République Algérienne Démocratique Et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahleb Blida 1

Faculté des sciences

Mémoire

Présentée pour l'obtention du diplôme de Master

En : INFORMATIQUE

Spécialité **: Traitement Automatique des Langues**

Sujet

Enrichissement de requêtes en utilisant le word2vec pour la recherche d'images

Devant le jury composé de :

Présidente : M^{me}.Bacha

Examinateur : M^{r}.Benaissi

Promotrice : M^{me}.Benblidia Nadjia

Co-Promotrice: M^{me} .Bakalem Mahdia

Présenté par :

M^{r} .Bettaoula Kouider

M^{r} .Bentaleb Mohamed

Année universitaire 2018/2019

Résumé

Les images numériques sont une source d'information très expressive qui joue un rôle très important dans de nombreuses activités humaines. Par conséquent, la recherche d'images est devenue un sujet d'intérêt pour de nombreux chercheurs.

La recherche d'images dépend de la formulation de la requête pour répondre à un besoin très bien précis. L'enrichissement de la requête jeux un rôle très important dans l'amélioration de la recherche d'image. Notre travail consiste à utiliser le modèle word2vec pour l'enrichissement de la requête dans la recherche d'image.

Dans notre travail, nous entrons une requête simple contenant des mots dans le modèle Word2Vec afin d'extraire des mots similaires dans le contexte et la signification de chaque mot de la requête initiale, ce qui signifie que la requête est enrichie.

Avec cette requête enrichie, nous pouvons obtenir des résultats plus pertinents et plus étendus lors de la recherche d images.

Mots-clés: recherche d'image, enrichissement de requête, Word2Vec.

Abstract

Digital images are a very expressive source of information that plays a very important role in many human activities. As a result, image search has become a topic of interest for many researchers.

The search for images depends on the formulation of the request to answer a very specific need. Enriching the game query plays a very important role in improving image search. Our job is to use the word2vec template to enrich the query in image search.

In our work, we enter a simple query containing words in the Word2Vec template to retrieve similar words in the context and meaning of each word in the original query, which means that the query is enriched.

With this enriched query, we can get more relevant and extended results when searching for images.

Keywords: image search, query enrichment, Word2Vec.

ملخص

تعد الصور الرقمية مصدرًا معبرًا للمعلومات و التي تلعب دورًا مهمًا للغاية في العديد من الأنشطة البشرية. نتيجة لذلك أصبح البحث عن الصور موضوع اهتمام لكثير من الباحثين.

يعتمد البحث عن الصور على إنشاء استعلاما للإجابة عن حاجة محددة للغاية ،لذلك اقترحنا قالبًا لتطبيق خوارزمية Word2Vec التي تعمل على إثراء الاستعلامات لتحسين نتيجة البحث عن الصور.

في عملنا، نقوم بإدخال استعلام بسيط يحتوي على كلمات في قالب Word2Vec لاسترداد كلمات متشابهة في سياق ومعنى كل كلمة في الاستعلام الأصلي، مما يعني أنه تم إثراء الاستعلام.

باستخدام هذا الاستعلام الثري، يمكننا الحصول على المزيد من النتائج ذات الصلة والموسعة عند البحث عن الصور.

الكلمات الرئيسية: البحث عن الصور، إثراء الاستعلام، Word2Vec.

Remerciements

Nous remercions d'abord le bon Dieu qui nous a aidés et qui nous a donné le courage et la patience pour réaliser ce modeste travail.

Notre profonde gratitude et sincère remerciement vont à nos promoteurs Benblidia Nadjia et Bakalem Mahdia de nous avoir encadré, pour son suivi et son orientation.

Nous leur devons également, leur profonde reconnaissance pour le temps précieux qu'ils nous ont consacré, leurs remarques pertinentes et précieuses, leurs conseils et leurs encouragements.

Nos remerciements vont aussi aux membres de jury pour avoir accepté de juger ce travail.

Nous tenons à remercier nos enseignants qui nous ont transmis le savoir et le bien.

Nous tenons à remercier également tous ceux qui nous ont aidé et contribué de près ou de loin à la réalisation de notre projet.

Dédicaces

Merci mon dieu de m'avoir donné la capacité d'écrire, de réfléchir et la force

D'y croire, Je dédie ce modeste travail à :

Notre promoteurs madame Benblidia Nadjia et madame Bakalem Mahdia qui nous a donné tout ce qu'il peut, nous avons appris beaucoup de chose avec eux, nous les remercions pour ces efforts.

Mes chers parents qui représentent mes deux yeux.

Ma petite seur INTISSAR.

Toute ma famille paternelle et maternelle.

Mon Binôme kouider : il était sérieux tout la période de la réalisation du projet, je te remercie pour tes efforts.

Tous mes enseignants en université saad dahleb blida1 Et tous mes amis(es) et surtout promo Master 2018/2019

BENTALEB MOHAMED

Dédicaces

Je dédie ce travail à ceux qui me portent

Dans leurs cœurs et m'accompagnent dans leurs pensées.

A ceux dont personne ne peut remplacer les sacrifices

Qu'ils ont consentis pour mon éducation et

Mon bien être

Mes très chers parents

A mes très cheres sœurs et leurs maries et leurs bébés et mon très cher

Frère Ahmed, et toute ma famille sans exception

A mon binôme Mohamed qui a fait un excellent travail

Et aux membres de notre promotion

A tous mes ami(e)s sans exception et toute personne qui de près ou de loin nous a aidé et a participé à réaliser notre projet

Et qui nous ont encouragé tout au long de notre

Formation

BETTAOULA KOUIDER

Table de Matières

Listes des figures

Liste des tableaux

Introduction générale

I. Introduction générale

Avec le développement de l'Internet, et la disponibilité de capture d'image des dispositifs tels que caméras numériques, l'image scanners, la taille de la base d'images numériques est en augmentation très rapide dans des domaines variés : la médecine, l'archives (patrimoine culturel, musées, . . .), l'agences photographiques, l'éducation, l'image satellites et aériennes,...etc.

Pour utiliser efficacement ces bases d'images de manière automatique, un système recherche d'images est nécessaire. C'est pourquoi le sujet de la recherche d`images devient un sujet très actif dans la communauté internationale depuis plus d'une dizaine d'années. La recherche d'images consiste à établir une correspondance entre l'image disponible et celle recherchée par l'utilisateur.

A cet effet, pour que la recherche d'images soit opérable pour communiquer des résultats fidèles aux attentes des utilisateurs, l'introduction de la requête est indispensable.

En fonction de la nature de notre projet, nous avons organisé notre rapport en 4 chapitres comme suit:

Le premier chapitre permet d'étudier le domaine de la recherche d'images en présentant l'image et ces caractéristiques, puis passant au concept de recherche d'images, de l'architecture jusqu'aux différentes catégories de recherche d'images CBIR et TBIR.

Le deuxième chapitre traite les techniques de représentation d'un mot, en focalisant sur le Word embedding et leurs modèles, plus précisément le modèle de word2vec.

Le troisième chapitre décrire notre approche proposée, qui consiste à enrichir la requête en utilisant word2vec, en détaillant toute la démarche.

Le quatrième chapitre présent l'implémentation et l'évaluation de notre approche proposée.

Chapitre I : *Recherche d'images*

1. Introduction

Une image est une représentation matérielle d'un objet ou d'un être, ou une représentation mentale de ce qui a déjà été perçu.

L'image est par essence un véhicule privilégié de l'intensité. Ne dit-on pas qu'une image vaut mille mots ? Notre cerveau est plus apte à traiter des données visuelles qu'un long discours, d'autant que ces données sont porteuses de sens en elles-mêmes. Couleurs et formes ont un impact sur le message perçu.

Le domaine de recherche d'images est un domaine en plein essor et très actif de nos jours. L'objectif principal de ce domaine de recherche est de développer des systèmes permettant de libérer l'utilisateur d'une recherche d'images manuelle laborieuse et fastidieuse.

Ainsi, un utilisateur est demandé d'exprimer ses besoins sous forme de requêtes visuelles ou textuelles, et c'est le système de recherche qui prend la charge de localiser un maximum d'images pertinentes à ces requêtes en un temps raisonnable.

Nous présentons dans ce chapitre l'image et leurs caractéristiques, Ensuite nous passons au concept de la recherche d'image. En présentant l'architecture de cette recherche et les différentes catégories de recherche d'image: la recherche d'images par contenu (CBIR) et la recherche d'images par le texte (TBIR).

2. Définition d'image

La définition du terme « image » lui-même, telle qu'elle est donnée par le Petit Robert, englobe une multitude de significations distinctes. Cela va de la« reproduction exacte ou représentation analogique d'un être, d'une chose », à la « représentation mentale d'origine sensible » ou à des concepts plus physiques comme un « ensemble des points » où vont converger des rayons lumineux (cas des images optiques). **[1]**

L'image numérique est l'image dont la surface est divisée en éléments de tailles fixes appelés cellules ou pixels, ayant chacun comme caractéristique un niveau de gris ou de couleurs prélevé à l'emplacement correspondant dans l'image réelle. **[Gruber,1993]**

3. Caractéristiques d'image

Caractéristiques de bas niveau ce sont les caractéristiques qui décrivent le contenu visuel de l'image. Il existe plusieurs caractéristiques visuelles, nous citons quelques caractéristiques principales :

- ***LA COULEUR :***

La couleur est l'un des plus reconnaissables éléments du contenu visuel d'une image, c'est la plus utilisé dans la recherche image. **[Christophe,2008]**

Il existe plusieurs distributeurs de couleur tel que : histogramme, les moments couleur…etc.

- ***LA TEXTURE :***

Une définition formelle de la texture est quasiment impossible. Mais d'une manière générale la texture se traduit par un arrangement spatial des pixels que l'intensité ou les couleurs seules ne suffisent pas à décrire. **[Christophe,2008]**

- ***LA FORME :***

La forme est utilisée pour caractériser les objets dans les images. On distingue deux catégories de descripteurs de formes : les descripteurs basés régions et les descripteurs basés frontières. Les premiers sont utilisés pour caractériser l'intégralité de la forme d'une région, Les seconds portent sur la caractérisation des contours de la forme. **[Christophe,2008]**

4. Définition de la recherche d'image

La recherche d'images est un domaine informatique pour la navigation, recherche et extraction des images d'une grande base d'images numériques, c'est une recherche de données spécialisées utilisées pour trouver des images. **[Ben Cheikh,2011]**

5. Architecteur générale de recherche d'images

L'architecture générale de la recherche d'image englobe deux phases, la première offline, appelée aussi indexation et la deuxième online, nommée aussi recherche.

La figure suivante présente l'architecteur générale de la recherche des images

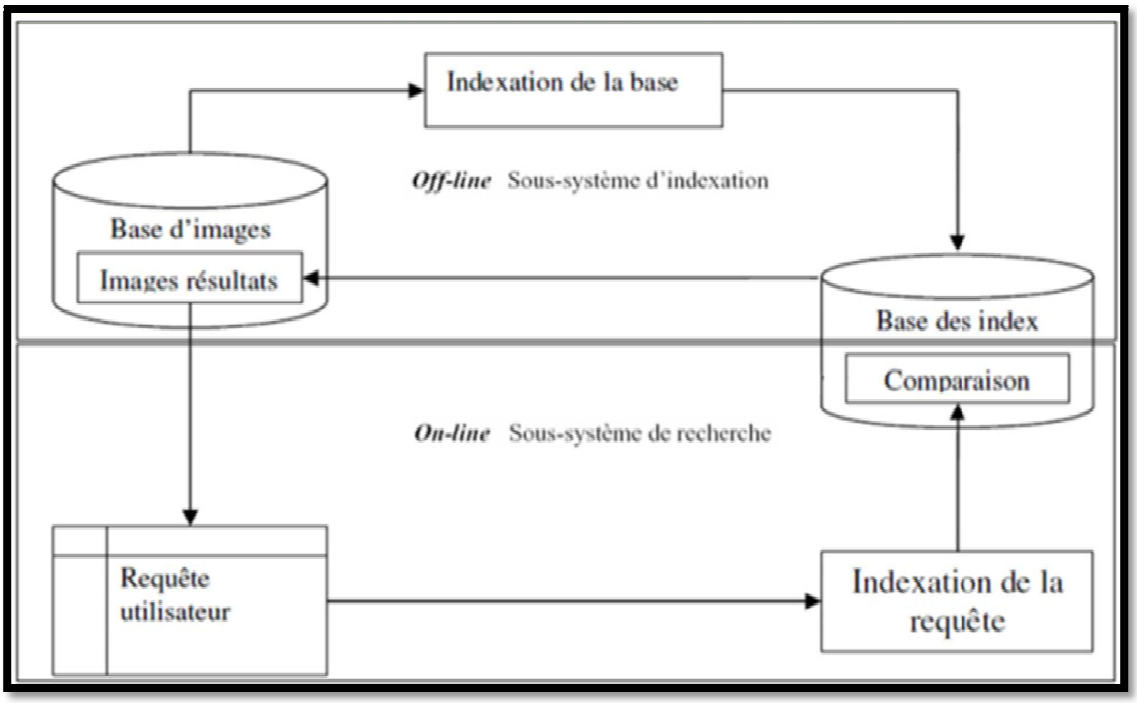


Figure 1.1: Architecture générale de recherche d'images [2]

5.1. **Phase offline (Hors-ligne) Indexation :** L'indexation des documents est une étape très importante dans le processus de la recherche d'image. Le processus d'indexation permet d'analyser chaque image de la collection afin de créer un ensemble d'éléments «clés» qui caractérisent le contenu de l'image. Ces éléments clés sont soit une information textuelle (mots simples, groupe de mots) ou une image (Couleurs, formes…). L'indexation peut être manuelle, semi automatique ou automatique :

- Indexation manuelle

L'indexation manuelle d'image est réalisée par des personnes (experts) qui décrivent le contenu sémantique de chaque image de la collection. Elle présente plusieurs inconvénients. Tout d'abord, elle est sujette à l'interprétation des personnes : pour un même document, des termes différents peuvent être attribués par des personnes différentes. Le coût de ce type d'indexation représente un inconvénient supplémentaire à sa généralisation. De plus, elle est très difficile à réaliser pour des collections volumineuses. **[Radia Abdi,2012]**

- Indexation automatique

L'indexation automatique est un processus complètement automatique, l'intervention d'experts n'étant plus nécessaire dans ce cas. Ce type d'indexation est très pratique pour des collections très volumineuses **[Radia Abdi,2012].**

- Indexation semi-automatique

L'indexation semi-automatique représente la combinaison des deux indexations présentées auparavant. Premièrement, un processus automatique extrait les descripteurs de chaque image de la collection. Ensuite, les experts ont la tâche de choisir quels termes décrivent le mieux les documents. D'habitude, ils se basent sur des vocabulaires contrôlés, tels que les dictionnaires, les thésaurus ou les ontologies **[Radia Abdi,2012]**.

