

# Rapport projet Python

Talkative Bot  
Génération textuelle par apprentissage profond



## Propri t s du document

Classification	Interne -	
Version	Version 1.0	
Auteurs	Duc-Vinh TRAN Samy BOUCHNAIF	<a href="mailto:duc.vinh.tran@ensae-paristech.fr">duc.vinh.tran@ensae-paristech.fr</a> <a href="mailto:samy.bouchnaif@ensae-paristech.fr">samy.bouchnaif@ensae-paristech.fr</a>
Responsable	Romain LESAUVAGE	<a href="mailto:romain.lesauvage@ensae-paristech.fr">romain.lesauvage@ensae-paristech.fr</a>
Encadrants	Daniel DE PAULA E SILVA	<a href="mailto:danielpes91@gmail.com">danielpes91@gmail.com</a>
Pages	23	

## Note de r daction :

Ce rapport a  t  r dig  dans le cadre du projet Python du second semestre pour le Mast re Sp cialis  en Data Science. Ce document constitue la synth se du travail r alis  et sert  galement de support de compr hension pour le livrable fourni   l'adresse suivante : <https://github.com/dvp-tran/LSTM>.

La page Twitter du bot est la suivante : <https://twitter.com/TheTalkativeBot>, n'h sitez pas   nous demander de mettre en ligne le bot afin de le tester (je l'interrompe d s que possible).

Nous nous effor ons d'expliquer le choix de notre sujet et l'int r t de notre probl matique, de d tailler notre plan de travail et de parler des briques technologiques manipul es. Enfin, nous faisons un bilan de notre projet, nous parlons des probl mes rencontr s, et les am liorations envisageables.

Nous restons   votre disposition pour de plus amples informations, ainsi que pour d' ventuelles remarques et suggestions qui pourraient notamment nous permettre de rectifier et compl ter notre travail. Nous souhaitons par la pr sente remercier Monsieur LESAUVAGE pour la coordination g n rale du module, ainsi que Monsieur DE PAULA E SILVA pour son encadrement tout au long du projet. Son regard, ses conseils, ses avis, et sa grande disponibilit  ont contribu    la r ussite de ce projet.

## Pr cautions :

- Le projet a  t  d velopp  et test  sur les configurations suivantes :
  - Ubuntu 16.04 LTS et Ubuntu 17.04
  - Python 2.7
  - CUDA  Toolkit 8.0
  - NVIDIA drivers 375.xx
  - cuDNN v5.1
  - TensorFlow (CPU & GPU) v.1.1
  - Keras v.2.0.3
  - Opencv v.3.2
- Une liste plus pr cise des librairies utilis es et des pr -requis logiciels est fournie dans le fichier README.md   la racine du *repository*.

## Sommaire

Motivations & résumé :	4
Introduction :	4
1. Acquisition de la donnée :	5
1.1. Le web scraping :	5
1.2. Requête sur serveurs FTP.....	6
2.1. Contexte du LSTM :	8
2.2. Cadre théorique autour des LSTM :	9
2.3. Illustration du fonctionnement d'un LSTM :	11
3. Apprentissage d'une image par réseau pré-entraîné :	13
3.1. Mise en place du contexte :	13
3.2. Fonctionnement du modèle :	14
4. Interfaçage du bot avec Twitter .....	15
4.1. Twitter API.....	15
4.2. Interfacer tous les modules.....	15
4.3. Composition de la réponse :	17
5. Outils de travaux :	19
5.1. Google Cloud :	19
5.2. Outils logiciels :	20
Conclusion :	21
Bibliographie :	22
Annexe :	22

## Motivations & résumé :

Dans le cadre du projet Python pour Data Scientist du second semestre de notre cursus en Mastère spécialisé, nous avons décidé d'élaborer une intelligence artificielle capable de s'exprimer à l'écrit, et ce dans plusieurs langues. L'objectif étant de s'intéresser à des problématiques de *deep learning* de pointe telles que :

- L'image captionning : description automatique du contenu des images
- La génération de texte automatique.

Par ailleurs, au-delà des considérations en *machine learning*, ce projet constitue également une opportunité pour se familiariser et appréhender un panel d'outils varié afin de monter en compétences en informatique sur des domaines qui n'ont pas été nécessairement abordés en cours.

## Introduction :

Imaginez une intelligence artificielle capable de vous générer un scénario de film ou un roman du genre de votre choix. Plutôt drama ? Comédie ? Fantastique ? Encore inimaginable il y a quelques années, le *machine learning* rend désormais ceci possible. Grâce à une bibliothèque riche en films et ouvrages, et un ensemble d'algorithmes, un ordinateur est capable de générer des scénarios et ouvrages inédits. Le *machine learning*, et plus particulièrement le *deep learning*, se met donc au service de l'art. L'imitation du fonctionnement inductif du cerveau humain est envisageable par l'utilisation d'un réseau de neurones - ou quand l'apprentissage profond est capable de vous écrire un scénario de court-métrage (presque) sur mesure !

À travers ce rapport, nous allons expliquer comment nous avons récupéré les scénarios de films et les ouvrages sur Internet (web scraping). Nous allons également en dire plus sur le type de réseau de neurones utilisé pour s'entraîner sur les scripts et ouvrages dans l'optique d'en générer de nouveaux (réseau de neurones récurrent de type LSTM : Long Short-Term Memory). Enfin, nous parlerons du côté applicatif de notre projet, avec le *bot Twitter* qui génère un script à la demande une fois que nous lui fournissons une image de notre choix (un réseau pré-entraîné a été intégré au bot afin d'associer un texte en sortie à l'image que nous fournissons au bot ; texte toutefois toujours généré par notre LSTM).

De nombreux choix s'offraient à nous en ce qui concerne les modules et choix technologiques pour ce projet. Nous avons codé certains modules de notre programme (du scraper au bot, en passant par le LSTM) en Python, avec pour interface le traditionnel Notebook afin de visualiser les sorties et avoir une interface didactique notamment concernant les parties d'apprentissages des réseaux de neurones. En effet, les Notebooks permettent aux lecteurs une compréhension plus facile des différents modules. Néanmoins le bot dans son intégrité a été développé directement sous forme de scripts Python exécutables.

Le livrable : un bot twitter capable de réagir et raconter une histoire face à une image. Ce bot fonctionne de la façon suivante :

- Suite à un post d'une image sur son flux d'actualité, le bot récupère l'image
- Le bot reconnaît ensuite le contenu de l'image
- Le bot génère du texte (histoire, poésie, etc...) concernant l'image
- Le bot répond au post avec le contenu généré.

Il se décompose en plusieurs modules :

- Le module d'acquisition de données (ou scraper)
- Le module d'apprentissage textuel : LSTM
- Le module d'apprentissage pour la description d'image : *Image Captionning*
- Le module d'interaction avec Twitter

# 1. Acquisition de la donn e :

Dans cette partie, nous d crirons comment nous obtenons de mani re automatique les donn es textuelles servant de base d'apprentissage   notre algorithme de g n ration de textes. Il est donc possible d'obtenir sans effort un volume cons quent de donn es bien pr cises.

Compte tenu de nos motivations de d part, nous nous sommes int ress s   deux sources de donn es :

- IMSDB <sup>1</sup>: site r f ren ant les scripts des films cin matographiques am ricains sous format html.
- Projet Gutenberg <sup>2</sup> : site qui compile les  uvres de litt rature libres de droit sous format ebooks.

