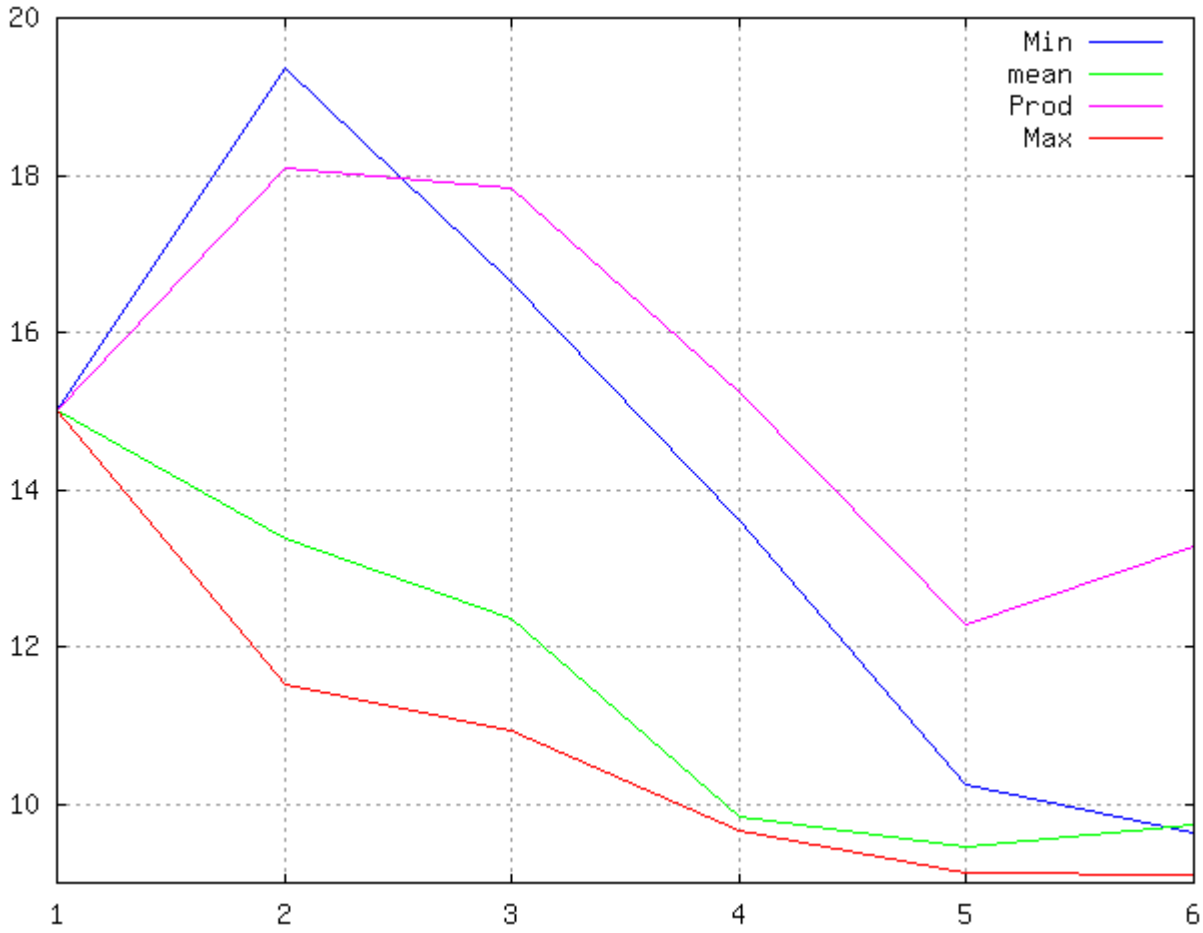
•CEALIST_global_tlep (linear SVM)

•LSIS_PEF150 (linear SVM)

•LIF_SEMANTIC_(128-128) (octave)

•CEALIST_global_tlep_(64-128) (octave)

•IRIT_BoW-SIFT (linear SVM)



En abscisse le nombre de modèles fusionnés et en ordonnée l'avérage précision en %

7.2 Topic Street

Liste des modèles fusionnés :