5.2.**Phase online (ligne) Rechercher :** la phase de recherche permet de retrouver des images pertinentes aux utilisateurs, en répondant à leurs requêtes. La première étape de la recherche d'images est la constitution de la requête, qui doit permettre au système de retrouver les images désirées par l'utilisateur. La requête est une expression linguistique utilisée pour faire une demande d'information, ou demande faite en utilisant une telle expression. La requête joue un rôle important sur les résultats retourné dans la recherche d'image. Suivant les besoins de l'utilisateur, plusieurs requêtes sont proposées. Requête par mots clés, requête par esquisse (croquis), requête par image exemple.

- **Requête par mots clés**

L'utilisateur exprime ses besoins en fournissant un ou plusieurs mots-clés. Le système se base sur l'annotation textuelle d'images [2] .

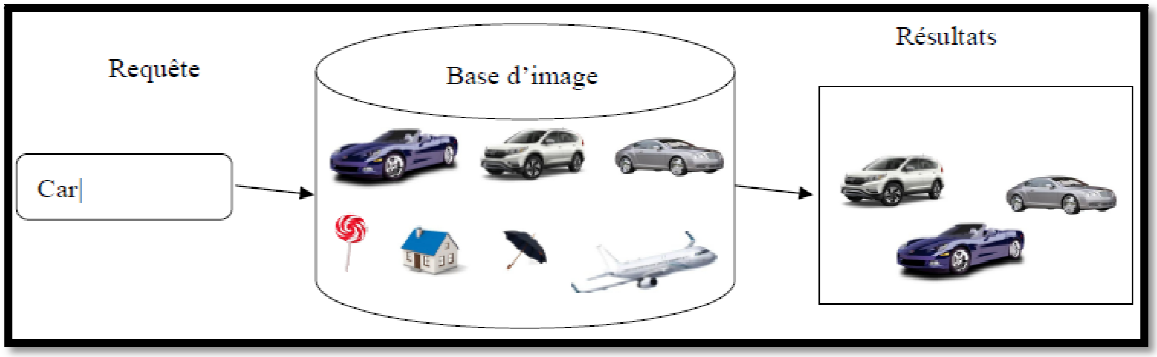


Figure 1.2 : schéma d'une Requête par mot clé

- **Requête par esquisse**

Dans ce cas, le système fournit à l'utilisateur des outils lui permettant de constituer une esquisse (dessin) correspondant à ses besoins. L'esquisse fournie sera utilisée comme exemple pour la recherche. L'esquisse peut être une ébauche de forme ou contour d'une image entière ou une ébauche des couleurs ou textures des régions d'une image. L'utilisateur choisira, en fonction de la base d'images utilisée, de ses besoins et préférences, l'une ou l'autre de ces représentations. Cette

technique présente l'inconvénient majeur qu'il est parfois difficile pour l'utilisateur de fournir une esquisse, malgré les outils qui lui sont fournis [2] .

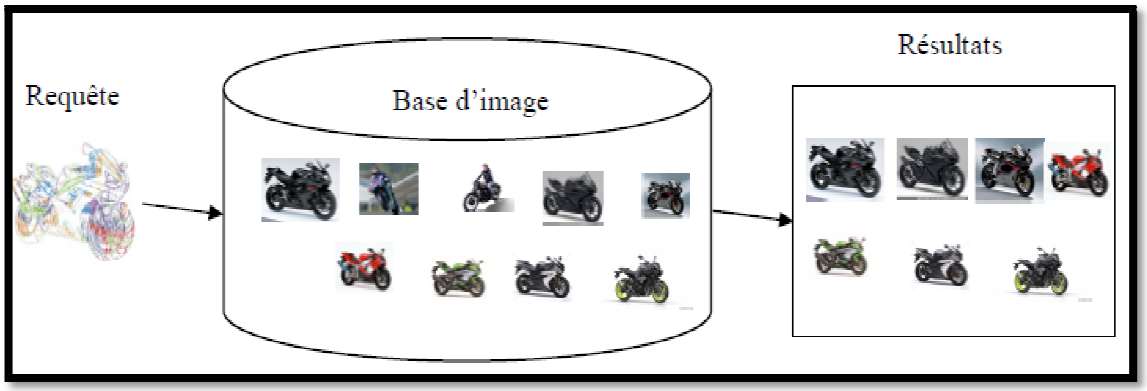


Figure 1.3 : schéma d'une Requête par esquisse

- **Requête par image exemple**

Pour les systèmes de recherche d'images à base d'exemples, l'utilisateur pour représenter ses besoins, utilise une image (ou une partie d'image) qu'il considère similaire aux images qu'il recherche. Cette image est appelée image exemple ou requête. L'image exemple peut soit être fournie par l'utilisateur, soit être choisie par ce dernier dans la base d'images utilisée. Cette technique est simple et ne nécessite pas de connaissances approfondies pour manipuler le système **[2] .**

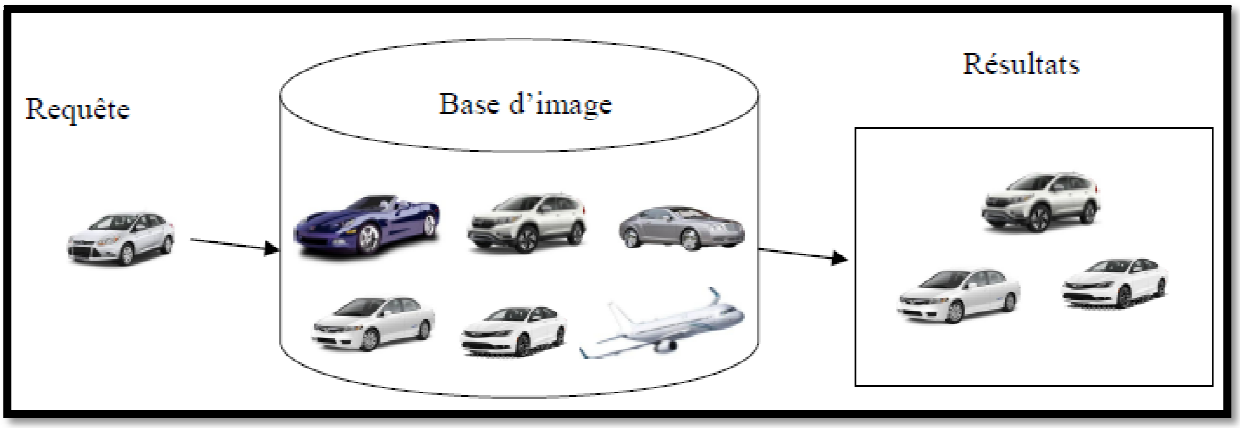


Figure 1.4: requête par image exemple.

6. Catégories de recherche d'image

Il ya deux catégories de recherche d'image: la recherche d'images par contenu (CBIR) et la recherche d'images par le texte (TBIR). Un système de recherche d'images par contenu accepte des requêtes image ou croquis, tandis qu'un système de recherche d'images par mots-clés accepte des requêtes textuelles.

6.1. Catégorie de recherche basée sur le contenu visuel (CBIR)

La recherche d'image par le contenu (en anglais : Content Based Image Retrieval ou CBIR) est une technique permettant de rechercher des images à partir de ses caractéristiques visuelles, c'est-à-dire induite de leurs pixels. Les images sont classiquement décrites comme rendant compte de leur texture, couleur, forme. **[**BEDOUHENE**]**

Il existe Plusieurs systèmes de recherche d'images basée sur le contenu comme QBIC, BLOBWORLD et KMED.

6.1.1. Composants d'un CBIR :

Nous décrivons brièvement ici les caractéristiques communes à la plupart des étapes : le traitement de la base d'images, les requêtes puis la mise en correspondance et la présentation des résultats. **La Figure 1.5** illustre l'ordonnancement de ces étapes :

Dans un premier temps, des descripteurs sont calculés à partir de chaque image de la collection (1) (le vocabulaire d'indexation).

Les données extraites (2) (à présent représentatives du contenu de l'image du point de vue du système constituent la base d'index (3).

Les requêtes de l'utilisateur (4) sont alors transformées afin d'être comparables avec la base d'index (5).

Une mise en correspondance (6) entre la requête transformée et la base d'index permet ensuite de produire le résultat de la requête (7).

Il se peut également que le système possède des composantes liées à la personnalisation, comme par exemple l'extraction, le stockage et l'utilisation d'un profil d'utilisateur [BEDOUHENE].

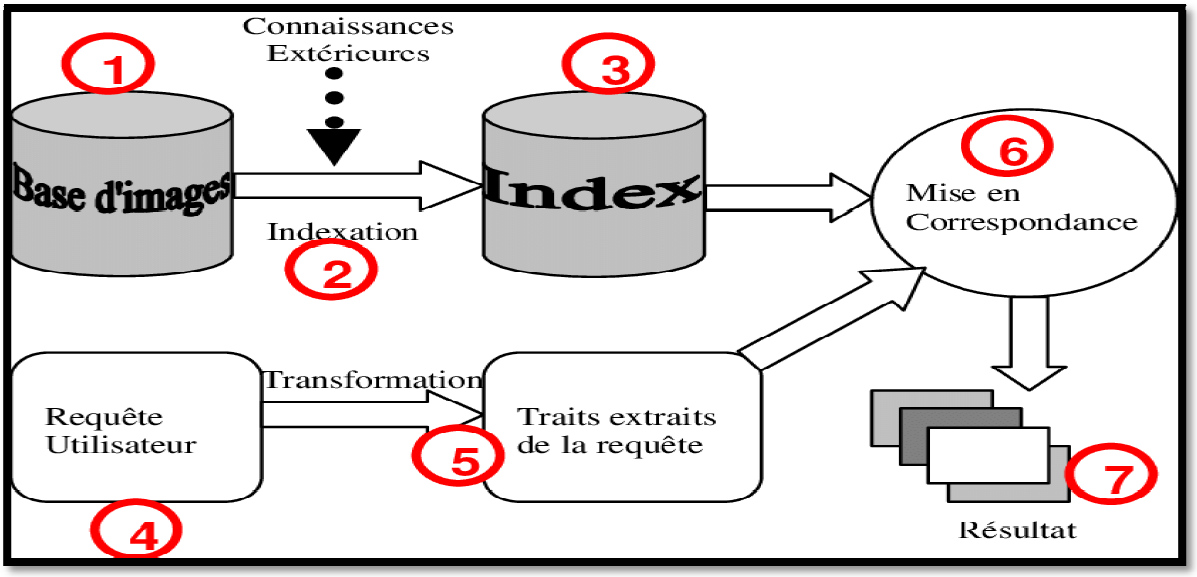


FIGURE 1.5 : Principaux composants d'un système de recherche par le contenu**. [**BEDOUHENE**]**

6.1.2. Les avantages et les inconvénients d'un CBIR

La recherche par le contenu comporte un certain nombre d'avantages, dont :

- Le fait qu'elle peut être utilisée même si la base d'image ne comporte aucune annotation.
- Elle s'applique bien aux images très complexes et celles qui contiennent une multitude d'objets qui ne peuvent être décrites avec du texte.
- Elle permet d'atteindre un niveau de raffinement que le texte ne permet pas.
- Le contenu des images est plus objectif que le texte.

Cependant la recherche par le contenu fait face à un certain nombre de défis tel que:

- L'extraction de caractéristiques visuelles : comment trouver les bonnes caractéristiques qui décrit mieux le contenu visuel d'une collection d'image ?
- Le fossé sémantique (*the semantic gap*) : Représente le problème de manque de lien entre le contenu visuel d'une image et les concepts sémantiques qu'on peut lui associer**[L.T.LAN, 2005]**.

6.2 Catégorie de recherche basée sur le contenu textuel (TBIR)

Un système de recherche d'images par texte TBIR (Text-Based Image Retrieval), comme son nom l'indique, vise à localiser automatiquement des images pertinentes à une requête utilisateur textuelle, en exploitant l'aspect sémantique associé aux images sous forme d'annotations.

Ainsi, un tel système identifie des images pertinentes en se basant sur une comparaison binaire entre la requête et les annotations des images de la base. Autrement dit, une image annotée explicitement par la requête est considérée comme pertinente, et une image qui ne contient pas la requête parmi ses annotations est écartée. Comme tout système de recherche d'images, un haut degré de précision et une recherche à grande vitesse (c'est-à-dire un temps de réponse court) sont deux critères principaux souhaitables pour la performance du système.