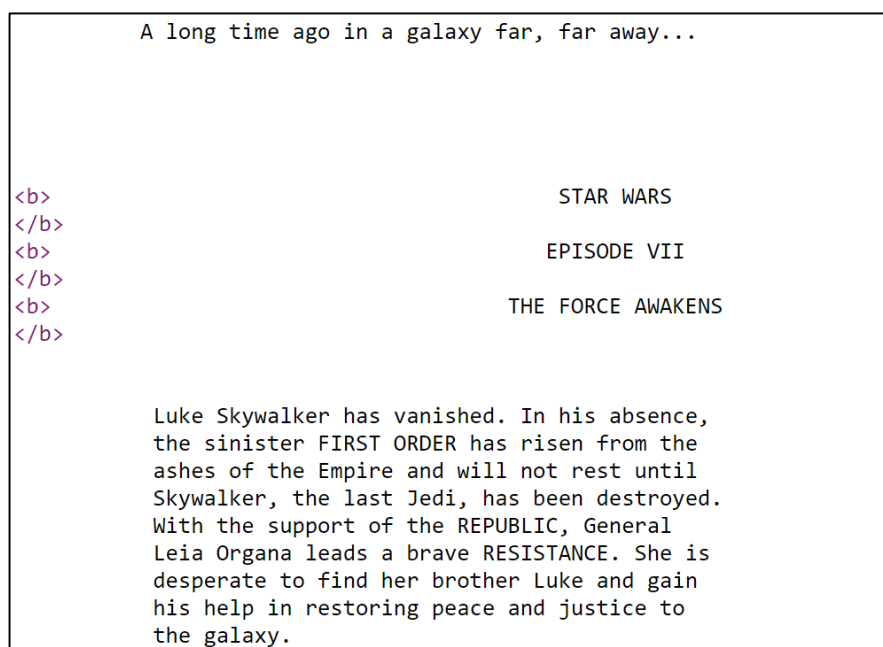
Ainsi il est possible   la fois d'apprendre   notre algorithme de g n rer des sc narios de films avec r pliques ou bien r diger un roman dans une langue quelconque.

## 1.1. Le web scraping :

Il s'agit d'un script qui parcourt un domaine de site web et r cup re des informations pr cises afin de les stocker et les exploiter ult rieurement. La m thodologie est simple :

- Une url servant de point d'entr e
- A partir de cette url, l'automate analyse le contenu de la page :
  - Rep re les liens int ressants (d fini par une r gle)
  - T l charge sur disque les donn es pertinentes

Dans le cas d'IMSDB : le script prend en entr e le genre cin matographique cibl e (ex : Action, Romance, etc.) et va ensuite se propager sur chaque lien de films concern s pour en t l charger le contenu -i.e. le script sous format html.



**Fig. 1 : Un exemple du format de script pour Star Wars ep. VII**

<sup>1</sup> <http://www.imsdb.com/>

<sup>2</sup> <https://www.gutenberg.org/>

Les fichiers sont ensuite simplement stock s sous format textes sur le disque. Dans une optique d'industrialisation de la d marche, il aurait  t  int ressant de les indexer dans une base de donn es orient e documents. N anmoins par soucis de simplicit  et manque de temps, nous avons d cid s de stocker simplement les fichiers dans un r pertoire par genre cin matographique.

A ensuite  t   labor  un parseur pour d composer chaque script sous un format plus riche en information que du texte brut, nous avons cherch    extraire :

- Les indications sc niques (didascalies et lieux),
- Identifier les personnages du texte,
- Identifier les r pliques par personnages.

En effet nous pensions   l'origine il  tait n cessaire pour l'algorithme de g n ration de texte de lui pr ciser ce qu' tait une r plique et de le diff rencier d'une indication sc nique. Cependant nous verrons plus tard que gr ce   la th orie des LSTM qui conservent la structure du texte que cela n' tait pas n cessaire.

Remarque : cette pratique est   r aliser avec pr caution, en effet un requ tage intempestif sur un serveur peut nuire   la qualit  de service de ce dernier et le rendre inaccessible pour autrui pouvant alors s'apparenter   un acte de malveillance tels que les attaques par DDOS (d ni de service). Dans le cas le plus probable, un requ tage intensif entra nera le serveur distant   bloquer votre adresse I.P. pour emp cher de nuire.

La bonne pratique est de se limiter   la donn e qui nous int resse sans superflu et de limiter la vitesse/quantit  de requ tes envoy es   la seconde (par exemple via une commande sleep).

## 1.2. Requ tage sur serveurs FTP

Concernant le site du projet Gutenberg, la m thodologie d'acquisition des donn es est diff rente : en effet, le site n'autorise plus le scraping depuis mars 2016 et bloque toutes tentatives. Cependant, des contributeurs ont mis en place des serveurs FTP (File Transfer Protocol mirrors) par r gion g ographique pour t l charger librement.

Dans un premier temps, nous avons pens  que cela  tait beaucoup plus pratique pour t l charger des textes pr cis, mais tr s vite nous avons r alis  que l'indexation des fichiers sur leurs serveurs  tait r alis e de fa on sous optimale :

Morsiuspuku, by Alexandre Dumas, p��re [Language: Finnish]	54592
An English Girl in Japan, by Ella M. Hart Bennett	54591
Talk of Uncle George to his Nephew about Draw Poker, by Uncle George	54590
The American Missionary, Volume 33, No. 10, October, 1879, by Various	54589
Bela, by Mikhail Iurevich Lermontov [Subtitle: Kaukaasialainen kertomus] [Language: Finnish]	54588
The Pope, the Kings and the People, by William Arthur [Subtitle: A History of the Movement to Make the Pope Governor of the World by a Universal Reconstruction of Society from the Issue of the Syllabus to the Close of the Vatican Council]	54587
Die Ostereyer, by Christoph von Schmid	54586

**Fig. 2 : Illustration de l'index Gutenberg, entour  : probl me d'encodage**

Comme il est possible de le constater, la structure d'indexation

Titre – by Auteur – N° Fichier

n'est pas tout le temps respectée, rendant le parseur peu efficace aux variations de nomenclature des fichiers, par exemple l'auteur William Shakespeare peut se retrouver sous plusieurs formes : 'W. Shakespeare' ou 'William Sh.'. Néanmoins les balises tels que l'indicateur de [Langue] sont invariantes. C'est pourquoi nous avons créé un parseur capable de rechercher le numéro de fichier pour une langue particulière plutôt que pour un auteur particulier.

L'inconvénient du système FTP est que les ouvrages ne sont pas répertoriés par genre littéraire contrairement sur le site web. On a donc une perte d'information, d'où le choix de se limiter au paramètre de la langue plutôt que par genre ou par auteur.

Le fonctionnement du script est donc le suivant :

- En entrée on fournit l'adresse du FTP, la langue souhaitée et le nombre de documents à télécharger (pour éviter de saturer le disque).
- Le script télécharge et parse le fichier d'index du serveur pour trouver les documents souhaités.
- Une fois les documents téléchargés, ces derniers sont dézippés.

Remarque : comme entouré sur la figure 2 plus haut, il faut s'attendre à de nombreux problèmes d'encodage selon les langues (UTF8, LATIN1, ISO-8859-1, etc...) qui vont causer des soucis pour le LSTM.

## 2. LSTM, réseau de neurones récurrent à l'architecture complexe :

Cette partie constitue le cœur de notre projet.

### 2.1. Contexte du LSTM :

Un réseau de neurones de type LSTM est une technologie utilisée sur les claviers pour « prédire » ce que vous allez écrire par la suite (utilisateurs d'iPhone, entre autres). Ce genre de réseau a déjà fait ses preuves, notamment à l'université de New York où Ross Goodwin, chercheur en intelligence artificielle dont la spécialisation est le langage et les réseaux neuronaux, en a mis un en œuvre. Il a fait entraîner Benjamin, son LSTM, en lui faisant apprendre des scénarios de science-fiction. Des films et séries comme 12 Monkeys, Blade, Blade Runner, Star Trek, le cinquième élément, Terminator, Star Wars, Jurassic Park ont été projetés sur des écrans avec les dialogues associés au format texte.

Au cours de sa phase de formation, Benjamin est allé jusqu'au niveau des lettres et a appris à prédire quelles lettres ont tendance à se suivre les unes après les autres au regard de ce qu'il lit dans les scripts qu'il apprend, et ensuite quels mots et phrases étaient souvent vus ensemble. L'avantage que représente un algorithme LSTM face à une chaîne de Markov est qu'il peut faire des échantillons de plus longues chaînes de caractères. Ainsi, il sera meilleur à la prédiction de paragraphes entiers au lieu d'une prédiction qui se limite à quelques mots. Il présente également l'avantage de générer des phrases originales au lieu de couper et coller des phrases qu'il a eu à rencontrer lors de son parcours.