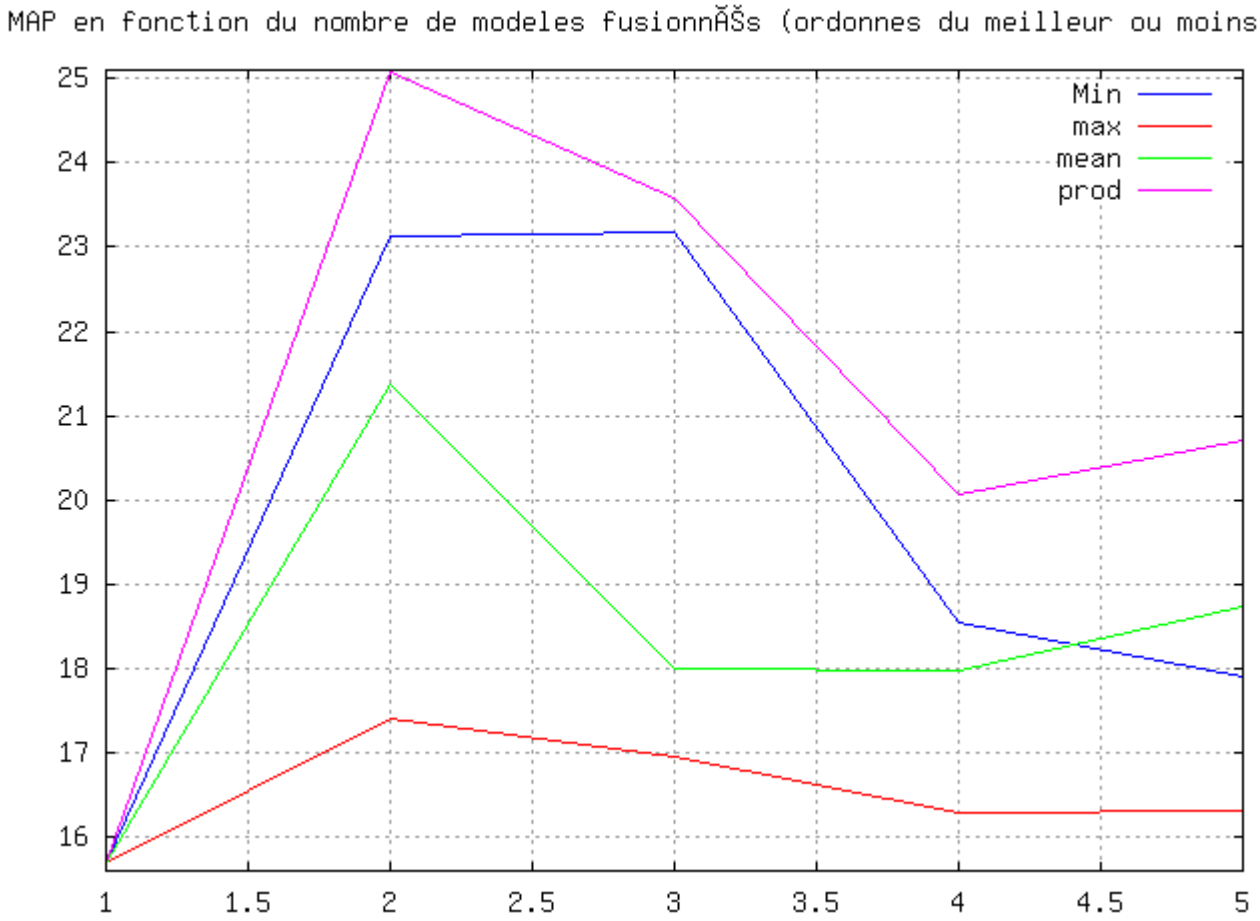
•LSIS_PEF150 (linear SVM)

•IRIT_BoW-SIFT (libsvm)

•LSIS_PEF45_(32-32) (octave)

•LSIS_PEF150_(128-128) (octave)

•LSIS_PEF150 (libsvm)



En abscisse le nombre de modèles fusionnés et en ordonnée l'avérage précision en %

7.3 Topic Two People

Liste des modèles fusionnés :

•IRIT_BoW-SIFT (libsvm)

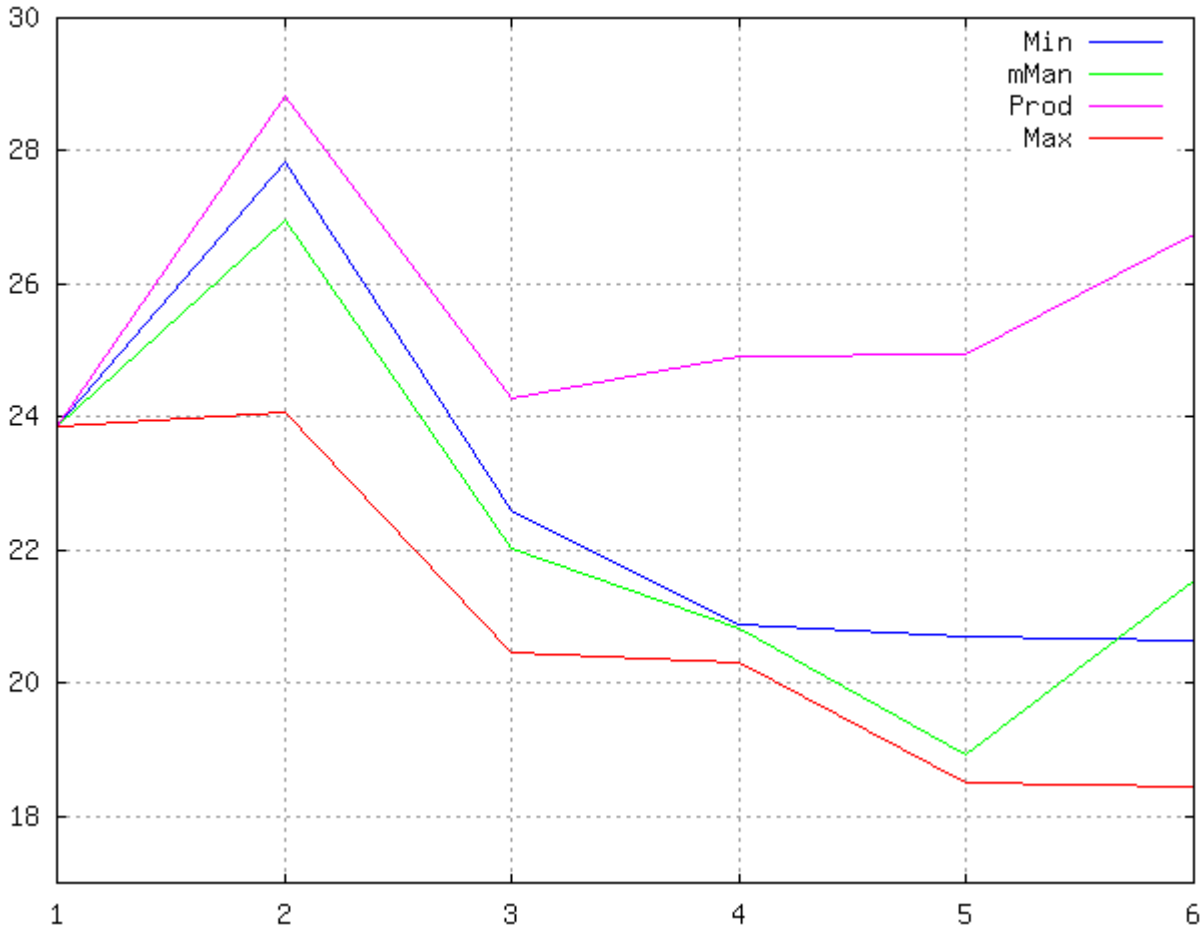
•LABRI_faces_(128-128) (octave)