La plupart des systèmes de recherche d'images accessibles au grand public se basent sur des informations provenant d'annotations de l'image et sont totalement indépendants du contenu par exemple le système Chabot et Google.

6.2.1. Composantes de TBIR :

Un système TBIR s'articule sur trois composantes fonctionnelles principales qui sont : l'annotation des images (indexation textuelle sémantique), la formulation des requêtes et le calcul de pertinence.

6.2.1.1. Annotation des images :

Le processus d'annotation consiste à assigner à chaque image de la base un ou plusieurs mots clés (ou concepts) décrivant son contenu. Ces concepts peuvent ainsi décrire le bas niveau d'abstraction d'une image (tel que sa couleur sa forme), le niveau intermédiaire (tel que les objets contenus dans l'image) ou bien le haut niveau d'abstraction (tel que les scènes et les sensations). En fait, l'annotation est primordiale dans un TBIR classique, car les résultats de recherches sont profondément liés à la complétude et à la qualité des annotations. Autrement dit, une annotation de qualité permet au TBIR de récupérer avec succès le maximum d'images pertinentes et donc avoir un degré élevé de précision. Le processus d'annotation d'images peut être manuel, semi-automatique ou bien automatique [9].

6.2.1.2. Formulation des requêtes

En se basant sur le fait qu'un sens peut être véhiculé par des mots différents, l'expansion de la requête considère des mots (ou termes) reliés pour étendre la requête. Cette dernière est aussi vue comme un traitement pour "Cibler" le champ de recherche de cette requête.

6.2.1.3. Calcul de pertinence.

Après qu'un utilisateur ait introduit sa requête, le système TBIR prend la charge de localiser des images pertinentes à cette requête. Pour calculer la pertinence d'une image, un TBIR suit généralement un mécanisme très utilisé en recherche d'information, c'est la comparaison booléenne ou binaire entre les concepts composants la requête et ceux annotant chaque image de la base. Autrement dit, s'il s'agit d'une requête atomique, une image qui est annotée explicitement avec le concept requête est considérée comme pertinente (valeur de pertinence = 1), et donc retournée à l'usager, alors qu'une image qui ne l'est pas est considérée comme non-pertinente (valeur de pertinence = 0), et donc ignorée. Dans le cas d'une requête composée, une image est considérée comme pertinente si son annotation satisfait l'expression booléenne de la requête.

6.2.2. Les avantages et les inconvénients d'un TBIR

La recherche par le texte comporte un certain nombre d'avantages, dont :

- Le système TBIR fonctionne rapidement, c'est-à-dire qu'il récupère rapidement les images de la base de données.

Cependant la recherche par le texte fait face à un certain nombre de défis tel que:

- Si un mot clé spécifié dans la requête est manquant, des images inappropriées seront trouvées.
- Les annotations manuelles ne sont pas toujours disponibles.
- L'annotation d'image est un processus qui prend du temps.

7. Conclusion

Dans ce chapitre, nous avons présentés quelque concept fondamental de l'image et de la recherche d'images. Ensuite nous avons présenté l'architecture générale les catégories de recherche d'image.

Dans le chapitre suivant, nous présentons la représentation de la requête et Word embedding.

Chapitre II : Représentation de requête et Word embedding

1. Introduction

Le choix d'une bonne représentation des mots est capital pour optimiser les résultats des taches de traitement automatique de langage. La méthode la plus couramment utilisée est celle du sac de mots ou les documents sont représentes par les vecteurs de fréquences des mots qui les composent.

Le Word Embedding est capable de représenter des mots par un vecteur de réels continus qui correspond à la position du mot dans un espace multidimensionnel.

Parmi ces approches la méthode Word2Vec se présente dans l'état de l'art comme une des méthodes les plus récentes et a prouve son efficacité a travers plusieurs taches de traitement automatique de langages .**[Tanjona,2016]**

Nous présentons dans ce chapitre un port sur la représentation de sac de mots et leurs problèmes, Ensuite nous avons passé au Word embedding et leurs modèles, nous focalisons sur le modèle de word2vec avec sa définition, les modèles et les domaines d'application.

2. Sac de mot (BOW)

La représentation en sac de mots est de loin la plus suivie dans la recherche et l'industrie. La plupart des articles de recherche sur la classification de textes se basent sur cette technique de représentation.

Comme son nom l'indique, cette représentation vise à représenter le texte avec une liste de mots simples contenus dans ce dernier. Suivant les différentes étapes d'extraction et de sélection certains mots comme les mots vides ne font pas partis du sac.

A chaque mot prend une valeur binaire (présence = 1, absence = 0), une fréquence d'apparition dans le document ou encore le nombre d'occurrences dans le texte.

L'intérêt de cette représentation est sa simplicité et ses performances. De nombreux sous domaines de l'apprentissage tels que la catégorisation des objets, la détection de spam, le "topic modelling" ou encore la biologie l'utilisent.

Cependant cette représentation soulève plusieurs problèmes notamment identifiés par :

- **La polysémie :** suivant le contexte, un mot peut avoir des sens différents. Par exemple, le mot "orange" peut se rapporter à la couleur, le fruit, la ville du sud de la France ou encore l'opérateur téléphonique. En suivant le paradigme du sac de mots, un article sur la ville d'Orange peut être catégorisé dans une catégorie traitant de fruits.
- **La synonymie :** La synonymie est des mots qui ont exactement le même sens ou un sens extrêmement proche. Par exemple, les synonymes de "maison" peuvent être: "un logement", "une baraque" (familier), mais aussi "une demeure", un "pavillon".
- **Les liens sémantiques :** Les mots proches sémantiquement sont considérés comme n'ayant aucun lien. Par exemple, les mots "animal" et "chat" sont liés par une relation hyperonymie mais restent considérés comme distincts l'un de l'autre.
- **Division de groupes de mots :** La représentation en sac de mots propose de décrire un texte par certains de ses mots. Si le texte contient la suite de mots "poste de police", les mots retournés seront "poste" et "police" en considérant que le mot vide "de" est supprimé automatiquement. Le mot "poste" ne se réfère pas forcément au domaine de la police. **[Jean-Charles, 2017]**

3. Word embedding

Le Word embedding est une manière de représenter des mots comme des vecteurs, typiquement dans un espace de quelques centaines de dimensions.

Le vecteur représentant chaque mot est appris via un algorithme itératif, à partir d'une grande quantité de texte, il a été massivement exploités ces dernières années, notamment pour l'apprentissage d'une représentation de mots .

3.1. Modèles de Word embedding

Il existe plusieurs modèles pour entrainer de Word embedding. Les plus connus Glove, FastText et Word2vec.

- **GLOVE :** Les vecteurs globaux pour la représentation des mots (Glove) sont fournis par Stanford. Ils ont fourni divers modèles de 25, 50, 100, 200 à 300 dimensions basées sur 2, 6, 42, 840 milliards de jetons.

L'équipe a utilisé la cooccurrence mot à mot pour construire ce modèle. En d'autres termes, si deux mots se répètent plusieurs fois, cela signifie qu'ils ont une similarité linguistique ou sémantique. ***[3]***

- **FastText** : Ce modèle est développé par Facebook. Ils fournissent 3 modèles de 300 dimensions chacun. FastText est capable d'atteindre de bonnes performances pour les représentations de mots et les classifications de phrases car elles utilisent des représentations au niveau des caractères.

Chaque mot est représenté comme un sac de caractères n-grammes en plus du mot lui-même. Par exemple, pour le mot partiel, avec n = 3, la représentation fastText du caractère n-grammes sont ajoutés en tant que symboles de limite pour séparer les n-grammes du mot lui-même. ***[3]***

- **Word2Vec :** Ce modèle est fourni par Google et est formé sur les données de Google Actualités. Ce modèle a 300 dimensions et est formé sur 3 millions de mots à partir des données de Google Il est sorti en 2013.***[3]***

3.2. Différence entre les modèles de Word embedding

Word2vec	**FastText**	**Glove**
* développé par Google * traite chaque mot du corpus comme une entité atomique * génère un vecteur pour chaque mot * utilise le réseau de neurone	* développé par Face book * traite chaque mot comme composé de n-grammes de caractères * le vecteur d'un mot est constitué de la somme de ce caractère n-grammes	* développé par Stanford * il tente de capturer le nombre de statistiques globales combien de fois il apparaît. * utilise pas de réseau de neurone

Tableau N°2.1 : La différence entre les modèles de Word embedding

Nous avons utilisé Word2Vec pour mettre en évidence des concepts similaires, rechercher des concepts liés et capturer différents degrés de similarité entre les mots.

4. Le modèle Word2Vec

Word2Vec est un outil d'apprentissage non supervisé mis au point par Tomas Mikolov et son équipe (Mikolov, T., Chen, K., Corrado, G., et Dean, J., 2013) ayant pour but d'apprendre les relations sémantiques entre les mots. Cet outil permet de créer une représentation vectorielle de l'ensemble des mots d'un texte. Il se présente sous deux formes : le modèle du sac continu de mots (CBOW) et le modèle de Skip-Gram. [Ghannay, 2017]

L'architecture du Word2Vec, comme illustré dans la figure 2.1 est compose de plusieurs couches. La couche d'entrée de Word2Vec est constituée d'un grand texte (corpus) traité dans la couche cachée et la couche sortie est une liste de mots similaires. Lorsque Word2Vec reçoit plus de texte, il devient plus intelligent et donne de meilleurs résultats.

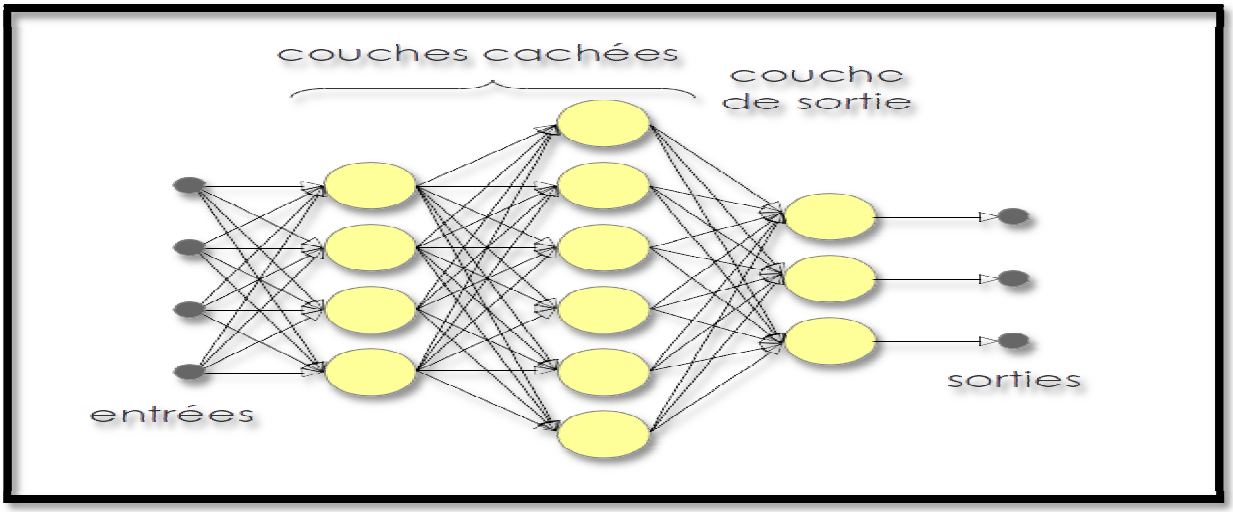


Figure 2.1 : Architecture de word2vec

4.1. Les modèles de Word2vec

Le modèle du sac continu de mots (CBOW) et le modèle de Skip-Gram sont des modèles similaires, sauf que CBOW prédit les mots cibles par exemple «tapis» à partir des mots de contexte source «le chat s'assoit sur le», tandis que le skip-gram fait l'inverse et prédit les mots de contexte source de la cible mots. Cette inversion peut sembler un choix arbitraire,

mais statistiquement elle a pour effet que le CBOW lisse une grande partie de l'information distributive (en traitant un contexte entier comme une observation).

Pour la plupart, cela s'avère être une chose utile pour les jeux de données plus petits. Cependant, skip-gram traite chaque paire contexte-cible comme une nouvelle observation, ce qui a tendance à être meilleur lorsque nous avons des ensembles de données plus volumineux. Dans ce qui suit, nous allons présenter les deux architectures CBOW et Skip-gram illustrées par la figure 2.2.