Après cet exercice, Benjamin a appris à imiter la structure d'un scénario, produisant des indications scéniques en définissant bien les dialogues de ses personnages. Benjamin n'a cependant pas été en mesure de bien définir les noms de ses personnages étant donné qu'ils ne sont pas utilisés comme les autres mots et qu'ils gardent donc un caractère imprévisible. Goodwin a dû se résoudre à modifier tous les noms des personnages définis dans le scénario de Benjamin. C'est pour ça que dans ce scénario, les personnages de Sunspring sont nommés H, H2 et C. En réalité, dans le scénario original, deux personnages distincts étaient nommés H, ce qui était troublant pour les acteurs. L'un d'eux a été nommé H2 pour plus de clarté. Tandis que l'équipe se réunissait autour de l'imprimante, Benjamin a lancé l'impression du scénario avec des indications un peu farfelues comme « il est debout sur les étoiles et assis sur le sol ». Par la suite Oscar Sharp a distribué les rôles de façon aléatoire aux acteurs présents dans la pièce. « Dès que nous avons parcouru suffisamment le scénario, tout le monde autour de la table s'est mis à rire de bon cœur », a confié Sharp.

Ce n'est pas la première fois que des intelligences artificielles se sont essayées à jouer les humains dans le domaine de l'art. Il faut par exemple citer le projet Magenta de Google, qui vise à créer des pièces d'art originales à travers un mélange de reconnaissance vocale, de traduction et de reconnaissance d'images. Magenta envisage d'apprendre des autres comment créer de l'art pour créer des œuvres « convaincantes ». Deep Dream, un projet de vision par ordinateur de Google, se sert d'un réseau neuronal convolutif pour trouver et renforcer des structures dans des images en utilisant des paréidolies créées par algorithme, donnant ainsi une apparence hallucinogène à ces images. En clair, ce programme de Google permet de faire apparaître des formes dans les images comme un cochon volant dans le ciel ou un chat apparaissant derrière un lampadaire. Un générateur en ligne est disponible pour le public et le code source est disponible sur GitHub.

Bien que Benjamin, Magenta et Deep Dream soient des projets impressionnants, il est peu probable qu'ils puissent prendre le pas sur les humains dans l'industrie artistique très bientôt.

Vous avez donc compris tout l'intérêt des LSTM. Dans notre projet, nous avons essayé de reprendre ces idées en créant une intelligence artificielle capable de générer de nouveaux films et ouvrages selon le genre de votre choix.



## 2.2. Cadre th orique autour des LSTM :

Les LSTM sont des cas particuliers de r seaux de neurones r currents. Vous n'avez peut- tre jamais entendu parler de tels r seaux. Un r seau de neurones peut r soudre presque tous les probl mes d'apprentissage automatique, quelle que soit sa complexit .

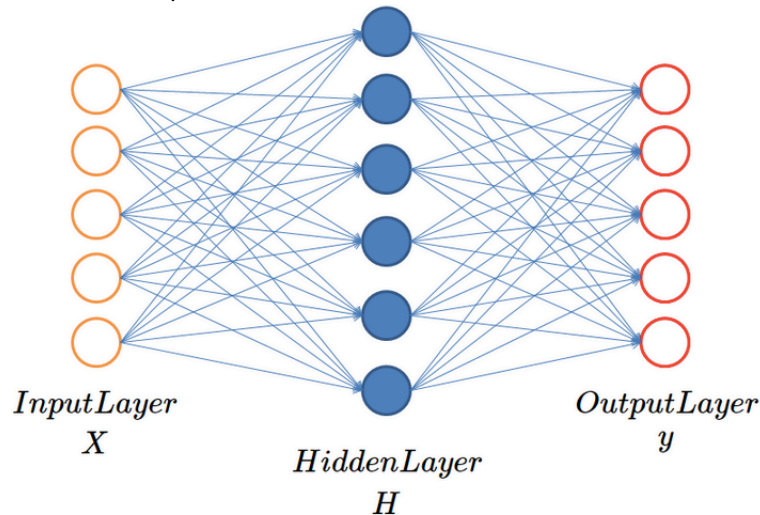


Fig. 3 : Illustration simple d'un r seau de neurones (source : [chunml.github.io](https://github.com/chunml))

Comme vous pouvez le voir dans l'image ci-dessus, un r seau de neurones peut surpasser d'autres algorithmes d'apprentissage en raison de ses couches cach es, qui permettent de cr er un ensemble de fonctionnalit s plus compliqu es – et donc une meilleure pr cision dans la sortie pr dite. Mais malgr  les performances remarquables que les r seaux de neurones ont montr es au cours de la derni re d cennie, leur grande faiblesse r side dans l'inintelligibilit  des s quences qu'il apprend (qui, par ailleurs, sont influenc es par les s quences les pr c dant). L'ajout d'une caract ristique de r currence dans de tels r seaux va permettre de pallier ce probl me.

Commen ons par introduire un RNN (r seau de neurones r current) r gulier, autrement dit avec un *feed-forward* dot  d'une seule couche cach e :

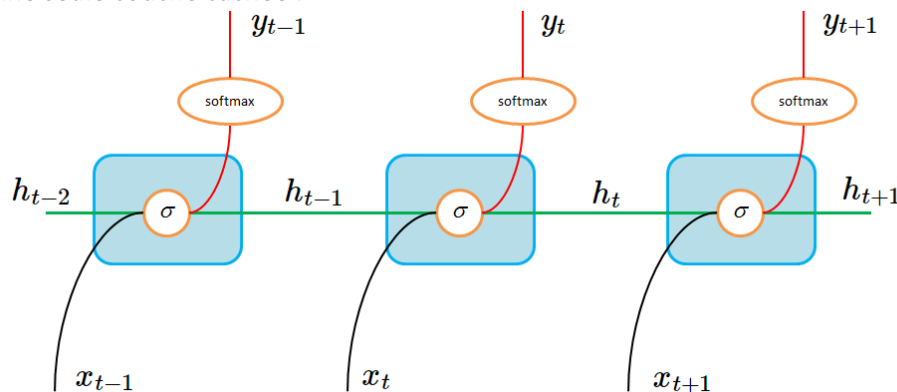


Fig.4 : RNN r gulier dit Vanilla RNN (source : [chunml.github.io](https://github.com/chunml))

Son id e de base consiste   empiler une ou plusieurs couches cach es de pr c dents instants ; chaque couche cach e d pendant de l'entr e associ e   l'instant en cours et   l'instant pr c dent. Nous avons :

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1})$$

De son c t , la sortie est calcul e en utilisant uniquement la couche cach e associ e :

$$y_t = \text{softmax}(W_{hy}h_t)$$

Ainsi, avec des couches cachées associées à des instants différents, le nouveau type de réseau de neurones (le RNN en l'occurrence) a la capacité de se souvenir des entrées apprises au fil du temps. Cependant, il ne peut pas se souvenir d'un long *timestep* (autrement dit, un instant qui remonte à loin). Ceci est due à un problème appelé « gradient de disparition ». De plus, le LSTM n'a pas la capacité de savoir quelles informations d'un instant doit-il garder, et il ne peut pas savoir quelles informations d'un *timestep* sont ou non à préserver. Donc, une amélioration était nécessaire, et elle est matérialisée par l'avènement du LSTM.