•LIF_GABOR_8x6_(32-32) (octave)

•LIF_GABOR_8x6 (libsvm)

•LIF_GABOR_4x3 (linear SVM)

•LSIS_PEF150 (libsvm)



En abscisse le nombre de modèles fusionnés et en ordonnée l'avérage précision en %

7.4 Commentaires sur la fusion

Le plus gros gain est pour le topic « street », qui est certainement le topic le plus variable et qui tire donc bénéfice de plusieurs modèles de visions différentes.

La fusion des meilleurs modèles permet un gain en performance important (entre 4 et 10 points) mais l'on peut remarquer que c'est la fusion des deux meilleurs modèles qui est systématiquement la meilleure dans notre étude. Rajouter encore des modèles d'un niveau de performance plus faible ne fait que dégrader les résultats. Ceci est assez logique puisqu'on a généralement uniquement deux ou trois modèles réellement performants.

On constate également que deux méthodes de fusion se détachent particulièrement. Le produit donne les meilleurs résultats suivi du minimum. La moyenne est un cran en dessous tandis que le maximum ne fonctionne vraiment pas et est donc à éviter.

8 Annexes

8.1 Annexe_1_Pré traitement des Données TRECVID

•Conversion Bin vers Ascii :

Les features ont été fournis au format Bin, un fichier étant un vecteur ligne où 4 octets représente un flottant en little endian.

En l'état les features sont donc inutilisables! Nous avons du programmer un Bin_Reader capable de lire ces fichiers binaires et de générer une matrice de flottants formatée correctement. A l'ouverture du fichier Bin, le programme récupère la taille du fichier ainsi que le nombre de dimensions du feature (fournie par l'utilisateur) puis lit chaque flottant un par un, le stocke dans une variable qui est ensuite recopiée dans le fichier destination. Un saut de ligne est inséré chaque fois que « n » flottant, où « n » est le nombre de dimensions du feature, ont été recopiés. Nous avons appris plus tard qu'une commande linux existit pour obtenir le même résultat. Ce qui a permis de valider le script.

•Etiquettage

Nous avons maintenant la possibilité de créer une matrice de flottant à partir du .Bin, mais nous ne disposons pas d'étiquettes pour chaque vecteur de la matrice donc impossible de tester quoi que ce soit avec LibSVM !

Pour étiqueter les vecteurs nous disposons d'un fichier Keylist, contenant entre autres, le nom du fichier image en respectant l'ordre de notre matrice (vecteur 1 = image 1 de la keylist). Nous avons aussi pour chaque topic un fichier d'annotations où pour chaque image il est indiqué si l'image répond au topic du fichier. Plusieurs étudiants ont écrit un script permettant d'obtenir des fichiers d'étiquetage binaire (deux étiquettes : 0 ou 1 selon que l'image appartienne ou non au concept), un par concept. Je me suis inspiré de leur travail pour écrire ma propre version du script, que j'estime plus simple : « etiquettage.sh » Elle consiste à parcourir ligne par ligne un fichier Keylist donné en argument et pour chaque ligne du fichier à récupérer le nom de l'image minus l'extension. Il suffit ensuite de récupérer dans tous les fichiers annotations l'étiquette attribuée àcette image. Une fois le script exécuté (très long !!!) nous disposons de 20 fichiers par base, fichiers contenant chacun un vecteur colonne. Il reste à compléter les matrices de flottants construites par le binreader pour obtenir des données exploitables.