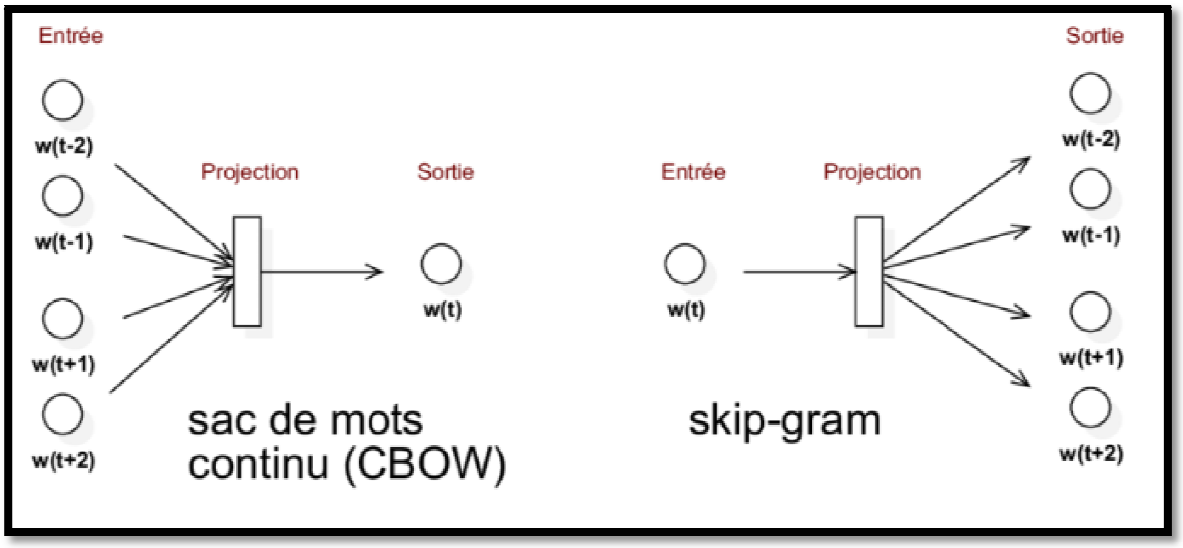


Figure 2.2: Architectures CBOW et Skip-gram du modèle word2vec. [13]

4.1.1. Le modèle de CBOW

La première architecture permet de prédire un mot (w_i) en fonction de son contexte, qui correspond aux mots précédents et aux mots suivants. Dans cette architecture, la couche de projection est partagée par tous les mots : tous les mots sont projetés dans la même position.

Ici, l'apprentissage des Word embedding consiste à prédire un mot en fonction de son contexte. Ceci est fait en calculant la somme des Word embedding du contexte, puis en appliquant sur le vecteur résultant un classifieur log-linéaire pour prédire le mot cible.[Ghannay, 2017]

Enfin, le modèle compare sa prédiction avec la réalité et corrige la représentation vectorielle du mot par rétro-propagation du gradient. Ce modèle cherche à maximiser l'équation suivante :

$$S = \frac{1}{I}\sum_{i=1}^{I} \log p(m_i | m_{i-n}, \ldots, m_{i-1}, m_{i+1} \ldots, m_{i+n})$$

Où I : correspond au nombre de mots dans le corpus, le modèle reçoit une fenêtre de N mots autour du mot cible (m_i). Ce modèle est nommé CBOW (sac-de-mots continus) ou (Continuous Bag-of-Words) pour deux raisons. D'une part, l'appellation sac-de-mots indique que l'ordre des mots du contexte n'a pas d'influence sur la projection. D'autre part, l'objectif souligne l'utilisation d'une représentation continue (Word embedding) des mots en contexte, ce qui diffère des modèles sac-de-mots standards. [Ghannay, 2017]

4.1.2. Le modèle de Skip-gram

L'architecture skip-gram est l'inverse de CBOW. Le mot cible (m_i) est maintenant utilisé comme entrée, et les mots de contexte sont utilisés en sortie. Elle consiste à prédire pour un mot donné le contexte dont il est issu.

Dans cette architecture le mot en entrée est projeté dans la couche cachée puis dans la couche de sortie pour produire un vecteur. Ce vecteur est ensuite comparé à chacun des mots du contexte et le réseau se corrige par rétro-propagation du gradient. De cette manière, la représentation vectorielle du mot d'entrée va être proche de celles des mots qui produisent le même contexte que lui. Cette architecture maximise l'équation suivante :

$$S = \frac{1}{I}\sum_{i=1}^{I} \sum_{-n \leq j \leq n, \neq 0} \log p(m_i | m_{i+j})$$

L'architecture Skip-gram avec échantillons négatifs (Négative Sampling) cherche à représenter chaque mot (m_i) $\in$ VI et chaque contexte c $\in$ VC par des vecteurs de d-dimensions ($\vec{m}_i$, $\vec{c}$), afin d'avoir des représentations vectorielles similaires pour les mots similaires. Ceci est fait d'une part, en maximisant le produit scalaire $\vec{m}_i . \vec{c}$ pour les paires (m_i, c) qui apparaissent dans le corpus D, d'autre part, en minimisant les exemples négatifs (m_i,c_{Neg}),qui ne sont pas nécessairement observés dans D. [Ghannay, 2017] .

Les exemples négatifs sont créés en corrompant stochastique ment les paires (m_i, c) observées dans D, d'où le nom d'échantillons négatifs. En outre, le contexte ne se limite pas au contexte immédiat, et les instances d'apprentissage peuvent être créées en ignorant un nombre de mots dans son contexte, par exemple, m_i-4, m_i-3, m_i+3, m_i+4, d'où le nom de Skip-gram. [Ghannay, 2017]

Un exemple de ces deux architectures est présenté dans la figure 2.3 extraite de :

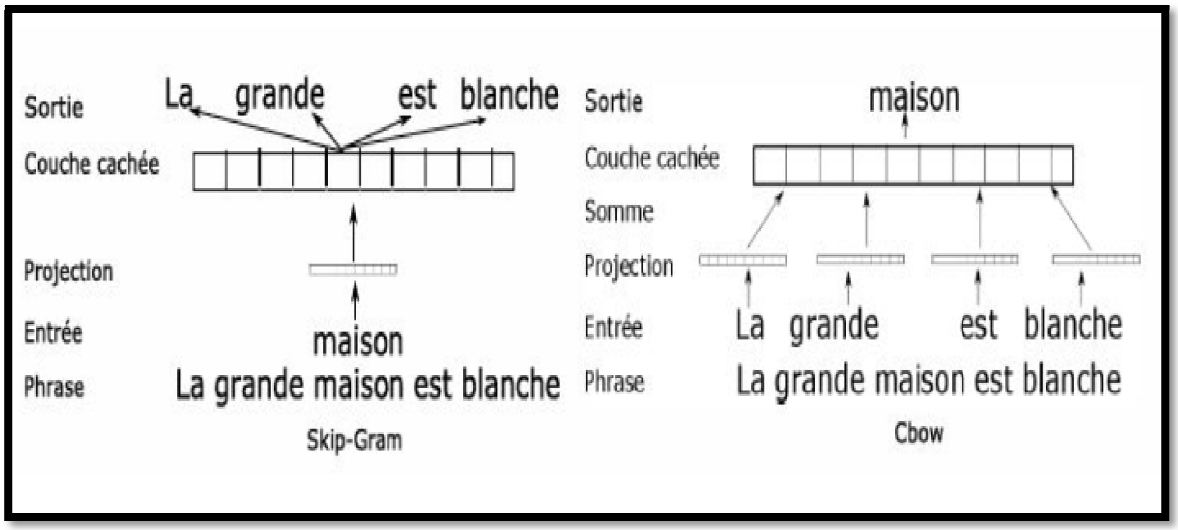


Figure 2.3: Exemples des architectures CBOW et Skip-gram de Word2vec. [5]

4.2. Les domaine d'applications de Word2Vec

Il existe de nombreux domaines d'application de word2vec

4.2.1. Similarité des mots

Méthodes classiques: Modifier la distance, Porters, Stemmer et Lemmatisation à l'aide de dictionnaires.

- identifie facilement les mots et les synonymes similaires, car ils apparaissent dans des contextes similaires.
- Racine (pensée -> penser).
- Inflexions, formes tendues.
- par exemple. Penser, penser, réfléchir, réfléchir,
- par exemple. Avion, avion, vol

4.2.2. Traduction automatique

Les premières méthodes de traduction se basent sur l'apprentissage automatique de modèles basées sur des règles grammaticales et syntaxiques.

La traduction pouvant être considérée comme l'´etude sémantique des similarités entre deux langues, plusieurs méthodes basées sur Word2Vec ont ´et´e proposées. Les auteurs dans entrainent un modèle Word2Vec pour chaque langue, une transformation géométrique est par la suite calculée entre les espaces vectorielles issus des deux langages. Plus récemment,

propose d'entrainer un modèle Word2Vec directement `a partir de couple de mot, par exemple ´étant donnes les couples de mots ≪ Athènes— Athens ≫ ≪ Grece — Greece ≫ et la question ≪ Londres — London ≫ la tache consiste a prédire ≪ Angleterre — England ≫, la figure suivante représente la Regroupement des vecteurs word2vec de mots en anglais et en espagnols. [4]

4.2.3. Analyse des sentiments

L'objectif de l'analyse des sentiments est de déduire a partir de grande quantité de données textuelles les sentiments qui y sont exprimes. La détection de sentiments peut se faire à différents niveaux : unité textuelle, paragraphe, document. Une première utilisation de Word2vec a été introduite en représentant une phrase par un vecteur en faisant la moyenne des vecteurs des mots la composant.

Les vecteurs obtenus sont utilises comme caractéristiques pour l'apprentissage de classifieur automatiques pour l'analyse de sentiments. Une modification de Word2Vec a été proposée dans pour représenter des phrases ou paragraphes par des vecteurs. [5]

4.2.4. Reconnaissance d'Entités Nommées (EN)

La tache de reconnaissance d'EN est une sous tache du domaine d'extraction d'information consistant à identifier et a catégoriser certaines expressions linguistiques autonomes et mono-référentielles.

La première approche s'appuie sur l'utilisation de grammaires formelles construites à la main.

La seconde démarche fait usage de techniques statistiques ou encore dites à base d'apprentissage pour apprendre des spécifiés sur de larges corpus de textes.

SK Siencnik et al ont introduit différentes approches d'utilisation de Word2vec comme caractéristiques d'apprentissage et ont montre a travers leurs expérimentations une possible amélioration de la tache de reconnaissance d'EN. [5]

4.2.5. Regroupement (Clustering)

Cette méthode consiste à regrouper les mots en clusters en s'appuyant sur leurs contextes .L'approche de Brown s'appuie sur l'utilisation d'un algorithme hiérarchique qui regroupe les

mots pour maximiser l'information mutuelle des bi-grammes. La nature hiérarchique du regroupement signifie que nous pouvons choisir la classe de mots à plusieurs niveaux dans la hiérarchie, ce qui peut ne pas pénaliser les clusters ayant un petit nombre de mots. [5] .

5. Conclusion

Dans ce chapitre, nous avons présentés la représentation de la requête et ses types, et aussi la représentation de sac de mots, ensuit, nous expliquons le Word embedding et ses modèle, puis nous présentons le fonctionnement de word2vec, les modèles et quelle que applications.

Dans le chapitre suivant nous présentons les approches de systèmes de recherche images.

Chapitre III: *L'approche proposée*

1. Introduction

Dans ce chapitre, nous expliquerons l'approche proposée en détail. Nous commençons par présenter le modèle word2vec (Skip-Gram), puis les étapes de prétraitement de la requête.

Ensuite, nous abordons le processus d'extraction des mots similaires pour chaque mot de la requête à l'aide de skip-gram. Et la dernière étape de notre travail c'est la recherche pour obtenir le résultat.

2. Approche proposée

Notre approche consiste à définir une méthode d'amélioration de la requête, en utilisant word2vec comme une technique principale qui permet d'ajouter des termes à la requête initiale pour obtenir des résultats précis.

Nous avons développé une application qui enrichir la requête de recherche d'images

Tout d'abord, nous présentons les différentes phases suivies pour concevoir l'approche proposée, en expliquant en outre les détails de chaque phase. La figure suivante illustre le diagramme général de la démarche de notre approche.

La mise en œuvre fonctionnelle de l'approche s'appuie sur plusieurs modules pour l'application de l'algorithme Word2vec afin d'améliorer les requêtes.

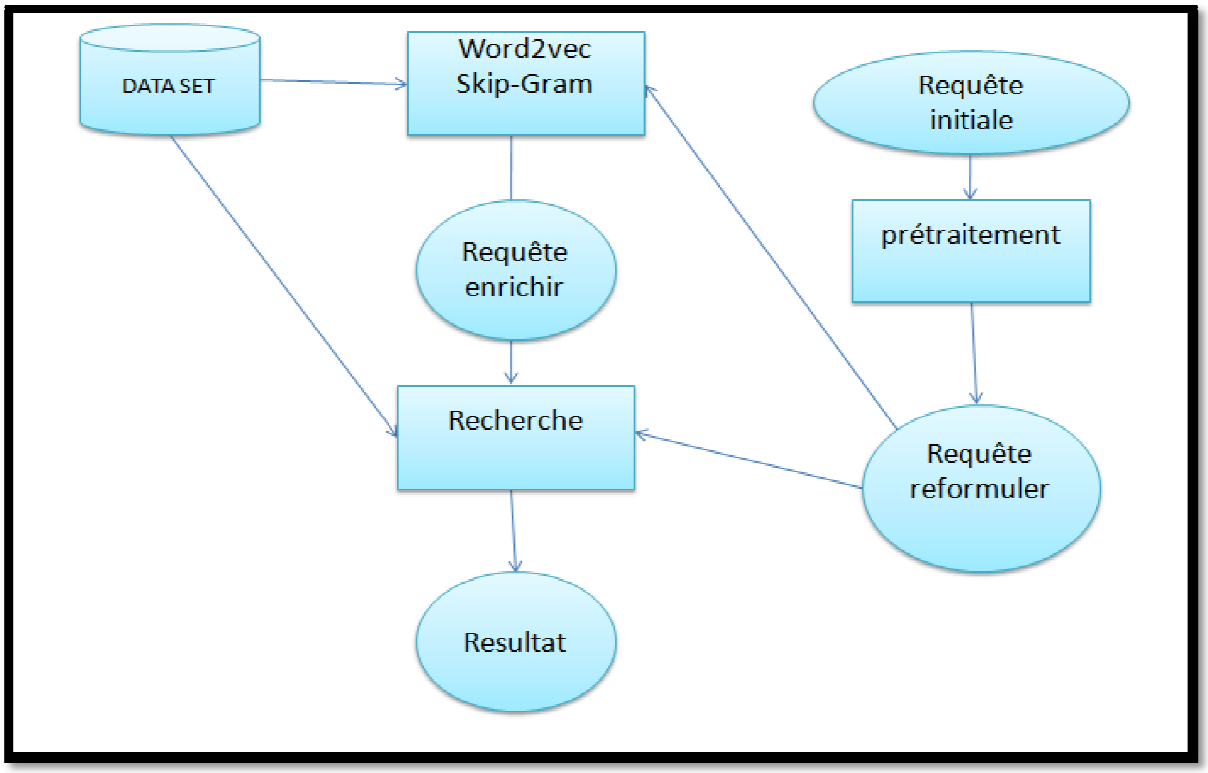


Figure 3.1: Architecture générale de notre approche proposée.