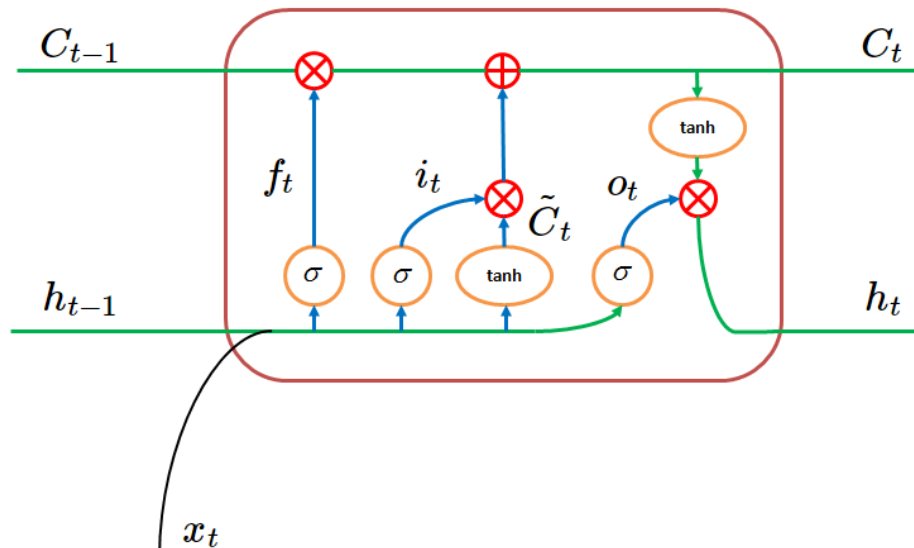


Fig.5 : Architecture d'un réseau LSTM (source : [chunml.github.io](https://github.com/chunml))

Vous l'aurez compris, l'architecture RNN classique vu en introduction ne suffit donc plus, et nous passons aux LSTM. Expliquer le fonctionnement intégral d'un LSTM serait un peu fastidieux et n'a pas sa place dans ce rapport. Toutefois, vous verrez que les fonctions évoquées sur cette représentation remplissent des opérations traditionnelles (application de poids  $W$  et ajout de biais, fonction d'activation, etc). Pour information, nous avons comme fonction de calcul de l'état caché à l'instant  $t$  (calcul de l'état caché au *timestep*  $t$ ) :  $o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$ .

La fonction suivante peut être vue comme une « porte de sortie ». En effet, vu le chemin qu'elle offre et qui permet de « s'échapper » de la couche cachée en allant directement vers la fonction d'activation, elle offre au LSTM la possibilité d'oublier le précédent état caché :  $f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$ .

Bien sûr, cette fonction n'existerait pas sans son nemesis, à savoir la fonction qui dit au LSTM s'il doit mettre à jour l'état en cours à l'aide du précédent état :  $i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$ . Désormais, il s'agit de calculer l'état temporel de la cellule pour le *timestep* en cours :  $\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$ . À présent, calculons l'état réel de la cellule pour le *timestep* en cours, par l'intermédiaire de l'utilisation des fonctions  $f_t$  et  $i_t$ . La motivation derrière la combinaison suivante est empirique : de nombreux informaticiens se sont rendus compte que ce procédé permet au LSTM d'écarter l'information pour ne garder que les informations nécessaires :  $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$ . L'état caché en cours se calcule par la formule  $h_t = o_t * \tanh(C_t)$ . Après tous ces calculs un peu complexes, nous nous retrouvons avec l'état caché pour le *timestep* en cours. Il ne reste plus qu'à calculer l'output réel :  $y_t = \text{softmax}(W_{hy}h_t)$ .

## 2.3. Illustration du fonctionnement d'un LSTM :

Assez parlé de théorie, passons à des exemples de sorties LSTM, pour que vous puissiez comprendre ce qu'est capable de produire un tel réseau. Nous avons essayé de générer du texte dans plusieurs langues et à partir de différentes sources. Voici une illustration de génération textuelle, à partir des écrits de Nietzsche. Au moins 20 *epochs* sont nécessaires pour que le texte soit cohérent d'un point de vue sémantique et linguistique. Bien entendu, un LSTM ne peut se nourrir du texte en substance, il est obligé de convertir les chaînes de caractères en valeurs numériques. **Keras** sera ensuite utile pour créer et entraîner notre LSTM.

Remarque : nous avons dû ajuster à plusieurs reprises les paramètres de notre LSTM, car au départ, il avait des problèmes d'apprentissage et le texte en sortie n'était pas intelligible (à titre d'exemple, lorsqu'il générerait une réplique de personnage, il avait tendance à rendre une phrase constituée d'un voire de deux mots répétés un grand nombre de fois). C'est le grand défaut des LSTM : ne pas comprendre le texte fourni en entrée et son sens. Toutefois, sa capacité à générer du texte de plus en plus intelligible au fil des *epochs* est assez impressionnante.

Nous allons vous montrer ci-après des exemples de sortie LSTM, en français et en anglais. Bien entendu, le texte en sortie est amené à s'améliorer au fil des *epochs*, mais les premiers résultats sont tout de même encourageants (bien que les morceaux de texte en sortie n'aient pas toujours du sens). Dans l'illustration en anglais, nous sommes à une étape plus avancée de l'apprentissage profond, ce qui donne un texte un peu plus intelligible (ceci étant, même avec un très bon LSTM, ce ne sera jamais parfait). Vous remarquerez par ailleurs que la *loss* décroît au fil du temps.

```
Epoch 1/1
92s - loss: 0.1989
2.00

-----

Un soupir moribond
trainait sur la place
de l'espoir
respirain, ils avaient lance les memes Ateints

-----

Epoch 1/1
97s - loss: 0.1869
Mais Joe, Samuel?

--Je ne l'abandonne pas! non certes! et dut l'ouragan s'enfoncer toute la partie de la
construction en la tablier! je

-----

Epoch 1/1
94s - loss: 0.1822
] et
neuf cent soixante-seize fois moins vite que dans cette atmosphA re place A
une vie de mon amie avec collA gues, et enfin aux belles
```

**Fig.6 : Exemple de sortie du LSTM de langue française pour une génération de 100 caractères**

Remarque : Nous pouvons noter que le texte ci-dessous a une connotation très biblique, en effet, parmi le corpus de texte était présent l'ancien testament, d'où un certain penchant « punitif ». A noter que la structure des textes bibliques est conservée mais que le texte lui-même est différent de l'original, par exemple :

```
13:024:004 And there were more chief men found of the sons of Eleazar
than of the sons of Ithamar, and thus were they divided. Among
```

the sons of Eleazar there were sixteen chief men of the house of their fathers, and eight among the sons of Ithamar according to the house of their fathers. [Texte Original]

# Six hundred thousand chosen men, whom the LORD called unto him, and said, Belteshazzan, Manasseh, and Aria, and Manasseh, and Shimeah, and Zerah. And the sons of Shephatiah the son of Asael, the son of Amariah,

13:024:004 They shall be a tine of the river Aethrard, and from the mountains of Ammon, and from t he desert of the wicked; and the spirit of a man hath not been entered into the sea.

46:015:050 But if a man have an end, or of a shearing, and that with a noise with the bow shall be forgiven unto men: but the first day of the week shall be burnt in the steps.

23:031:011 Then shall the land tempts the sea, than that soul shall be cut off from the face of the earth.

19:037:031 Among my people were confing, that thou mayest call my people Israel, and they shall deal treacherously with thee in all things.

42:012:020 And they that hated me her womb-four chamber and with the sword shall be blessed: for h e worketh all that he hath declared to be done again: he shall seek for a meat offering, and a little one is for the sin offering.

03:006:008 And he shall break it in the place whither he shall choose: because the priest the king of Naomin kings.

23:028:011 Send the people away, and a great nation shall be glorious in the things which are able to deliver the sin of the seas.

20:011:019 She shall see the pledge of the streeCPU times: user 11min 59s, sys: 0 ns, total: 11min 59s  
Wall time: 11min 49s

**Fig.7 : Exemple de sortie du LSTM de langue anglaise : réinterprétation de la bible selon Talkative Bot**

Le résultat encourageant sur la première illustration se confirme. Nous avons également entraîné notre LSTM sur de l'allemand. Il serait intéressant de voir les performances de notre LSTM sur des langues aux sémantiques totalement différentes (chinois, etc).

Les résultats sont encourageant et quelques peu effrayant à une certaine mesure.