•Passage au format Torch

Comme mentionné précedemment nous nous sommes fixés comme standard le format de données utilisé par Torch Vision, j'ai donc écrit un script Bash capable de générer 20 matrices (une par topic) par feature à partir des fichiers étiquettes et des matrices de flottants. Ce script s'appelle « mat2torch.sh » Les fichiers étiquettes ont le même nombre de lignes et possèdent le même ordre que les matrices, il suffit donc d'ajouter une colonne étiquette à chaque ligne de la matrice puis d'écrire sur la première ligne du nouveau fichier le nombre de lignes et de colonnes. Ce traitement est effectué pour tout les fichiers .mat construits par le binreader. Le fichier généré se nomme »NomduFeature_NomduTopic.TORCH », par exemple: « CEALIST_global_pigment_Airplane_flying.TORCH ». Les étudiant travaillant sous Torch disposent après cette étape de fichiers exploitables, pour les autres il suffit de faire appel au convertisseur construit en début de projet (dans mon cas « Torch2libsvm »).

8.2 Adaptation / autres machines

8.2.1 Lssvm

torch2lssvm.c : Script convertissant les fichiers TORCH en fichier de format d'entrée LSSVM.

(division du fichier TORCH en deux fichiers:données et etiquettes).

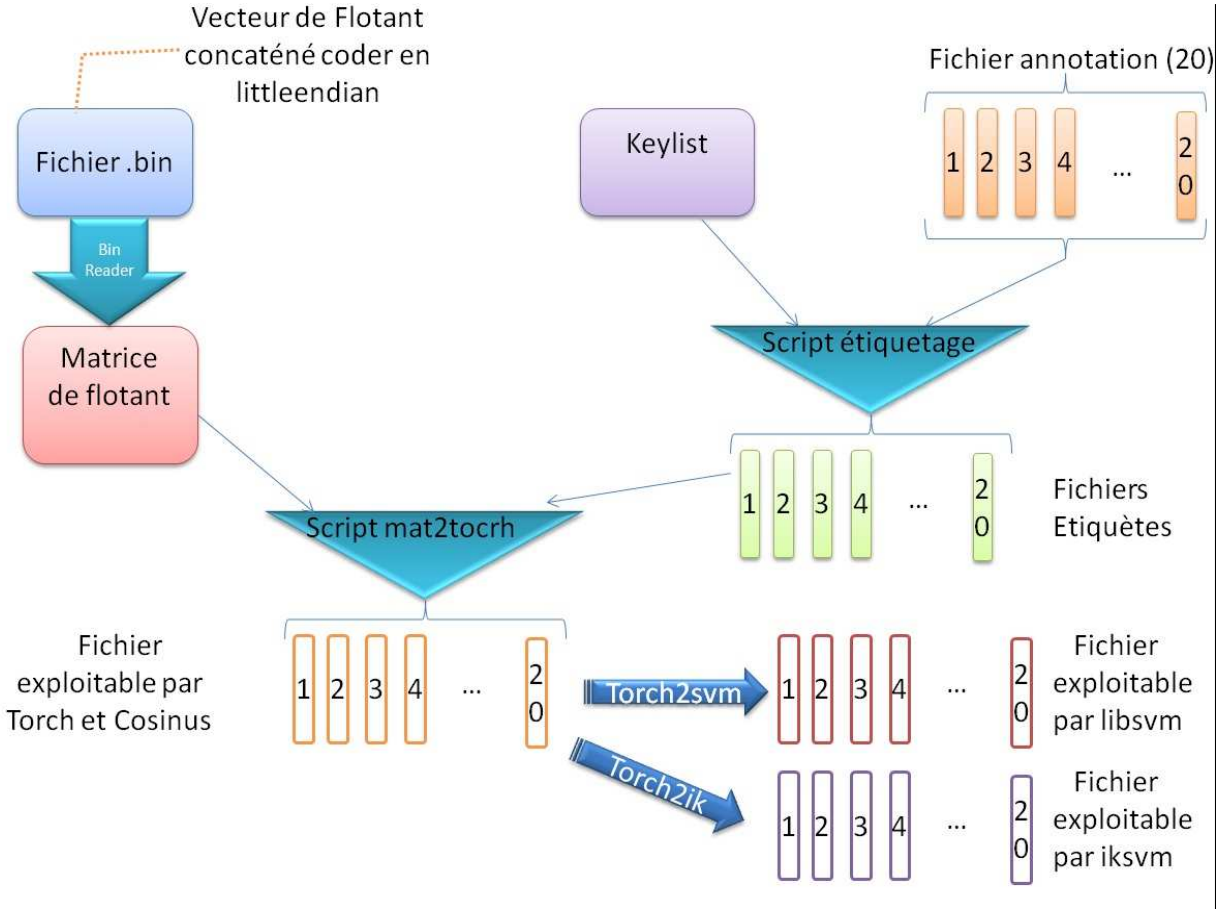
ligne de comande :

```
./torch2lssvm.x fichier_torch fichier_donnee fichier_etiquette
```

8.2.2 Iksvm

8.3 Schéma récapitulatif

Pour y voir plus clair voici un schéma revenant sur toutes les étapes nécessaires pour obtenir 20 matrices exploitables avec LibSVM à partir d'un fichier binaire et des fichiers annotations. Script convertissant les fichiers TORCH



8.3.1 Familiarisation avec LibSVM

La première chose à faire a été de se familiariser avec LibSVM. Pour ce faire un Test basé sur des matrices au format TORCH, format qui a été défini comme base de travail de par sa simplicité, a été effectué. TORCH utilise des fichiers débutant par une ligne où figure le nombre de lignes et le nombre colonnes. Puis chaque ligne contient un vecteur de features, se terminant par une composante entière correspondant au numéro de l'étiquette. LibSVM attend un fichier où chaque ligne débute par l'étiquette de l'image, puis chaque composante non nulle précédée de son numéro suivi de « : ».