2.1. Modèle Word2Vec

Word2Vec contient deux modèles cbow et skip-Gram. Nous avons choisi le modèle Skip-Gram dans notre travail car il est le plus proche pour extraction des mots similaires.

Dans l'architecture de skip-gram, le modèle prédit les mots de contexte environnants dans une fenêtre donnée, en fonction du mot actuel. La couche d'entrée contient le mot actuel et la couche de sortie contient les mots de contexte. La couche masquée contient le nombre de dimensions dans lesquelles nous voulons représenter le mot actuel présent dans la couche en entrée.

L'architecture du Skip-Gram pèse plus lourdement les mots de contexte proches que les mots de contexte plus éloignés selon la note des auteurs.

```
file_name="C:\\Users\\Admin\\Desktop\\DATASET2.csv"
data = pd.read_csv(file_name, usecols=['TITLE','DESCRIPTION','NOTES'],delimiter=";",encoding="utf-8")

l=data.values.tolist()
input_list=[item for sublist in l for item in sublist]
input_list=[item for item in input_list if item not in '\n']
documents=[remove_punct(item) for item in input_list ]

stoplist = set('for a of the and to in'.split())
#print (input_list)

#print(input_list)

texts = [
    [word for word in document.lower().split() if word not in stoplist]
     for document in documents
    ]
#print (texts)

model = gensim.models.Word2Vec(texts, iter=100 , min_count=1,sg=0,workers=6)
model.save(r"C:\Users\Admin\Desktop\mymodelskipgram")

```

Figure 3.2 : création le modèle de word2vec (Skip-Gram)

La fonction objective du modèle skip-gram c'est de maximiser le journal du produit scalaire du vecteur mot et de son vecteur mot contextuel.

Le produit scalaire et la similarité de cosinus et sont des mesures de similarité, mais le produit scalaire est sensible à la magnitude, contrairement à la similarité cosinus. Selon le nombre d'occurrences d'un mot, il peut avoir un produit scalaire petit ou grand avec un autre mot. Nous normalisons notre vecteur pour éviter cet effet, de sorte que tous les vecteurs ont une magnitude unitaire. Mais si tâche en aval nécessite un compte d'occurrence en tant que fonction, le produit scalaire pourrait être la solution, nous pouvons simplement calculer la similarité en cosinus qui les normalisera.

Après avoir transformé un mot en vecteurs, le cosinus de l'angle entre les vecteurs de mots peut être utilisé pour comparer le degré de ressemblance des mots. Plus un angle est petit, plus la valeur du cosinus sera grande (plus proche de 1). Dans ce cas, une valeur de cosinus supérieure indique les vecteurs de mots qui ont significations similaires.

Lorsque l'angle entre eux est grand, le cosinus de l'angle entre eux est faible, ce qui signifie que ces mots ne sont pas similaires.

Nous avons entraîné le modèle skip-Gram en utilisant une base de données contient des images annoter.

2.2. Prétraitement de la requête

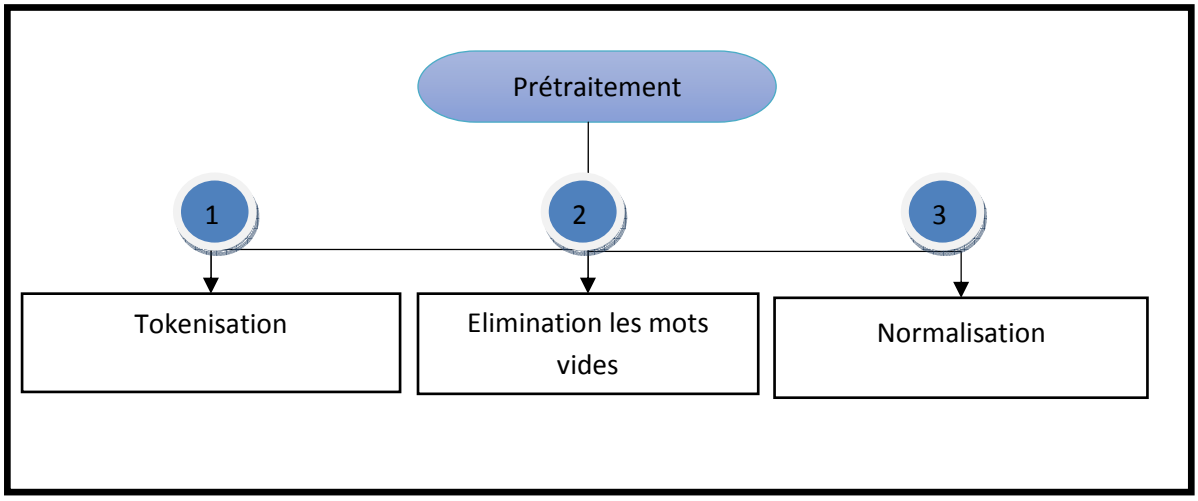
Les requêtes sont constituées de 2 parties: Mots vides, mots centraux (mot clé).

- **Mots vides:** ils contiennent plusieurs types tels que : Noms de référence (This, that,..), Noms associés (who..) et prépositions (for, of ,in,..)
- **Mots centrales** : Ce sont les mots clés dans la requête. Exemple : flower, exercise ,……ect.
- Exemple : la requete : **The Child at home.**

Mot	**Type**
The	Mot vide
child	Mot clé
at	Mot vide
home	Mots clé

Tableau 3.1 : requête exemple

Pour prétraiter la requête, il est nécessaire de suivre les étapes suivantes illustrées dans la figure



:

Figure 3.3 : Les étapes pour prétraiter la requête

2.2.1.Tokenisation

Nous avons découpé le texte en unités de type des mots appelées « tokens »ensuite nous avons délimité les tokens par des espaces.

Le processus de tokenisation que nous avons utilisé est décrit par le pseudo algorithme suivant :

```
Algorithme tokenisation :
S : [] tableau de chaine de caractères ; I, J=entier, Ch.='' ; T : [] tableau de chaine de caractères
Début :
Tant que (s[i] <>\0) faire
Si s[i] =''
        Alors i++
Fin si
      Tant que(S[i] <>'') et (S [I] <>\0) faire
Concaténation (Ch, S [I])
        I++
Si (ch. <>'')
           Alors T[J ]←Ch.
J ++   Fin si
Fin
```

Input : The city is pretty much

Output : The city is pretty much

2.2.2. Elimination les mots vides :

Les mots vides sont généralement les plus fréquents dans un texte comme : the, or, a, you, i, us,…etc.

Les instructions d'elimination des mots vides décrites par le pseudo-algorithme suivant :

```
Algorithme supp Mot vide :
Mot vide :[ the,or,a,you,i,us], i :entier
Début :
Tant que ( s[i] <>'\0') faire
X= tokenisation(s [i])
i ++
Si x in Mot vide
Alors   suppression(x)
Fin si
Fait
```

2.2.3. Normalisation

Dans notre projet, nous avons utilisé trois techniques de normalisation:

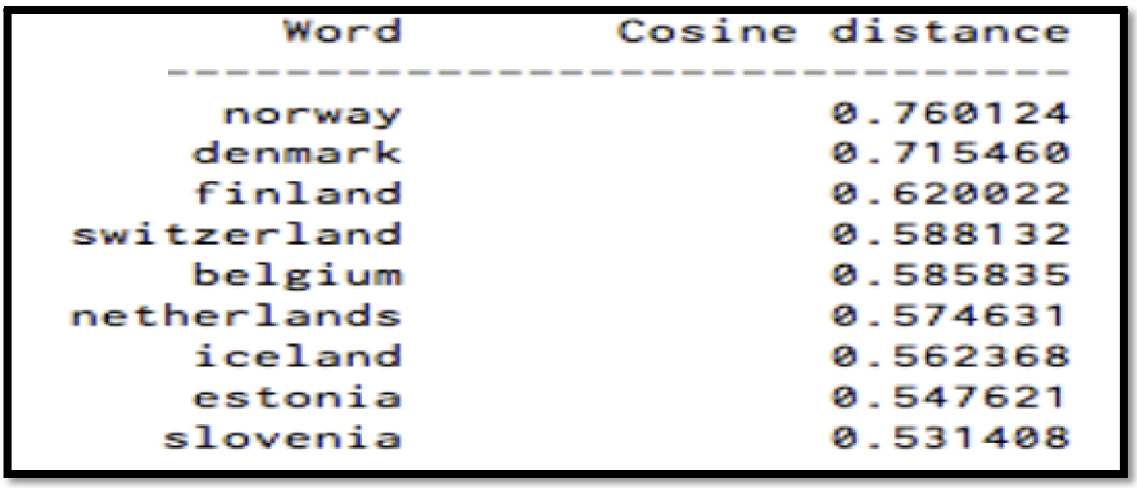
- **Normalisation par lemmatisation :** Exemple: taken → take
- **Normalisation par racinisation :** Exemple :retrieve, retrieving, retrieval, retrieved, retrieves → retriev.
- **Normalisation par troncature :** Exemple : troncature à 7 caractères : economically → économi

2.3. Enrichissement de la requête

L'enrichissement de la requête se fait à travers l'extraction des mots similaires en utilisant le modèle de Word2Vec (Skip-Gram), puis chaque mot similaire prend la valeur de similitude par rapport le mot d'origine.

Pour calculer la similarité, nous utilisons la bibliothèque genism qui est chargée d'extraire des mots similaires, lesquels extraient pour chaque mot N mots similaires avec leurs valeurs de similarité. Genism utilise la fonction cosinus pour calculer la similarité par défaut.

Par exemple, les mots similaires pour le mot « **Sweden** » sont montrés dans la figure suivante :



Word	Cosine distance
norway	0.760124
denmark	0.715460
finland	0.620022
switzerland	0.588132
belgium	0.585835
netherlands	0.574631
iceland	0.562368
estonia	0.547621
slovenia	0.531408

Figure 3.4: les mots similaires de " Sweden " par Word2Vec (Skip-Gram).

L'algorithme adopté d'extraction des mots similaires:

Entrée: Requête utilisateur (Q), Skip-Gram_Model (WM)

Sortie: Requête enrichir (QE)

Étape 1: Extrayez des mots-clés W_1, W_2, ..., W_n de la requête de l'utilisateur Q.

Étape 2: pour I de 1 à n

Nous utilisons Skip-Gram pour extraire les N premiers termes équivalents du point de vue des mots-clés. Pour les mots-clés W1, W2,……., Wi, et les mots sémantiquement équivalents sont W'_{i1}, W'_{i2}, ... W'_{im}.

Étape 3: Construire une nouvelle requête à l'aide d'opérateurs booléens "ET" et "OU" comme (W'11 | W'12 |... | W'1m) + (W'21 | W'22 |... | W'2m) + ... + (W'n1 |W'n2 |... | W'nm).

Skip-Gram peut créer une représentation distribuée très riche de mots qui réserve également les relations sémantiques que les mots ont avec d'autres mots. Skip-Gram peut comprendre fondamentalement la signification d'un mot basé sur les apparences passées.

Cela signifie que nous pouvons maintenant regrouper des mots, trouver des mots similaires et rechercher des relations similaires de mots différents, etc. [6]

3. Conclusion

Dans ce chapitre, nous avons décrit notre approche proposée, qui vise à améliorer la recherche d'image. Notre approche consiste à enrichir les requêtes à travers Word2vec.

Notre approche sera mise en œuvre dans le chapitre suivant.

Chapitre IV : *Implémentation et Evaluation*

1. Introduction

Après avoir exposé les différents axes requis pour la compréhension du contexte de notre approche.

Dans ce dernier chapitre, nous allons présenter la mise en œuvre de notre projet. Nous commençons tout d'abord par l'environnement de développement ainsi que la présentation de langage de programmation, en détaillant les différents outils et matériaux utilisés dans chaque étape. Puis nous montrons les différentes étapes de déroulement de l'application et évaluons enfin les résultats obtenus.

2. Environnement de développement

Chaque projet d'informatique dépend sur plusieurs ressources et outils car chaque ressource va nous aider dans un point donnée, pour cela nous présentons les différents langages de programmation ainsi que les outils de développement:

2.1. Langage utilisée

Le choix du langage Python et java utilisé dans le projet n'est pas une coïncidence car le Python et java est connu mondialement et notre choix est bien expliqué dans les prochaines lignes.

2.1.1. Python

Python est un langage de programmation de haut niveau pour la programmation à usage général. Créé par Guido van Rossum et premièrement sorti en 1991, Python a une philosophie de conception qui met l'accent sur la lisibilité du code , notamment en utilisant des espaces importants. Il fournit des constructions qui permettent une programmation claire sur les petites et grandes échelles [7] .

Python dispose d'un type dynamique de système automatique de gestion de la mémoire. il prend en charge plusieurs paradigmes de programmation, y compris orienté objet, impératif, fonctionnel et procédural, et dispose d' une grande bibliothèque complète et standard .

