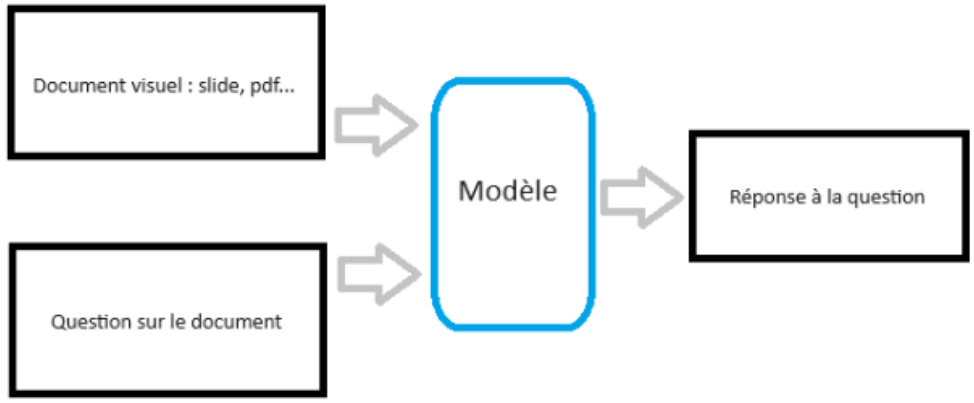
JustAI – Protocole : construction d'un dataset de slides en français pour le Document Visual Question Answering.

Alexandre MYARA – Stage Master 1 2024

Ce document a pour objectif de décrire la démarche utilisée pour construire un dataset servant à entrainer des modèles à faire du **Document Visual Question Answering (DocVQA)**.

Pour entrainer un modèle sur cette tâche il est nécessaire d'avoir un dataset associant au minimum : (document visuel, question) avec une réponse.



Les datasets de Doc VQA populaires sont en majorité en anglais. De plus ils sont très peu souvent constitués de slides. Notre cas d'usage étant des slides en français, il est donc pertinent de construire un dataset recouvrant les deux exigences.

Ce document retranscrit la méthode derrière la construction d'un dataset pour DocVQA en français. Cette méthode a été testée afin de construire un dataset d'environ 100 documents mais avec plus de temps elle serait tout à fait applicable à la construction d'un dataset de taille plus grosse.

Sommaire

I. Description complète du dataset
II. Récolte des données
III. Datasets brut et Filtres sur les données brutes
IV. Génération des Questions/Réponses

I. Description du dataset

url présentation	titre présentation	auteur	date	nombre de slides	description	langue	Dim. slide	Nombre de likes	Transcripte de chaque slide	Liste des slides les plues	images	Q/A

Chaque ligne représente une présentation d'une longueur N slides.

url présentation : **string**, url slideshare.net de la présentation.

titre présentation : **string**, titre de la présentation

auteur : **string**, username de l'auteur sur slideshare

date : **string**, format AAAA-MM-JJ

nombre de slides : **int**, nombre de slides de la présentation

description : **string**, description de la présentation

langue : **string**, langue renseignée par l'auteur de la présentation

dim slides : **tuple**(int, int), dimension d'une slide HxW

nombre de likes : **int**, nombre de like sur la présentation

transcripte : **List**[string], liste contenant la transcription du texte de chaque slide

slides les plus lues : **List**[int], one-hot vecteur où les 1 correspondent aux slides les plus lues.

Images : **List**[Image], il s'agit d'une liste de toutes les slides de la présentation.

Q/A : **List**[dict], il s'agit d'une liste de dictionnaire de format {question1 :, réponse1:, question2 :, reponse2 : ...} recessant les paires questions/réponses de chaque slide (une slide peut avoir plusieurs paires question/réponse)

II. Récolte des données

La première étape est de **sélectionner une source de donnée**. Ici on a fait le choix de travailler avec des présentations slides. Une présentation slide peut contenir de multiples ou une seule slide.