Ex : 1.000000 1:1.462360 2:0.645135 3:0.835925 4:1.470420 5:0.919971

Il faut donc écrire une routine permettant Script convertissant les fichiers TORCH la conversion de données au format TORCH en données de format LibSVM.

Le concept du programme est simple :

•On donne en arguments le nom du fichier TORCH source ainsi que le fichier destination souhaité.

•On ouvre le fichier source en lecture et on récupère le nombre de lignes et de colonnes de la matrice (première ligne du fichier TORCH)

•Puis on récupère les données ligne par ligne, données que l'on recopie dans le fichier destination avec le bon formatage.

Le programme se nomme « torch2libsvm.c »

Grâce à cet exécutable nous pouvons donc disposer de données au bon format. Mais les données TORCH comportaient des composantes allant de 0 à 255, valeurs qui posaient visiblement problème à libsvm qui ne pouvait prédire qu'une seule classe et donc faire énormémentd'erreurs...

Pour faire face à ce problème j'ai utilisé l'exe svm scale présent avec libsvm :

•./svmscale l 0 u 1 s scale_parameters fichier_train

Cette commande normalise les données entre 0 et 1 pour le fichier de training.

•./svmscale r scale_parameters fichier_test

Celleci applique la même normalisation aux données d'entraînement. A partir de là j'ai pu lancer un entraînement, en gardant les paramètres par défaut, à l'aide de l'exécutable « svmtrain » :

•./svmtrain fichier_train

A l'issue de cet entraînement libsvm a généré un modèle, prêt à être exploité sur les données de Test pour une prédiction :

•./svmpredict fichier_test fichier.model result

Dans « result » se trouve la liste des prédictions effectuées.

8.4 SCRIPTS ET PROGRAMMES UTILISES

Readme:

```
 ------------------------------------------------------------
May 19, 2008
Subhransu Maji (smaji@cs.berkeley.edu)
------------------------------------------------------------
This code is an addon to LIBSVM. Follow the instructions in README to compile the code.

In addition to the executables in libSVM, the Makefile is modified
to compile fastclassify.mex* which is a fast SVM classifier for intersection kernel.

test.m : contains code to test the fast classifier and benchmark test times and errors
         compared to the standard implementation.

svm.cpp/svm.h have been modified to support intersection kernel (-t 4).

Note instersection kernel can be used when features are histograms
(i.e. L1 normalized and positive).

TO USE first train the model using -t 4 option and C-SVC (-s 0)
model=svmtrain(training_label_vector, training_instance_matrix, '-t 4 -s 0 ...');

[ model ] = svmtrain('../LARAKI/SCRIPT/test_label.txt','../LARAKI/SCRIPT/test_IRIT_BoW-
SIFT_Two_people.iksvm', '-t 4 -s 0 -b 1')


use fastpredict to classify the examples

>>fastpredict

 Usage: [exact_values, pwconst_values, pwlinear_values,[times]] = ...
        fastpredict(testing_label_vector, testing_instance_matrix, model,'libsvm_options')

svmpredict('../LARAKI/SCRIPT/test_label.txt','../LARAKI/SCRIPT/test_IRIT_BoW-
SIFT_Two_people.iksvm','model','-b 1 -v 1')

 Output:
   exact_values    : predictions using binary search
   pwconst_values  : approximation using piecewise constant function
   pwlinear_values : approximation using piecewise linear function
   [times]         : running times


 libsvm_options:
   -b probability_estimates: whether to predict probability estimates, 0 or 1 (default 0);
   -v verbose flag         : 0 or 1 (default 0);
   -n number of bins       : [2,...] (default 100);

 (Note: Only SVC and 2 class classifier is supported)
```