### 3. Apprentissage d'une image par réseau pré-entraîné :

#### 3.1. Mise en place du contexte :

Avant de parler du *bot Twitter* qui constitue la partie applicative de notre projet, évoquons ce qui se déroule « beneath the surface ». Les modèles de génération de légende d'image ont connu une très forte expansion ces dernières années grâce aux progrès considérables en vision artificielle et en traduction automatique pour produire des légendes d'image réalistes à l'aide de réseaux de neurones. Ce genre de modèle est entraîné pour maximiser la probabilité de produire un texte de légende associé à une image d'entrée, et peut également être utilisé pour générer de nouvelles descriptions d'image.



a woman is sitting on a bus with a white umbrella a dog bites a white dog .

*Fig.8 : illustration de fonctionnement de notre bot Twitter sur deux images*

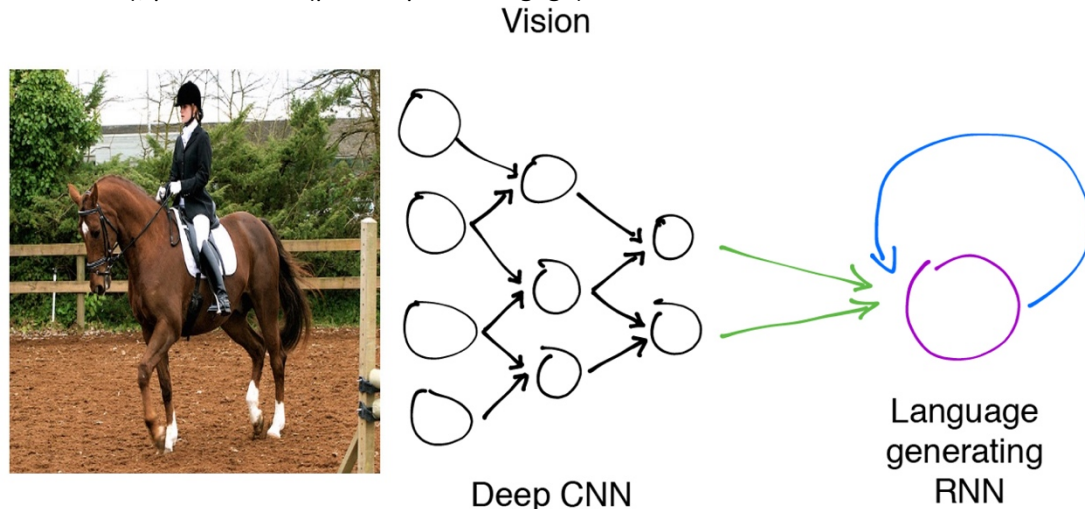
Bien entendu, ces générations de textes restent imparfaites (elles ne correspondent pas forcément au sens premier de l'image), mais nous pouvons d'ores et déjà nous féliciter d'avoir des phrases qui ont un sens (selon votre réseau et votre image, vous n'aurez pas toujours une telle qualité linguistique et sémantique au rendez-vous). Nous allons rapidement parler de la façon de former un générateur de légende d'images. Pour information, nous sommes tombés sur cette technique dans un papier de recherche de Google datant de septembre 2016. La technique en question a donc certainement moins d'un an, bien que cette idée de générer du texte à partir d'images germe dans les esprits des Data Scientists depuis bien plus longtemps.

L'intersection de ces deux champs traditionnellement indépendants offre de nouvelles perspectives très intéressantes. En effet, des applications créatives peuvent nettement améliorer la qualité de vie des individus. Un exemple célèbre d'application est l'algorithme de YouTube qui automatise le *video tagging* à l'aide de l'image, du son, et des mouvements de la vidéo. Par cet intermédiaire, l'objectif de YouTube est de pouvoir taguer n'importe quel vidéo automatiquement. De la même manière que la vision informatique traditionnelle qui a pour but de rendre le monde plus accessible et compréhensible pour les ordinateurs, cette technologie a le potentiel de faire de notre environnement un endroit plus accessible et compréhensible pour tous.



### 3.2. Fonctionnement du modèle :

Avant que le *bot Twitter* ne génère directement du texte à partir d'une image, les images doivent au préalable passer dans deux réseaux de neurones qu'elles vont entraîner : un réseau de neurones convolutifs (dit CNN ; pour la partie vision), puis un RNN (pour la partie langage).



**Fig.9 : Description vulgarisée de notre modèle**

À un haut niveau, c'est le modèle que nous allons former. Chaque image sera encodée par un CNN profond par l'intermédiaire d'une représentation vectorielle de  $2^{12}$  dimensions. Générateur de texte, le RNN décodera cette représentation séquentiellement dans une description de langage naturel. Toutefois, nous pouvons nous éviter la longue phase d'apprentissage, en prenant un couple (*CNN*, *RNN*) déjà entraîné sur un certain nombre d'images. En effet, l'apprentissage par transfert (*transfert learning*) nous permet de récupérer les transformations de données acquises par les réseaux de neurones formés sur différentes tâches et de les appliquer à nos données.

Maintenant que nous avons une représentation d'image, nous avons besoin de notre modèle pour apprendre à décoder cette représentation dans une légende compréhensible. En raison de la nature en série d'un texte, nous allons tirer profit d'un RNN de type LSTM : c'est là que la seconde étape de « *language generating* » intervient. Les LSTM sont formés pour prédire le prochain mot dans une série sachant les mots le précédant et la représentation de l'image. Les cellules LSTM permettent au modèle de mieux sélectionner les informations à utiliser dans la séquence des mots de légende, de mieux savoir ce qu'il faut retenir, ainsi que les informations à oublier. **TensorFlow** fournit une fonction « wrapper » pour générer une couche LSTM pour une entrée donnée et la dimension de la sortie (dans le cas de figure du LSTM, ils s'agit de la dimension totale comptabilisée par le nombre d'éléments de l'espace des lettres ajouté au nombre d'éléments de l'espace des ponctuations). Je ne vais pas approfondir les détails techniques, l'important ici était d'expliquer l'architecture mise en place et ses particularités, dans l'optique de comprendre la transition entre une image fournie par un utilisateur et la production de texte par le *bot Twitter*.

Remarque : Nous tenons à apporter une clarification sur l'ensemble de ce modèle. Le training du modèle n'a pas été réfléchi par nos soins, car cette partie ne constitue pas le cœur du projet. Il s'agit d'une fonctionnalité supplémentaire permettant d'améliorer l'expérience utilisateur lorsqu'il interagit avec le bot. De plus, nous ne maîtrisons pas totalement les spécificités du framework bas niveau **TensorFlow**. Voici le GitHub pour la partie training : <https://github.com/mlberkeley/oreilly-captions>

## 4. Interfaçage du bot avec Twitter

Pourquoi un bot Twitter ? A l'origine, l'idée du projet était de réaliser une interface web interactive capable d'exécuter du code sur une instance distante (ex. un serveur Cloud). Néanmoins, nous nous sommes très vite aperçus que cela était trop ambitieux : bien que désireux d'apprendre un nouveau framework tel que Django, on a préféré s'attarder sur un livrable fonctionnel et simple plutôt que de rajouter de l'incertitude en rajoutant de la complexité... En effet le projet étant déjà très riche en fonctionnalités, on ne souhaitait pas prendre le risque d'échouer sur la partie « front » rendant alors tout le travail en amont invisible pour l'utilisateur.

L'idée du *bot Twitter* permet donc d'éviter de nombreux risques tout en conservant un aspect interactif par le biais des tweets et des commentaires.

### 4.1. Twitter API

Une fois le compte créé et les clés API récupérées, il est facile de créer un automate grâce à la librairie `python-twitter`<sup>3</sup> (`pip install python-twitter`) qui est un wrapper de requêtes html à l'interface API de twitter. Cela nous évite donc de gérer des appels URL verbeux qui sont remplacés par des méthodes python plus claires.

Voici les fonctionnalités que l'on a développées en s'appuyant sur l'API :

- La récupération du flux d'actualités du bot
- La récupération des médias postés ainsi que de l'identifiant du post concerné pour y répondre ultérieurement
- L'automatisation des réponses avec contenus spécifiques (images, textes, etc...)