2.1.1.1. Bibliothèques de Python utilisés

- **Gensim :**

Gensim est une bibliothèque libre de Python conçue pour extraire automatiquement des relations sémantiques des documents, pour traiter les textes numériques bruts et non-structurés. Les algorithmes de gensim, tels que Word2vec, n'ont besoin que d'un corpus de documents en texte brut.

Nous avons utilisé le «genism» pour travailler avec toutes les fonctions liées aux algorithmes word2vec.

- **Pandas**

Pandas est une bibliothèque de manipulation de données basée sur Numpy qui fournit de nombreuses fonctions utiles pour accéder, indexer, fusionner et le regrouper des données.

La structure de données principale (Data Frame) est proche de ce qui peut être trouvé dans le paquet statistique ; autrement dit, des tableaux de données hétérogènes avec l'indexation par nom, les opérations sur les séries temporelles et l'auto-alignement des données [8] .

- **NLTK**

NLTK est une plate-forme leader pour la création de programmes Python compatibles avec les données de langage humain. Il fournit des interfaces faciles à utiliser pour les corpus et ressources lexicales, ainsi qu'une suite de bibliothèques de traitement de texte pour la classification, la tokenisation. [9]

2.1.2. Java

Java est un langage de programmation orienté objet, développé par Sun Microsystems. Il permet de créer des logiciels compatibles avec de nombreux systèmes d'exploitation (Windows, Linux, Macintosh, Solaris). Java donne aussi la possibilité de développer des programmes pour téléphones portables et assistants personnels. Enfin, ce langage peut-être utilisé sur internet pour des petites applications intégrées à la page web (applet) ou encore comme langage serveur (jsp)**.[10]**

2.2. Les outils utilisées

Nous avons utilisé deux outils pour notre travail Anaconda et Netbeans.

2.2.1. Anaconda/Spyder

Anaconda est une plateforme logicielle libre comprenant l'environnement Python Complet (V 3.6) et de très nombreuses librairies (numpy, pylab …). Elle tourne sur tous les OS courants dont Windows.

Anaconda est fourni avec l'environnement de travail SPYDER (Scientific Python Development EnviRonment) .

La figure suivante représente l'interface d'un programme python sous anaconda :

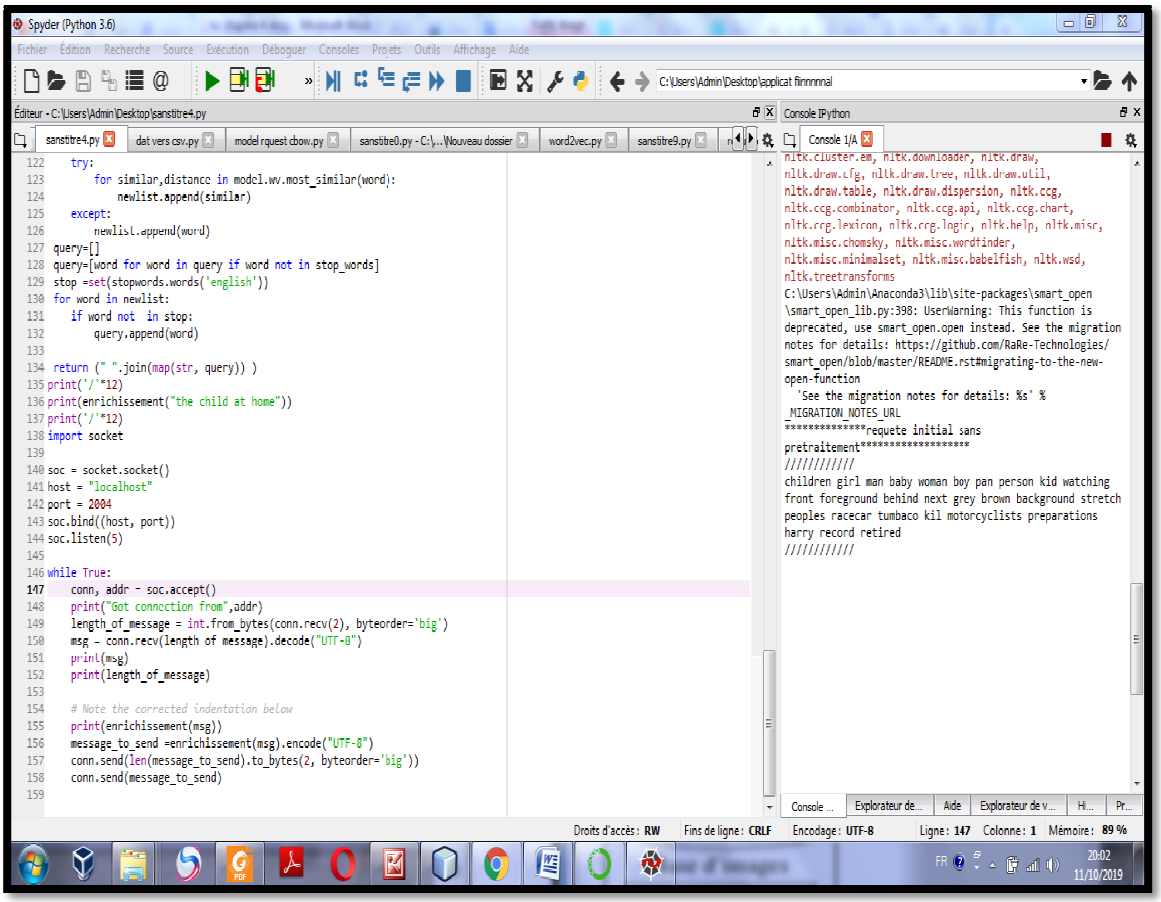


FIGURE 4.1 : Interface d'un programme python sous anaconda

2.2.2. NetBeans

NetBeans est un environnement de développement intégré et open source qui nous offre la manipulation des différents langages de programmation et qui facilite le travail. Dans notre projet, nous utilisons NetBeans (version Netbeans IDE 8.1) pour créer une application pour la recherche d'images en utilisant le langage Java

La figure suivante représente l'interface d'un programme Java sous NetBeans

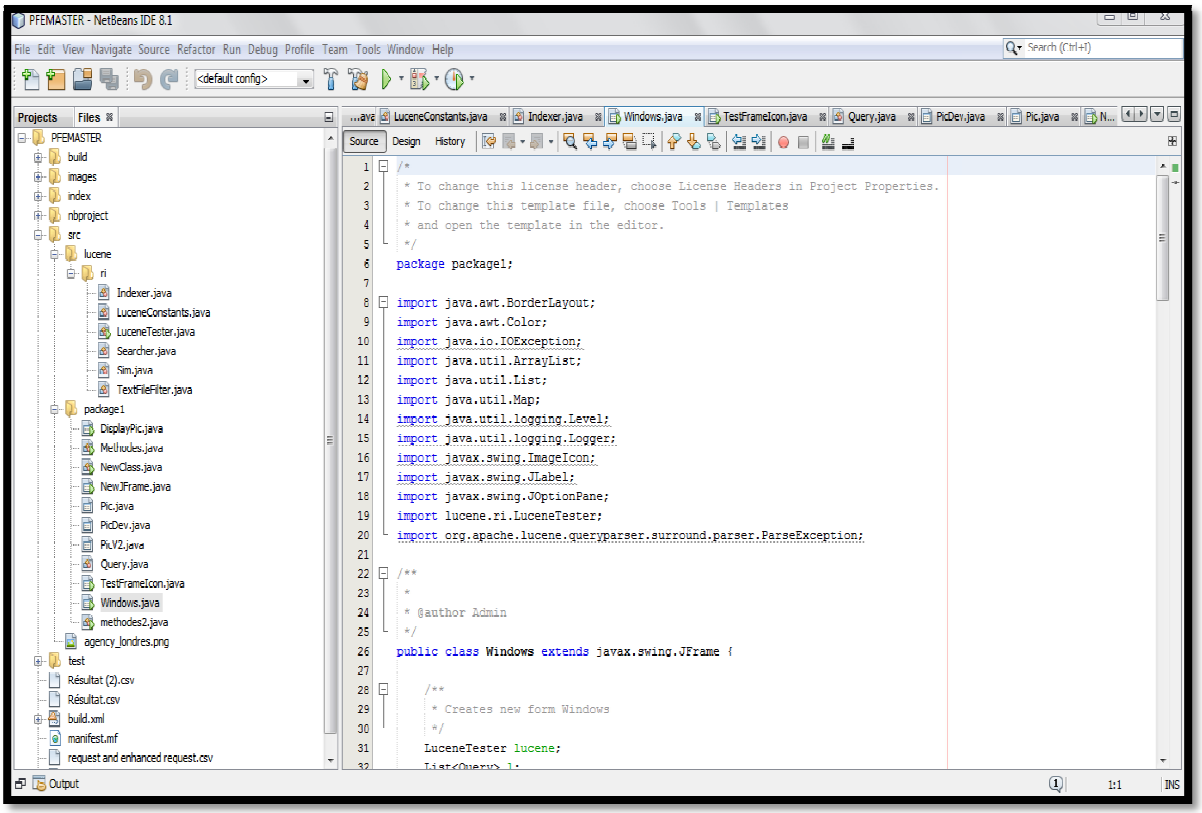


Figure 4.2: Interface d'un programme Java sous NetBeans

2.3. Matériaux utilisés

Pour de développement de notre application, nous avons utilisé les machines suivante :

- **La machine 1**

  Nom de l'ordinateur : LENOVO-PC

  Processeur : Intel(R) Core(TM) i3-3110M CPU @ 2.40GHz 2.40 GHz

  Mémoire installée (RAM) : 4.00 Go

  Type du système : Système d'exploitation 64 bits Windows 7 professionnel

2.4. Moteurs de recherche utilisée

Nous avons utilisé le moteur de recherche Lucene afin de retrouver des images pertinentes aux requêtes.

Lucene est un moteurs de recherche d'où il utilise des techniques et des algorithmes pour donner la liste des documents pertinent par rapport à une requête donnée., aussi ce dernier est programmer par Java, la version utiliser est la dernière (6.5.1), elle est créé le 27 avril 2017 par la fondation apache .

La figure suivante représente Interface d'un programme java de moteur de recherche lucene :

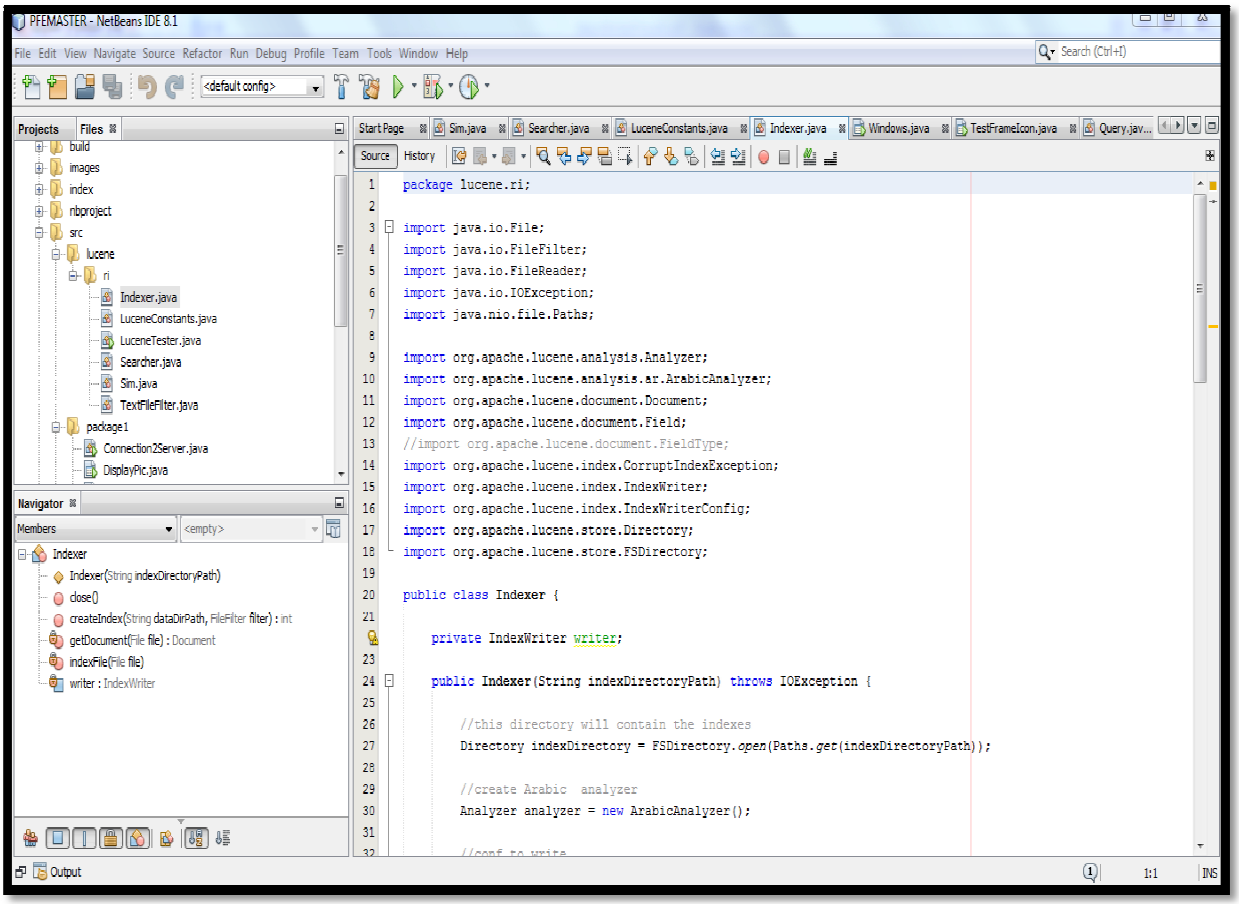


FIGURE 4.3 : Interface d'un programme java de moteur de recherche lucene

3. Présentation notre travail

Dans cette section, nous présentons et expliquons les différentes phases de notre travail qui commencent par traiter notre base d'images, puis à présenter notre travail dans notre application.