Une source de donnée efficace est le site **Slideshare.net**. Il s'agit d'un site rassemblant, par thème, des présentations slides (PowerPoint, Google Slides etc).

De plus, le site offre la possibilité de sélectionner la langue du document. En allant sur le site et en extrayant efficacement les slides de la partie « En français » du site il est possible d'avoir un très grand nombre de présentations slides (et donc une grande base de données de slides).

Pour chaque présentation, on s'occupe d'extraire les slides et les métadonnées associées à chaque slide (les descriptions slides par slides, les mentions « slide plus lue », url des images) mais aussi les métadonnées de la présentation (Titre, Nombre de slides, Nombre de partages, nombre de likes).

On automatise la tâche de récupération des données en bouclant sur les étapes suivantes :

- ➔ Pour chaque page de résultat de recherche on enregistre tous les liens des présentations.
- ➔ Pour chaque lien de présentation on scrap les données nécessaires.

```
#page_i is the url format of the content search page in French on slideshare
#the first page is page_i.format(1), the second page is page_2.format(2) etc...
#nb pages is the number of the last content search page that we scrap.
page_i = "https://fr.slideshare.net/search?searchfrom=header&q=fran%C3%A7ais&page={}"
nb_pages = 1
```

✓ 0.0s

```
#Execution of the scraper
for i in range(1,nb_pages+1) :
    urls = extract_urls_from_search(page_i.format(i))
    for url in urls :
        scrap(url)
```

✓ 26.3s

Ce script nécessite seulement l'url générale de la page des résultats de la recherche « français » (page_i ici)

Pour chaque page de recherche, on scrap les urls des présentations figurant sur cette page de recherche avec **extract_urls_from_search(search_url).**

Une fois les urls stockées dans une liste de string *urls* on peut scraper chaque présentation à l'aide de la fonction **scrap(url)**.

La fonction scrap(url) créé un dossier du nom de la présentation dans le répertoire des données.

Elle le remplit avec :

- un fichier json qui répertorie les métadonnées (titre, date, slides les plus lues etc)

- les slides une par une au format .jpg

Ce script est fait tourner par le notebook scrap.ipynb

III. Datasets brut et Filtres sur les données brutes

A ce stade on a pu récupérer les données brutes. Elles sont réparties dans différents dossiers.

On répartit les données au format Datasets du module datasets.

On utilise le notebook create_dataset.ipynb pour instancier un objet Dataset au format décrit plus haut (I).

Ensuite il est nécessaire de trier les données, certaines présentations ou slides peuvent ne pas être pertinentes pour l'entrainement.

Par exemple, une présentation d'une seule slide ou avec peu de partages/likes peut être une présentation peu instructive.

On construit donc des filtres sur les données avant de passer à l'étape génération de paires de questions/réponses.

Liste des filtres :

On dresse les fonctions de répartitions des différentes grandeurs numériques. Pour les strings on vérifie qu'il n'y a pas de valeurs manquantes (nulle).

➔ Une présentation doit avoir une longueur >1er décile
➔ Le nombre de partage >1er décile
➔ Le nombre de like > 1er décile

➔ Les descriptions des slides doivent être en français

Les slides ne passant pas les filtres sont supprimées du dataset.

Dans le notebook create_dataset.ipynb il suffit d'utiliser la fonction **filters(dataset)** permet d'exécuter les filtres.

IV. Génération de questions/réponses

On a sélectionné les données d'assez bonne qualité pour être envoyées à un modèle afin de générer les paires de questions réponses.

On fait le choix ici de travailler avec Gemini mais tout autre modèle similaire fonctionne tant qu'il a un API avec documentation disponible.

On a à cette disposition le notebook generate_qa.ipynb.

Il prend en entrée le dataset filtré puis génère pour chaque slide une paire de question/réponse au format : {question :, réponse :}

On s'assure que les réponses ne soient pas trop courtes ou que les questions/réponses ne soient pas vides.

On a finalement un dataset prêt au format dataset.