### 4.2. Interfacer tous les modules

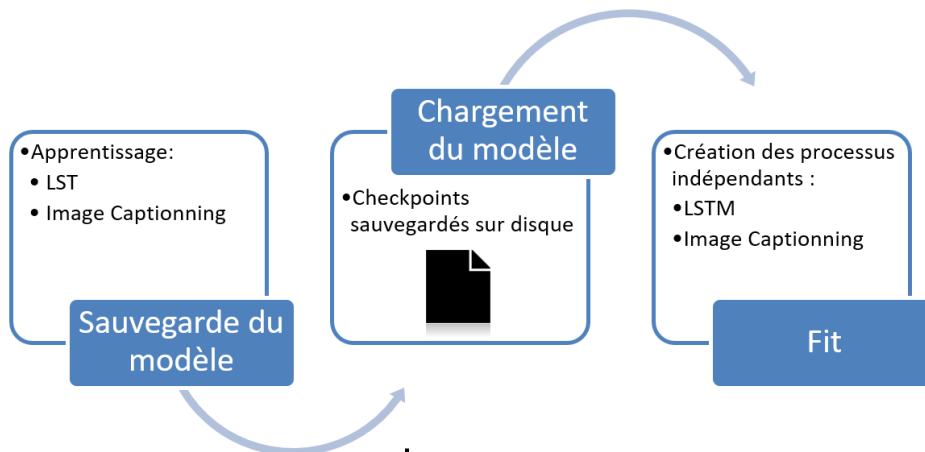
Pour être pleinement fonctionnel, le bot se doit d'intégrer toutes les fonctionnalités que l'on décrit plus haut, c'est-à-dire :

- Récupérer l'image à identifier sur son flux d'actualité
- Identifier l'image à partir du modèle d'image captionning pré-entraîné
- Gérer du texte à partir du modèle LSTM pré-entraîné
- Composer la réponse : texte ou image
- Poster la réponse.

Il est donc nécessaire de s'assurer que toutes les briques fonctionnelles s'emboîtent correctement et que chaque module sache chercher correctement la donnée qui lui est nécessaire pour fonctionner.

---

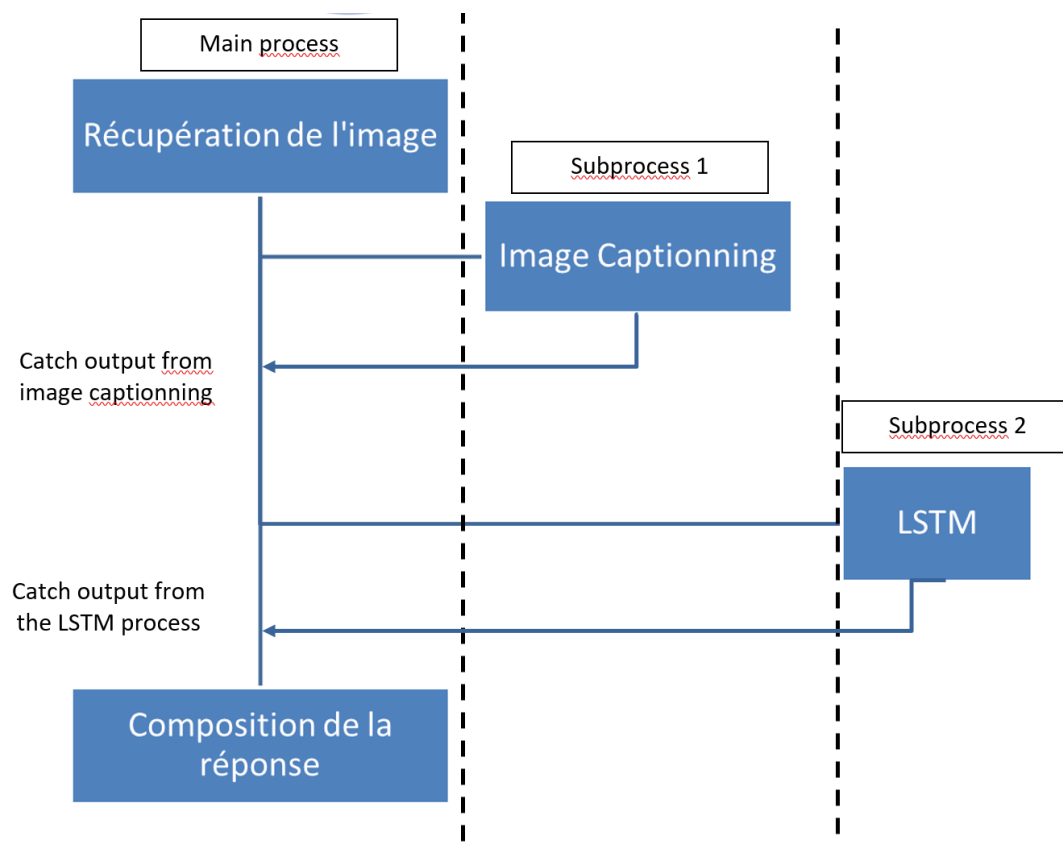
<sup>3</sup> <https://python-twitter.readthedocs.io/en/latest/>



**Fig.10 : Illustration de l'utilisation des modèles sauvegardés**

Notre méthode est sous optimale : par soucis de temps nous avons stocké en dur les chemins relatifs des appels aux différents fichiers, rendant le code peu flexible aux erreurs de chemins. Il est donc déconseillé de modifier l'arborescence fichier du livrable. Par ailleurs, cela implique également que les checkpoints des modèles d'apprentissage soient dans des répertoires spécifiques afin que le modèle puisse le charger.

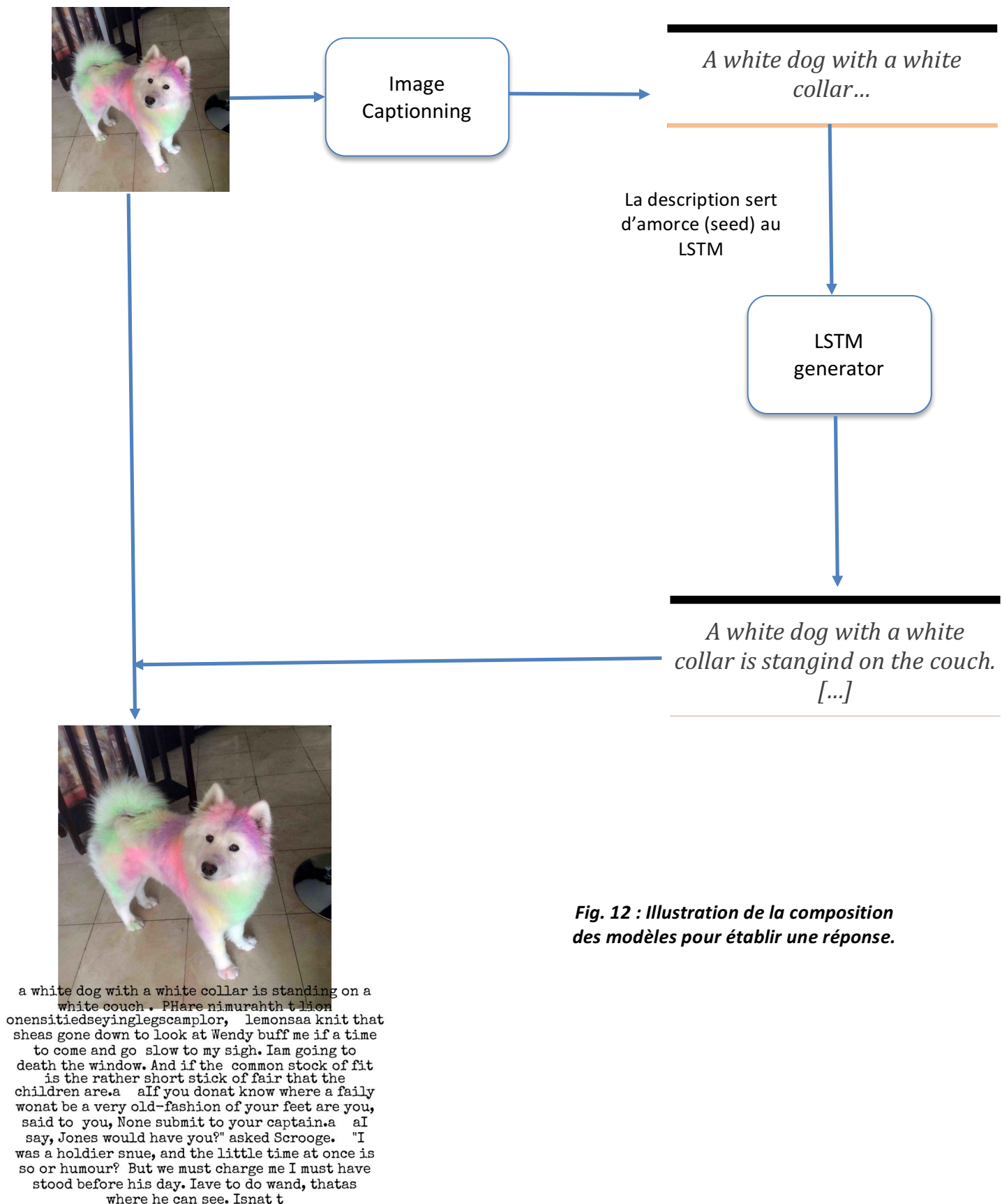
Par ailleurs les deux modèles de machine learning étant différents et n'utilisant pas les mêmes librairies, des conflits apparaissent lorsque les deux modèles sont appelés de façons séquentielle : en effet un modèle chargé dans la carte graphique ne libère la mémoire de celle-ci qu'à la fermeture de son processus. Or, dans un script, les modèles sont enchainés par le même processus. D'où la nécessité de créer des sous processus indépendants qui se clôturent automatiquement :