3.1. Data set

L'ensemble de données IAPR-TC12 de 20 000 images contient des descriptions de texte libre de chaque image en anglais, allemand et espagnol. Elles sont divisées en «titre», "description", "lien de l'image", "notes",…….,etc.

Nous avons créé une base de données sous forme Excel (fichier csv), le tableau suivant montre les types d'informations récupérées.

	B	C	D	E	F	G	H	I
1	DOCNO	TITLE	DESCRIPTION	NOTES	LOCATION	DATE	IMAGE	THUMBNAIL
2	annotations	Excursion	About 20		Quilotoa,		images/00/	thumbnails
3	annotations	Termas de	a rock hole		Papallacta,		images/00/	thumbnails
4	annotations	The Plaza	a yellow	The Plaza	Cochabamb	1 February	images/00/	thumbnails
5	annotations	Plaza de	a large		Cochabamb	1 February	images/00/	thumbnails
6	annotations	Statue of	a white,	People in	Cochabamb	February	images/00/	thumbnails
7	annotations	Aerial	View of a		Cochabamb		images/00/	thumbnails
8	annotations	Panoramic	a cascading	this picture	Foz do Igua		images/00/	thumbnails
9	annotations	View of the	a man in a	Original	Foz do Igua	February	images/00/	thumbnails
10	annotations	The Iguazu	View of	This view is			images/00/	thumbnails
11	annotations	The Itaipu	a long dam	One of the	Foz do Igua	February	images/00/	thumbnails

Figure 4.4: Les différentes informations récupérées pour chaque image en anglais

La figure suivante représente le code source qui fait crée data set sous forme d'un fichier CSV.

```
path="E:\\application\\iaprtc12\\annotations_complete_eng\\"# schema
def getAll(path):#avoir tout
    files = []   #liste
    # r=root, d=directories, f = files
    for r, d, f in os.walk(path):
        for file in f:
            if file.endswith(".eng") :#or file.endswith(".ger") or file.endswith(".rnd")or file.endswith(".spa"):
                files.append(os.path.join(r, file))#ajouter
    return files
import re

TAG_RE = re.compile(r'<[^>]+>')

def remove_tags(text):
    text=TAG_RE.sub('', text)
    text=re.sub(r'[^\x00-\x7F]+',' ', text)

    return text
def filetotext(path):
    ch=[]

    f = open(path,"r")
# ================================================================================
    for x in f:

        #         #print(x)
        x=remove_tags(x)
        if len(x)!=0:
            ch.append(x)

    return ch
ll=[]
out="C:\\Users\\Admin\\Desktop\\DATASET2.csv"
# ================================================================================
# import pandas as pd
for d  in    getAll(path):
     s=filetotext(d)
     if len(s)!=0:
         ll.append(s)
import csv
with open(out,"w", newline='') as f:
    writer = csv.writer(f, delimiter=";")
    writer.writerow(["num", "DOCNO", "TITLE","DESCRIPTION","NOTES","LOCATION","DATE","IMAGE","THUMBNAIL","DOC"])
    writer.writerows(ll)
```

Figure4.5: Code source python qui fait la création de fichier csv.

3.2. Présentation de l'application

L'utilisateur doit entrer une requête pour laquelle il veut avoir un résultat. Nous avons appliqué un prétraitement (tokenisation, élimination des mots vides, normalisation) sur la requête initial pour avoir une requête reformuler.

Nous avons crée une application java qui nous permet d'introduire la requête, puis nous pouvons voir la requête reformuler et la requête enrichis.

Dans la suite nous pouvons lancer la recherche afin de retrouver les images pertinentes à la requête et aussi choisir une requête simple ou bien enrichir pour la recherche.

Cette application est présentée dans la figure suivante.

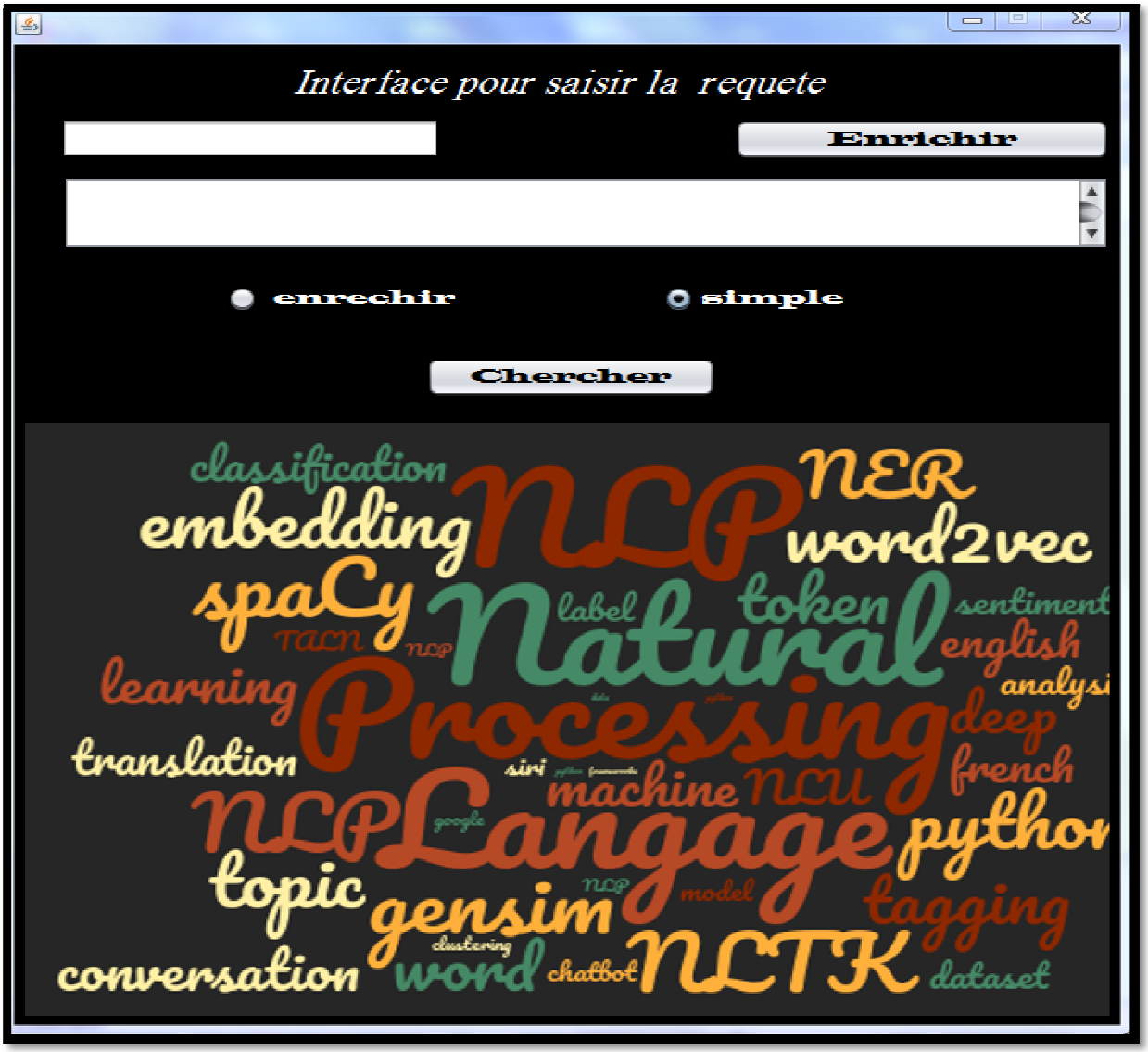


Figure 4.6: l'interface de l'application.

Nous montrons notre application à travers un exemple comme suit :

La figure suivante montre comment entrer la requête.

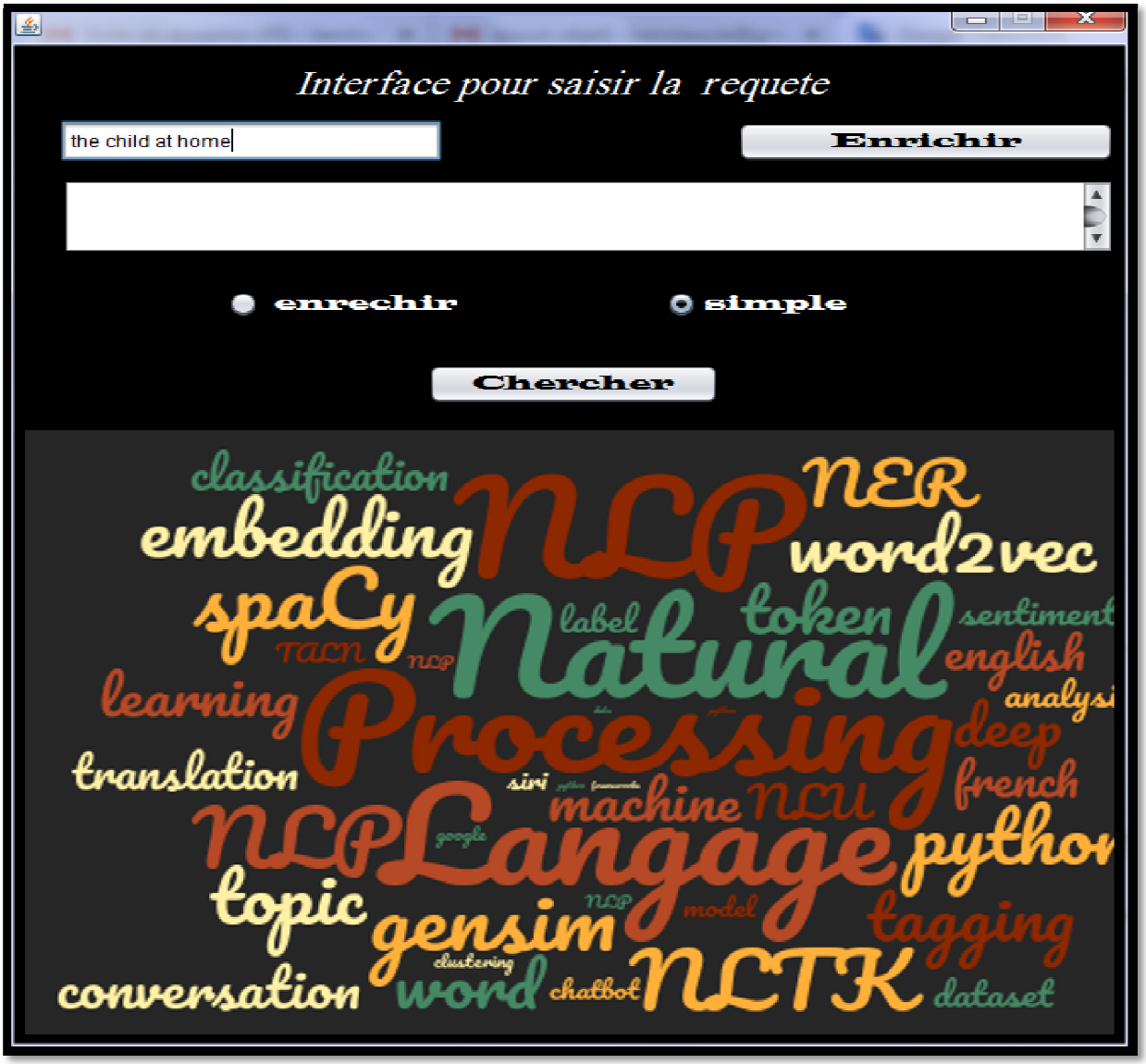


Figure 4.7: pour saisir la requête

Puis cliquer sur le bouton enrichir pour afficher la requête enrichir

Figure 4.8: enrichir la requête

Les figures suivantes expliquent les résultats d'une recherche, d'où l'application affiche les 6 premières images pertinentes à la requête introduit.