*8.4.1 Génération des fichiers étiquettes « **etiquettage.sh** »*

```
#!/bin/bash
#Script servant à générer les 20 vecteurs colonnes correspondant aux étiquettes binaires des 20
topics
#Le vecteur généré est dans le même ordre que le fichier keylist ainsi que les matrices
#Il sera assez simple d'intégrer les étiquettes aux matrices pour générer des données Torch
#USAGE   :   ./etiquettage.sh   DOSSIER_CONTENANT_FICHIERS_ANNOTATIONS NOM_FICHIER_KEYLIST
#les 20 fichiers générés ont pour nom : NOMCONCEPT_etiquettes
#Ecrit par Guillaume Foot le 27 octobre 2009 à partir du travail d'Aurelien Couvert et Thibault
Dailly
oldIFS=$IFS     # sauvegarde du séparateur de champ
IFS=$'\n'     # nouveau séparateur de champ, le caractère fin de ligne
#on se place dans le repertoire des fichiers annotations
cd $1
#on récupère le nom du fichier Keylist
keylist=$2
#pour chaque ligne du fichier
for line in $(cat $keylist);
        do
                #on recupère le nom du fichier image minus l'extension
                nom_fichier=(`echo "$line" | cut d ' ' f 3`)
                nom_fichier=${nom_fichier%.*}
                #et pour chaque fichier annotation
                for ann in $(ls | grep .ann);
                        do
                                #on recupère l'étiquette de l'image courante (P ou N) et on
l'insère chiffrée dans le fichier de sortie correspondant
                                cat $ann | grep $nom_fichier' '| cut d ' ' f 6 | tr 'NP' '01' >> $
{ann %.*}_etiquettes
                        done
done
#on remet l'ancien séparateur de champ
IFS=$old_IFS
Conversion en flottants des matrices Binaires
int main(int argc, char *argv[]){
/*nombre d'arguments incorrect*/
if (argc != 4)     {
    printf("syntaxe: %s fichier_bin fichier_destination nb_dimensions\n",argv[0]);
    return(1);
    }
 FILE * source;
 FILE * dest;
 long lSize;
 float tmp;
 size_t result;
 int i, j;

//nombre de lignes
int n = atoi(argv[3]);

//ouverture des fichiers
 source = fopen ( argv[1] , "rb" );
 dest = fopen(argv[2],"w");
  // Recuperation de la taille du fichier
  fseek (source , 0 , SEEK_END);
  lSize = ftell (source);
  printf("%ld\n", lSize);
  rewind (source);
  //on recupère chaque flottant du fichier
  for(i=0; i < lSize/n; i=i+sizeof(float)){
            for (j = 0;j<n;j++){
               result = fread(&tmp, sizeof(float), 1, source);
                   //on le copie dans le fichier destination
                   fprintf(dest, "%f ", tmp);
                   }
            //saut de ligne dans le fichier destination
            fprintf(dest,"\n");
         }

  fclose (source);
 return 0;
}
```

*8.4.2 .Construction des fichiers TORCH « **mat2torch.sh** »*

```
#!/bin/bash
#script qui génére des fichiers format TORCH à partir des matrices de flottants et des fichiers
étiquettes
#un fichier par topic
#les   matrices   doivent  IMPERATIVEMENT  être  en  .mat,  les   fichiers   étiquettes  se
terminer   par
_etiquettes
#USAGE : ./mat2torch.sh dossier_contenant_matrices dossier_contenant_fichier_etiquette
#Ecrit par Guillaume Foot le 28 octobre 2009
oldIFS=$IFS     # sauvegarde du séparateur de champ
IFS=$'\n'     # nouveau séparateur de champ, le caractère fin de ligne
cp $2*_etiquettes $1
rep_courant=(`pwd`)
#pour chaque fichier matrice
for fic in $(ls $1 | grep .mat);
        do
                #pour chaque fichier étiquette
                for fic_etiquette in $(ls $2 | grep _etiquettes);
                        do
                                cd $1
                                #on ajoute la colonne étiquette au fichier matrice
                                #on stocke le tout dans un fichier temporaire
                                paste d ' ' $fic $fic_etiquette > temp
                                #suppression des lignes SKIPS
                                cat temp | grep v S$ > temp2
                                #on recupère le nombre de lignes et de colonnes du fichier
                                nblignes=(`cat temp2 | wc l | cut d ' ' f 1`)
                                nbmots=(`cat temp2 | wc w | cut d ' ' f 1`)
                                nbcolonnes=$(($nbmots/$nblignes))
                                echo $fic
                                #on   inscrit   ces   deux   infos   sur   la   première   ligne
du   futur   fichier
.TORCH
                                echo   "$nblignes   $nbcolonnes"   >   ${fic%.*}_${fic_etiquette
%_etiquettes}.TORCH
                                #on recopie la matrice étiquettée dans le fichier final
                                cat temp2 >> ${fic%.*}_${fic_etiquette%_etiquettes}.TORCH
                                cd $rep_courant

                done
done
rm $1*_etiquettes
rm $1'temp'
rm $1'temp2'
Conversion d'un fichier TORCH en Libsvm
include<stdio.h>
#include<stdlib.h>
/*
 * argv 0 nom de l'exe
 * argv 1 fichier source
 * argv 2 ficher destination
 */
int main( int argc, char *argv[] ) {
/*nombre d'arguments incorrect*/
if (argc != 3)
    {
    printf("syntaxe: %s fichier_torche fichier_libsvm\n",argv[0]);
    return(1);
    }
int i,j;
int l,c;//ligne ,colonne
//les fichiers sources et destination
FILE * source;
FILE * dest;
//ouverture des fichiers
source = fopen(argv[1],"r");
dest = fopen(argv[2],"w");
//nombre de lignes et de colonnes
fscanf(source, "%d %d\n", &l, &c);
printf("%d %d\n", l, c);
//vecteur de c dimensions, destiné à recevoir une ligne de la matrice de données
float vect[c];
//pour chaque ligne de la matrice
```

```
for (i = 0; i < l; i++) {
          //on récupère le vecteur de données et on le stocke dans notre tableau
          for (j = 0;j<c;j++){
                   fscanf(source,"%f",&vect[j]);
                   }
          /*puis on recopie la ligne au format libsvm
        c'est à dire avec l'étiquette en premier puis les composantes numérotées*/
        fprintf(dest,"%f ",vect[c1]);
        for (j = 0;j<c1;j++){
                 //on prend soin de ne pas recopier les valeurs nulles
                 if(vect[j]!=0)
                         fprintf(dest,"%d:%f ",j+1,vect[j]);
        }
        //saut de ligne dans le fichier destination
        fprintf(dest,"\n");
        }
fclose(source);
fclose(dest);
 return 0;
}
```