**Fig.11 : Illustration des sous processus cloisonnés pour éviter les incompatibilités entre les modèles**



### 4.3. Composition de la réponse :



**Fig. 12 : Illustration de la composition des modèles pour établir une réponse.**

On peut constater que le LSTM a quelques difficultés à générer une histoire à partir d'une amorce trop restrictive : en effet on constate que juste après le texte d'amorce, le LSTM génère un signal incohérent (texte non compréhensif) car cette amorce (seed) est trop éloignée du corpus sur lequel le LSTM a appris. Néanmoins, après un certain nombre de caractère, le LSTM se stabilise et génère un texte plus cohérent. C'est notamment le cas avec l'exemple ci-dessous : le corpus sur lequel le LSTM a appris ne comporte aucun terme de vocabulaire automobile qui est fondamentalement très éloigné des textes à caractère biblique.

Ci-dessous un exemple où le modèle d'image captionning n'est pas très loin de la réalité :



a woman is sitting on a bus with a white umbrella  
 . PHalastspenistine evergeashnedermysaipints,  
 maimeto May well-clothed on his mind. Starriest  
 heart heally hairs about him with his head to the  
 other hand, and the street winds profound at  
 Seat. They were deeply in a pack old laugh, and  
 soon the sam wither he had become all right.  
 aNow,a said he, aitas you moregretf, Phinty-me!"  
 "Why, I never come to a pity will obey me?" "We  
 mustnat think of something I liked so much to  
 speak. The Banker brought him in her face  
 toward Peter With life, when he was tak'ng the  
 dore and look'd Allow'd th' Ethereal wings, and  
 the deeds And from the great Common Profest  
 vertue are then

**Fig.13 : Illustration de l'approximation cohérente de l'Image Captionning**

On constate que :

- Woman, signifie qu'il a bien identifié un humain
- Bus signifie qu'il a bien détecté un composant automobile
- Umbrella est dû à la forme et la couleur de la portière de la voiture.

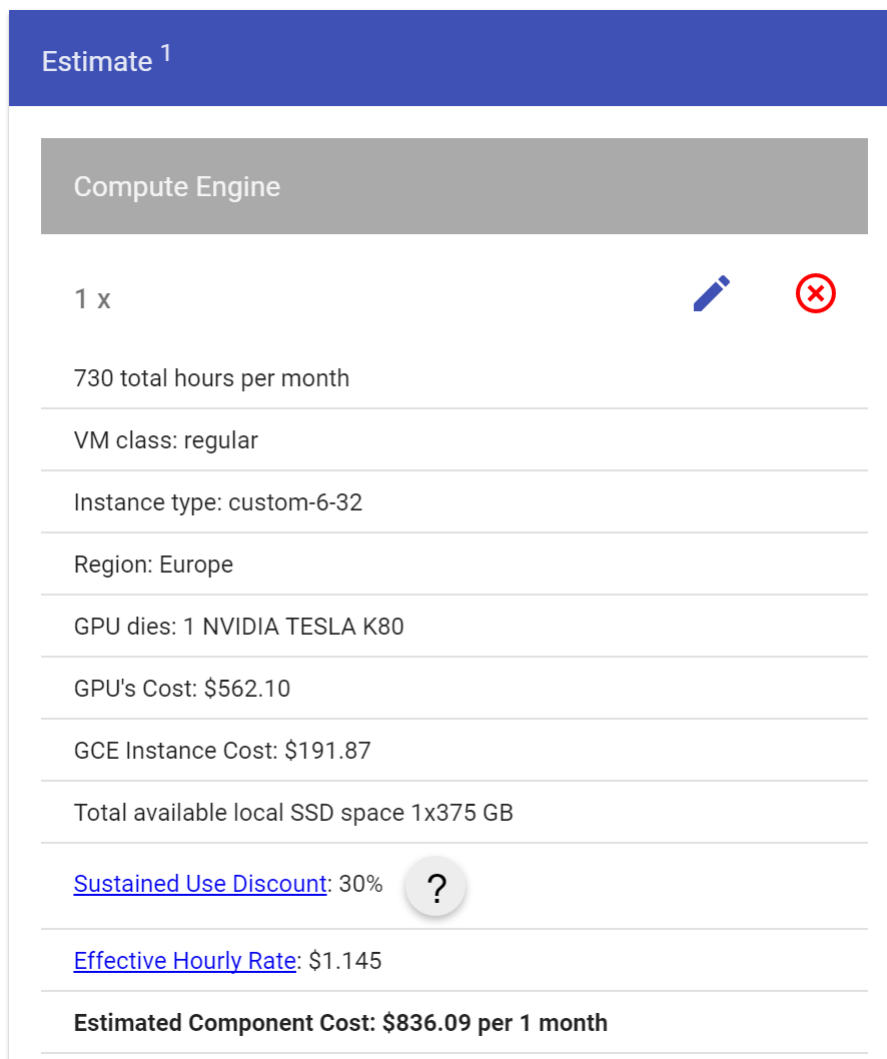
L'image captionning est donc cohérent, l'identification des features de haut niveau est relativement performante, mais éloignée de la réalité (« Tony Stark sortant de son Audi R8 »).

## 5. Outils de travaux :

### 5.1. Google Cloud :

Etant donné que les LSTM sont coûteux en temps pour entrainer ce type de réseaux, nous avons profité de l'offre de Google Cloud (\$300 offerts sur leur plateforme pour la création d'un compte découverte) pour déployer un serveur cloud équipé d'une GPU : Nvidia K80, afin de booster les temps de calcul. De plus, cela permet de lancer sereinement des calculs qui durent plusieurs jours sans se préoccuper de la santé de son matériel informatique.