Cette figure représente les images de la requête initiale

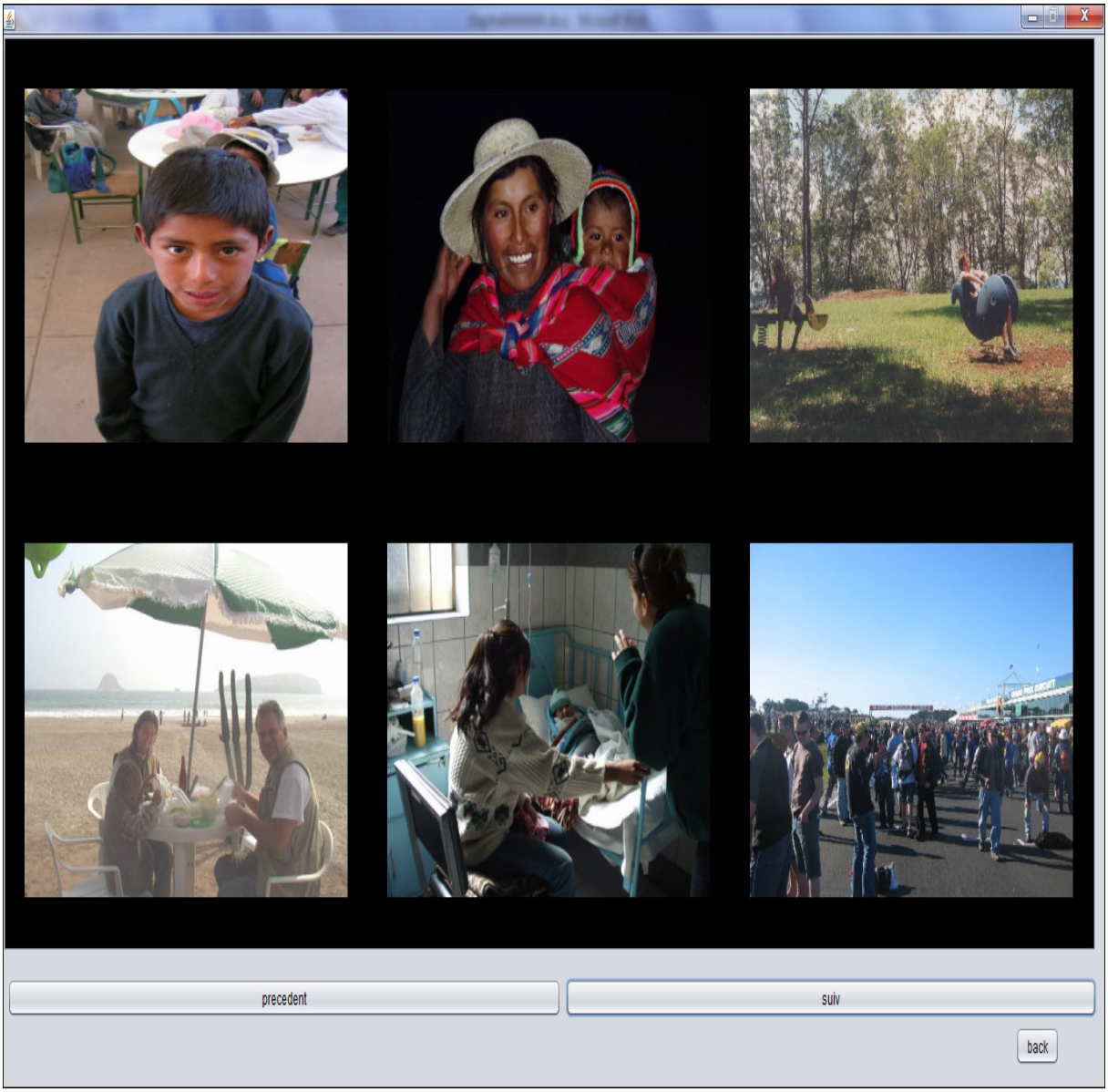


Figure 4.9:les images retournées par la requete initiale

Et cette photo représente le résultat de recherche par requête enrichir.

Figure4.10 :les images retournées par la requete enrichir

4. Evaluation :

Nous avons calculé le rappel pour chaque requête, a fin de faire une comparaison entre les différentes requêtes, pour trouver l'évaluation de notre projet, de là on peut calculer la moyenne entre le rappel des requêtes, afin de faire des comparaisons entre la requête initiale et la requête enrichir.

L'objectif de cette expérimentation est de comparer notre approche entre la requete enrichir et la requête initiale.

- Le scénario d'évaluation le plus courant est le suivant :

Nous avons créé 10 requêtes, puis enrichi toutes les requêtes créées. Chacune de ces requêtes a été entrée dans le moteur de recherche, puis évaluée à l'aide de mesures de pertinence.

Une fois les résultats obtenus, nous avons évaluent en leur attribuant des scores selon leur pertinence par rapport à la requête.

Les scores des différentes requêtes sont ensuite combinés, dans une sorte de moyenne pondérée par exemple, ce qui permet d'obtenir différents indicateurs dont le rappel.

Le Rappel (Re) mesure la proportion de bonnes images retournées à l'utilisateur parmi toutes les bonnes images dans la base d'images.

Une fois calculé le rappel peuvent être représentés avec un diagramme pour évaluation.

Signe	Requête initiale	Requête enrichir
R1	the child at home	children girl boy baby man woman pan boys kid shop front next foreground behind blue around brown stretch peoples tumbaco racecar abeac today wildly godmother kil harry
R2	the jungle is great	orest ohio schoolyard rainforest excited vegetation coppice huemul nowhere coniferous another man person woman black next people brown panecillo tatanga shihlin sangay busan coit turkey condition tip sabancaya
R3	the sky is blue	background clouds dark behind range high trees another man person woman black next people brown grey brown white black green red dark light background
R4	the fire	Cabiets fireplace stiring canada campfire hangers extinguisher cupboard sink shelves
R5	the flowers	blooms plants green dark hedges flower lamps palms blooming
R6	passengers from the bus	hostesses chess hoisted navimag montt activities dany magallanes christians ranger versatile reeds navegating tunel aerol gliding neas got jvc aerosur scraping batsman comfortable goggles cockpit chunks mudbath skier minutes minibus train jeep truck van minivan airplane campervan buggy car
R7	the ship in the sea	boat ships deck boats freighter docked lookout yacht jetty speedboat bay beach water port lake river coast breaking chaos jetty
R8	the audience in the stadium	recognized sandhills bodywork turning italians lizards garbag chairlifts damage srubs grandstands grandstand players floodlight scoreboard hard medal occupied campaign qualifying
R9	the car	winches minibus meter beetle fountains truck van basis aater bicycle
R10	the Bicycle	bike car shoulder horse tricycle bicycles racebike bikes meter

Tableau 4.1 :les requêtes initial et enrichir pour le test

- **Calcule le rappel**

Après avoir créé et enrichi les requêtes, nous avons calculé les rappels pour chaque requête enrichir et chaque requête initiale.

Nous avons utilisé la formule ci-dessous pour calculer le rappel.

$$rappel = \frac{Nombre\ de\ documents\ pertinents\ séléctionnés}{Nombre\ total\ de\ documents\ pertinents}$$

Les résultats obtenus sont illustrés dans le tableau suivant:

Requêtes	Rappel de la requête Initiale	Rappel de la requête Enrichir
R1	72 %	94 %
R2	61 %	77 %
R3	38 %	61 %
R4	50 %	61 %
R5	61 %	66 %
R6	66 %	80 %
R7	83 %	88 %
R8	55 %	72 %
R9	72 %	80 %
R10	61 %	88 %

Tableau 4.2: Comparaison entre la requête initiale et la requête enrichir

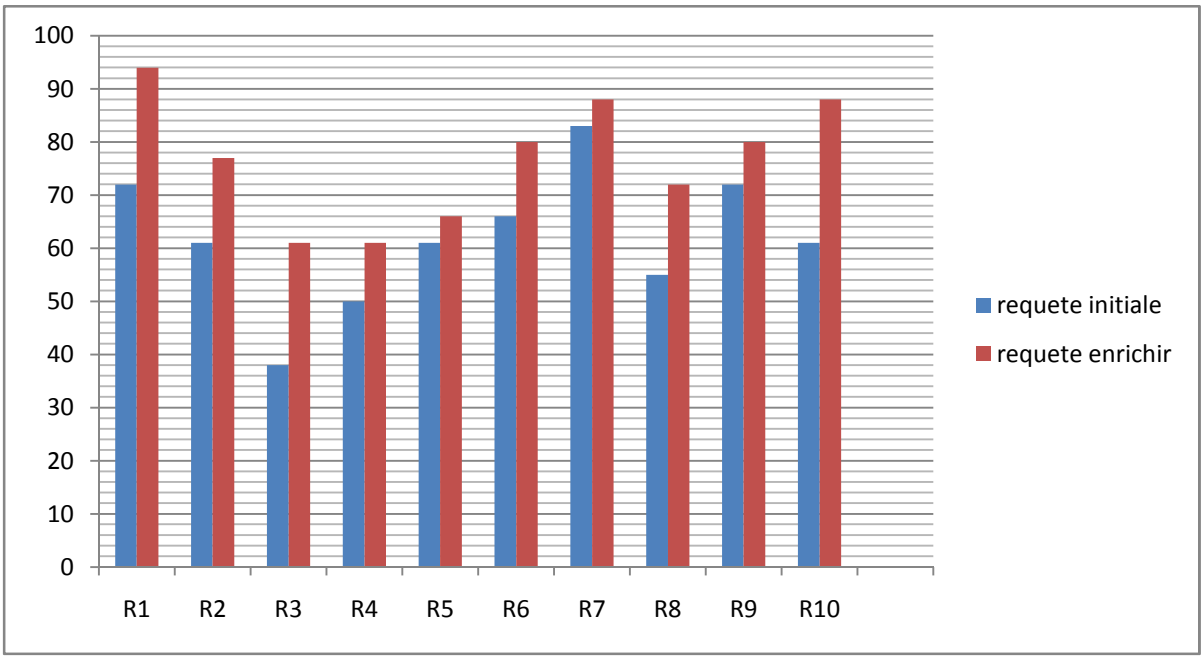


Figure 4.11 : histogramme de comparaison entre les résultats de la requête initiale et de la requête enrichir

Nous remarquons que le rappel est augmenté d'une façon considérable avec notre approche. Nous justifions ces résultats par le fait que l'utilisation de word2vec (skip-gram) permet de des nouveaux concepts cachés au plus des concepts explicites.

Cela montre que notre projet a obtenu de bons résultats.

5. Conclusion

Au cours de ce chapitre, nous avons présenté l'environnement de développement afin de mettre on œuvre notre approche. Dans la suite, nous présentons les différentes bibliothèques, moteur de recherche et le langage utiliser pour crée notre application.

A la fin, nous avons comparé les résultats obtenus pour les requêtes initiales et les requêtes enrichies en calculant la valeur de rappel pour évaluer notre approche.

Conclusion générale

Conclusion générale

Nous avons présenté dans ce mémoire notre travail qui s'inscrive dans la recherche d'image. Nous avons proposé l'enrichissement de la requête en utilisant word2vec pour la recherche d'image.

Word2Vec est un modèle de Word embedding est parmi les plus connus. Il a été développé par une équipe de recherche de Google sous la direction de Tomas Mikolov.

Cette approche joue un rôle important dans le développement de notre travail parce que word2vec gère les textes et est facile à appliquer en plus de donner des résultats impressionnants, il donne à chaque mot plusieurs mots ayant le même sens ou une même convergence de sens. et notre objectif est enrichir les requêtes pour d'obtenir des résultats satisfaisants Quand vous rechercher des images.

Nos travaux ont été évalués au moyen d'un ensemble d'images illustrées et développement une application en java pour la recherche.

Et après les tests et l'évaluation, nous avons constaté que l'enrichissement de la requête donnait des résultats positifs par rapport à la requête initiale lors de la recherche d'images, alors le travail que nous avons réalisé était très enrichissant.

Et malgré les difficultés rencontrées au début, nous avons pu progresser et nous avons appris beaucoup de choses au cours de la période de travail, telles que le développement et l'expansion d'idées, et nous sommes certains que le monde de l informatique est riche et vaste.

En dernier lieu, nous souhaitons bien améliorer ce travail dans les prochains jours afin de faciliter le processus de recherche pour de nombreuses personnes.

Références Webographies

[1]: http://www.map.toulouse.archi.fr/works/panoformation/imagenum/imagenum.htm

[2] :https://www.memoireonline.com/05/14/8871/La-recherche-d-images-par-la semantique11

[3] : https://fr.sciencewal.com/28496-what-the-heck-is-word-embedding-b30f67f01c81-86

[4] : https://www.researchgate.net/figure/Architecture-des-deux-modeles-sac-de-mots continu-Continuous-Bag-Of-Words-CBOW-et_fig13_323543226

[5] : https://www.synthesisproject.org/resources

[6] : https://scoms.hypotheses.org/657.

[7] : https://www.tresfacile.net/introduction-au-langage

[8] : https://python-guide-pt-br.readthedocs.io/fr/latest/scenarios/scientific.html

[9] : https://riptutorial.com/fr/nltk/topic/4077/demarrer-avec-nltk

[10] : https://www.futura-sciences.com/tech/definitions/internet-java-485/

Références bibliographies

[Gruber,1993]: T. R. Gruber, «A Translation Approach to Portable Ontology Specifications,» Stanford University, Stanford,California , 1993.

[Christophe,2008]: Christophe M « Annotation automatique d'images : Annotation cohérente et création automatique d'une base d'apprentissage », Thèse de Doctorat, Paris, Soutenu le 14/01/2008.

[Ben Cheikh,2011] : Ben Cheikh. Ben Bezziane : La recherche d'image par la sémantique, mémoire en vue de l'obtention du diplôme de Master Académique, univ kasdi merbah Ouargla 06 /2011.

[Radia Abdi,2012] : Radia Abdi, intégration d'une application d'indexation dans un environnement Cloud, mémoire de master,2012.

[BEDOUHENE]: Saida BEDOUHENE, Recherche D'images Par Le Contenu, Thèse de doctorat, Université Mouloud Maameri, Tizi-Ouzou

[L.T.LAN, 2005]: L.T.LAN (indexation de recherche d'image par le contenu) institut de polytechnique de hanoi ,2005.

[SALTON] : SALTON, Gerard et MCGILL, Michael J. *Introduction to modern information retrieval.* New York: Mcgraw-Hill Book Company, 1983. 448 p. (Computer Science)
[Tanjona,2016] : Mr Tanjona Michael Rasolomanana, Représentation encapsulée de mots pour l'extraction Automatique de mots clés, Université D'ANTANANARIVO, Master 2 en Informatique, 2016. .

[Jean-Charles, 2017] : Jean-Charles RISCH, Enrichissement des Modèles de Classification de Textes Représentés par des Concepts, UNIVERSITÉ DE REIMS CHAMPAGNE–ARDENNE, en informatique, 2017.

[**Ghannay, 2017**] : Sahar Ghannay, Etude sur les representations continues de mots appliquees à la detection automatique des erreurs de reconnaissance de la parole Université du Maine, 2017