8.4.3 Conversion et normalisation des données TORCH vers LIBSM ***« torch2libsvm_scaled.sh »***

```
#!/bin/bash
#pour un feature donné en deuxième paramètre du script, les 20 matrices libsvm sont construites à à
l'aide d'une liste des topics fournie en premier paramètre
#USAGE : ./torch2libsvm_scaled.sh liste_topics liste_features dossier_contenant_TORCH
current_directory=(`pwd`);
#pour chaque feature
for feature in $(cat $2);
        do
#pour chaque topic
        for topic in $(cat $1);
                do
                        #on se déplace dans le repertoire des données TORCH
                        cd $3
                        #on convertit la matrice Torch correspondante au format libsvm
                        ./torch2libsvm            ${feature}_$topic.TORCH          ${feature}_$
{topic
%.TORCH}.libsvmtemp
                        #on normalise les données de cette nouvelle matrice pour être en 0 et 1
                        #on range le résultat dans une nouvelle matrice d'extension.libsvm
                        ./svmscale   l   0   u   1   ${feature}_${topic%.TORCH}.libsvmtemp   >   $
{feature}_${topic%.TORCH}.libsvm
                        #on supprime la matrice non normalisée pour éviter d'exploser les disques
durs
                        rm ${feature}_${topic%.TORCH}.libsvmtemp
                        cd $current_directory
                done
        done
```

8.4.4 Script convertissant les fichiers TORCH > Lssvm ***« torch2lssvm.c »***

#include <stdio.h>

```
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
 FILE *rawData = NULL;
 FILE *X = NULL; // fichier d'entree avec le format torch
 // fichier de sortie contenant les donnees
 FILE *Y = NULL;  // fichier de sortie contenant les etiquettes
 int nbRow = 0; // nb total de ligne
 int nbCol = 0; // nb total de colonne
 int curCol = 0; // current column
 int curRow = 0; // current row
 int val = 0; // valeur lu dans le fichier
 // recuperation des arguments de la ligne de commande
 if (argc < 4) {
    printf("Usage : %s fichier_torch fichier_donnee fichier_etiquette\n\n", argv[0]);
    return 1;
 }
 rawData = fopen(argv[1], "r"); // ouverture des fichier
 if (rawData == NULL) {
    perror(argv[1]);
    return 2;
 }
 X = fopen(argv[2], "w+");
 if (X == NULL) {
    perror("X");
    return 2;
 }
 Y = fopen(argv[3], "w+");
 if (Y == NULL) { perror("Y"); return 2; }
 fscanf(rawData, "%d %d\n", &nbRow, &nbCol);  // recuperation du nombre de ligne et colonne
 printf("Taille : ligne = %d\tcolonne = %d\n", nbRow, nbCol);
 // creation des fichiers de sortie
 for (curRow = 1; curRow <= nbRow; curRow++) {
    for (curCol = 0; curCol < nbCol; curCol++) {
    fscanf(rawData, "%d", &val);
    if (curCol == nbCol - 1) {
        // c'est l'etiquette  21
        fprintf(Y, "%d\n", val);

        fprintf(X, "\n");
    }
    else
        fprintf(X, "%d ", val);
    }
```

8.4.5 Script convertissant les fichiers TORCH > iksvm « ***Torch2iksvm*** *»*

```
% This function is made to convert TORCH files to IKSVM files
% It also generate two files : 1 - A Matrix called M
%                              2 - A label vector called label
% Whatever you can change the name of those two files if you disire
% NB IKSVM is based on matlab and libsvm so watch out your directories before running it.
% Before running this function be sure that the first line of torch files was deleted.
function [ M label ] = torch2iksvm(torchdatasansentete)
% made by L.Omar
T = load(torchdatasansentete);
label = T(:,end)';
save -ascii label.txt label
M = T(:,1:end-1);
m = min(M');
M = M - (m' * ones( 1 , size(M,2)));
M = M ./ (sum(M')' * ones( 1, size(M,2)));
save -ascii iksvm.txt M
endfunction
```

8.4.6 Script éffectuant les trains et les tests

```
#!/bin/bash
#USAGE : ./script.sh liste_topics listefeatures
current_directory=(`pwd`);
#pour chaque feature
for feature in $(cat $2);
        do
#pour chaque topic
        for topic in $(cat $1);
                do
                        ../../libsvm/svmtrain   b   1   h   0
../../DATA/train/2007/EXTENSION_MAT/$ {feature}_${topic}.libsvm
                        ../../libsvm/svmpredict   b   1   ../../DATA/test/2007/EXTENSION_MAT/$
{feature}_${topic}.libsvm ${feature}_${topic}.libsvm.model ${feature}_${topic}.result
                done
        done
```