Pour notre configuration, la tarification est de 1.14\$/heure :



The screenshot shows the Google Cloud Compute Engine configuration interface. At the top, there's a blue header with 'Estimate <sup>1</sup>'. Below it, a grey box contains the text 'Compute Engine'. The main configuration area lists the following details:


- 1 x (with edit and delete icons)
- 730 total hours per month
- VM class: regular
- Instance type: custom-6-32
- Region: Europe
- GPU dies: 1 NVIDIA TESLA K80
- GPU's Cost: \$562.10
- GCE Instance Cost: \$191.87
- Total available local SSD space 1x375 GB
- [Sustained Use Discount](#): 30% (with a question mark icon)
- [Effective Hourly Rate](#): \$1.145
- Estimated Component Cost: \$836.09 per 1 month

**Fig.14 : Capture d'écran de la configuration**

Une fois le serveur déployer, il est nécessaire de configurer et d'installer la suite logicielle nécessaire pour développer (**CUDA toolkit, Anaconda, TensorFlow**, etc). Il a fallu ensuite configurer Jupyter pour s'exécuter dès le démarrage et à se comporter comme un service continu. Enfin, il a fallu ouvrir les ports du serveur afin d'exposer Jupyter à l'extérieur ; le rendant accessible depuis partout.

## 5.2. Outils logiciels :

Niveau d'abstraction



- Nous nous sommes servis de **Git**, et plus particulièrement de GitHub, pour travailler de façon collaborative et pour pousser/synchroniser le code sur nos différentes plateformes (ordinateurs personnels et serveur cloud). Au début, nous avions pour ambition d'utiliser GitHub de la manière la plus professionnelle possible, mais on s'est retrouvé très vite à devoir gérer de nombreux conflits : on s'est majoritairement servi de l'outil pour synchroniser le code plutôt que pour développer collaborativement dans les règles de l'art...
- **Keras** est la librairie qui a été privilégiée pour développer le réseau de neurones cœur à notre sujet : le LSTM. En effet, il s'agit d'une API de haut niveau s'appuyant sur **TensorFlow** ou **Theano** comme backend.
- La seule partie développée en **TensorFlow** pur est le modèle d'image captionning. Néanmoins, il est nécessaire de rappeler que nous nous sommes très fortement appuyés sur un code déjà rédigé (notamment pour l'apprentissage), car nous ne maîtrisons absolument pas ce framework. Nous avons néanmoins modifié une partie du code du modèle pour répondre à nos exigences d'interfaçages.
- **CUDA** pour accélérer les calculs sous GPU : il s'agit d'un framework très bas niveau, transparent pour l'utilisateur, **TensorFlow** s'appuie sur **CUDA** pour exécuter ses calculs sous GPU.

Ce projet nous a donné l'occasion d'appréhender une grande diversité d'outils nous permettant ainsi de développer de nouvelles compétences et de mieux comprendre ce qui se passait dans le monde du deep learning.

Remarque : face au coût important d'une instance cloud, nous avons développé les modules et entraîné des petits jeux de données sur nos ordinateurs portables avant de pusher le code au dernier moment sur la plateforme Cloud afin de ne pas perdre du temps de calcul à tester des itérations infructueuses et nous ajouter des coûts inutiles.

## Conclusion :

Nous avons synthétisé les phases clés de notre projet et les améliorations envisagées dans le tableau ci-dessous :

Objectif	Réalisé	Satisfaction vis-à-vis de l'objectif	Améliorations futures
Scrapping web	Oui	Oui	<ul style="list-style-type: none"> <li>• Stockage dans base de données</li> <li>• Industrialisation du crawling</li> </ul>
LSTM	Oui	Oui	<ul style="list-style-type: none"> <li>• Augmenter le nombre de corpus et diversifier les sujets d'apprentissage</li> </ul>
Image Captionning (non prévu)	Oui	Non	<ul style="list-style-type: none"> <li>• Implémenter soit même la partie learning</li> <li>• Passer le code sous Keras</li> <li>• Renforcer le modèle avec du word2vec</li> </ul>
Site web	Non	Non	<ul style="list-style-type: none"> <li>• Diversifier les fonctionnalités</li> <li>• Appréhender le framework Django</li> </ul>
Bot	Oui	Oui	<ul style="list-style-type: none"> <li>• Apporter une compréhension du langage naturel</li> <li>• Consolider sa robustesse aux erreurs</li> </ul>

Comme nous pouvons le constater, le projet s'est écarté de ce qui était prévu à l'origine :

- La décision a été prise d'abandonner l'interface interactive sous forme de site risquant de ne pas fonctionner à temps et rendant le travail réalisé jusqu'alors vain.
- Pour pallier cette décision et pour conserver l'interaction avec un utilisateur, nous avons décidé d'élaborer un *bot Twitter*, ce qui nous a contraint à ajouter quelques fonctionnalités telles que :
  - L'image captionning pour interpréter les images des utilisateurs. Etant donné que cette fonction n'était pas prévue et ne constitue pas le cœur du sujet, nous nous sommes appuyés sur des travaux pour la partie d'apprentissage du modèle d'image captionning.
  - Composer du texte sous format image afin de surmonter la limite des 140 caractères imposée par Twitter.

Il est intéressant de constater que le projet est si vaste que de nombreux axes d'améliorations sont à envisager afin de parfaire le bot. Ce travail restant un travail d'amateurs, beaucoup de travail reste à faire concernant l'industrialisation du bot afin de le rendre plus robuste.

En définitive, nous faisons quelques constats sur les caractéristiques techniques de notre projet, ainsi que ses apports :

Dans un premier temps, nous aimerions rappeler qu'il est impératif d'entraîner le LSTM :

- soit du des sujets très précis
- soit sur des corpus très larges

Problème ? Cela implique d'avoir un volume très important de données.

Autre problème rencontré : si les textes de mon corpus n'ont aucun rapport avec l'image que je donne au *bot Twitter*, cela va certes générer une première phrase relativement « potable » (comme toujours, car cette phrase résulte du modèle (CNN,RNN) pré-entraîné que nous avons évoqué auparavant), mais le reste des phrases sont déstructurées et n'ont soit pas de sens, soit un sens trop éloigné de l'image. D'où l'intérêt d'avoir un corpus large qui recouvre suffisamment de sujets.

Ce projet a également représenté une belle opportunité de toucher à de nombreuses briques technologiques. Nous avons eu la chance de pouvoir réaliser un travail assez complet qui couvre un large spectre, et d'un point de vue personnel, nous avons beaucoup appris en travaillant sur un projet d'une telle envergure et qui nécessite la manipulation de nombreux outils. C'est un projet que nous envisageons de pousser pour lui apporter les améliorations envisagées.

## Bibliographie :

<https://twitter.com/thetalkativebot>

<https://keras.io/>

<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

<https://github.com/mlberkeley/oreilly-captions>

<https://www.tensorflow.org/>

<https://medium.com/google-cloud/running-jupyter-notebooks-on-gpu-on-google-cloud-d44f57d22dbd>

## Annexe :

Nous avons parlé du fait que le LSTM rédigerait lui-même son script de cinéma. Le LSTM est suffisamment intelligent pour conserver et reproduire la structure (mise en page) d'un script de film.

En effet, il prédit à la lettre près et est donc capable de gérer la ponctuation, les espaces, retours à la ligne, etc.

AUSTIN  
 Of course, I had my nine-making my dedicated.  
 CUT TO:

INT. DOCTOR'S OFFICE

DR. EVIL  
 (the light shoot)  
 It's not too much.

AUSTIN  
 (hoarse)  
 It's a secirical comedal. Everyone.

AUSTIN  
 (looking around)  
 Why don't need a fuck out of the job.

REGA  
 (trance-like)  
 No, they came like some creating. I can imagine mething for hugg man, I don't son fond of stuck boxes?

\*  
 407 NORM \*  
 Approaching the beans, and fired up playing her face.

NICO  
 -- Nobody. \*

AGENT HALLOWAN  
 Brown from Unite for her like it fuckin' permission.

MAX  
 I didn't say. What the hell you?

INT. MAX'S OFFICE - NIGHT

A STEDICAM in front of them with someone shifting perfect tightly.

BURNETT  
 Yeah... I feel a little funny.

BURNETT  
 What the fuck?

MAX  
 (aside)  
 Now, why don't we still the group and heart you are stupidger over the reprecadestion. I have to hang in. I escause me machine and a ren't far team.

IVANA  
 No, this inevery and it's only mon.

INT. DR. EVIL'S MOON BATER, CLOBEROOM

IN A SMOPHING VIBLAS her back to the next room.