8.4.7 Script Bash : formatage de OutPutMeasurer > Octave « outputer.sh »

```
#!/bin/bash

liste=( LIP6_text LSIS_DF IRIT_MFCC-Var-Min LABRI_varianceResidualMovement_Mean LIF_HISTCOL_2x2
GIPSA_faces_9 LEAR_bow_sift_1000  LIF_GABOR_2x2  CEALIST_global_cciv CEALIST_local LSIS_PEF150
GIPSA_audio_intensity )

liste=( Hand Two_people Street )
nhu=(128 256)
itr=(64 128 256)

for l in ${itr[@]}; do
for i in  ${nhu[@]}; do
       for j in  ${topic[@]}; do
              for k in ${liste[@]}; do
                     output=$k"_"$j"_("$i"-"$l").outputMeasurer"
                     octave=$k"_"$j"_("$i"-"$l").octave"
                     echo "TRANSFORMATION OutputMeasurer"
                     echo $output
                     echo ""
                     cat $output | grep -v "^1 2$" | cut -d ' ' -f 2 | sed -e 's/e\-/\*10\^\-/g' |
sed -e 's/\(.*\)/e(\1)/g' | bc -l > $octave
              done
       done
done
done
```

8.4.8 Fonction Octave : construction du rappel/précision **« rappel_precisionV4.m»**

```
function [ AveragePrecision ] = rappel_precision(  output_file   ,  truth_file )
% Foot Glotin nov09
% V0.0.0.4
% pour output LIB SVM, à modifier pour torch ...
% Truth_file = fichier binaire, 1 pour élément pertinent
T = load(truth_file);
out = load(output_file);
% pour libsvm
p_out = out(:,3);
% pour torch, si classe 1 = la bonne
% p = exp(out(:,2));
[ v original_lig ] = sort(p_out ,'descend');
card_pertinent_recalled = cumsum(T(original_lig));
precision = card_pertinent_recalled ./ [1: length(T)]'; % / number of returned by the system
recall = card_pertinent_recalled / sum(T) ; %  / card(pertinent in the test set)
plot(recall, precision)

ylabel('Precision');
xlabel('Rappel');
Average_Precision_min = sum( diff(recall) .* precision(1:end1) );
Average_Precision_sup = sum( diff(recall) .* precision(2:end) );
output_file
disp(['Tested on ' num2str(length(T)) ' images'])
AveragePrecision = round(10000*mean( [ Average_Precision_min , Average_Precision_sup ] )) /
100;
score = num2str(AveragePrecision);
p10 = num2str(precision(10)*100);
p100 = num2str(precision(100)*100);
disp(['Average Precision (in %)= ' score ])
%partie   optionelle,   je   choisis   de   supprimer   le   .result   de   mon   outputfile   pour
l'affichage   et   les
fichiers générés
output_file = output_file(1:end  7);
eval(['save ascii AverPrecinPerCent_of_' output_file(1:end  7) '.txt AveragePrecision '])
grid("on")
title(["Courbe Rappel Precision de ", output_file,"\n\nAverage Precision = ", score, "%\nPrecision
sur les 10 premiers documents retournes : ", p10, "%\nPrecision sur les 100 premiers documents
retournes : ", p100, "%"])
%remplacer ImagesCourbes/ par le dossier souhaité où sauvegarder les graphiques
print('dpng', ["ImagesCourbes/recall_precision_of_", output_file, ".png"])
endfunction
```

8.4.9 Fonction Octave : Fusion des modèles basée sur la moyenne « **fusion_mean.m** »

MIN ET MAX SONT PARFAITEMENT ANALOGUES

```
function [] = fusion_mean(  output_tab   ,  truth_file )
% Foot nov09 / V1.0
% pour output LIB SVM, à modifier pour torch ...
% Truth_file = fichier binaire, 1 pour element pertinent
%output_tab est un tableau de chaines de caractères : les noms des fichiers résultats à fusionner
%chargement des étiquettes
T = load(truth_file);
%chargement des fichiers résultats
for i = 1:length(output_tab);file_name=["Tab",
num2str(i)];load_str=[file_name,"=load(output_tab{1,i});"];eval(load_str);str=[file_name,   "=",
file_name,"(:,3);"]; eval(str);end
output= []
%output = concaténation de tout les fichiers résultats
for i=1 : length(output_tab);str=["output=[output, Tab",num2str(i),"];"];eval(str);end;
%output = vecteur colonne contenant la moyenne des composantes de tout les fichiers à fusionner
output=mean(output,2);
[ v original_lig ] = sort(output ,'descend');
card_pertinent_recalled = cumsum(T(original_lig));
precision = card_pertinent_recalled ./ [1: length(T)]'; % / number of returned by the system
recall = card_pertinent_recalled / sum(T) ; %  / card(pertinent in the test set)
plot(recall, precision)
ylabel('Precision');
xlabel('Rappel');
Average_Precision_min = sum( diff(recall) .* precision(1:end1) );
Average_Precision_sup = sum( diff(recall) .* precision(2:end) );
disp(['Tested on ' num2str(length(T)) ' images'])
AveragePrecision = round(10000*mean( [ Average_Precision_min,Average_Precision_sup ] ))/100;
score = num2str(AveragePrecision);
p10 = num2str(precision(10)*100)
p20 = num2str(precision(20)*100)
p100 = num2str(pre10cision(100)*100)
disp(['Average Precision (in %)= ' score ])
%partie optionelle, je choisis de supprimer le .result de mon outputfile pour l'affichage et les
fichiers générés
grid("on")
title(["Average Precision = ", score, "%\nPrecision sur les 10 premiers documents retournes : ",
p10,
"%\nPrecision sur les 20 premiers documents retournes : ", p20,"%\nPrecision sur les 100 premiers
documents retournes : ", p100, "%"])
print('dpng', ["test.png"])
endfunction
```