RAPPORT TECHNIQUE

Aude, Baptiste LG, Patricia, Ronan

ATTENDU :

Le développement doit comporter :

- Un corpus permettant au chatbot Microsoft IA Brest+ d'apprendre à répondre en français (en anglais par la suite),
- Une API donnant accès au corpus,
- Un modèle d'IA pour les réponses et une étude de performance.
- Une page web intégrant le chatbot et utilisant Tensorflow JS,

Contexte du projet

Un chatbot (to chat + bot) est un robot conversationnel qui permet de répondre automatiquement, via une messagerie instantanée, aux requêtes/demandes d'un client (un être humain).

L'objectif phare du projet est de créer un chatbot. Dans ce cas de figure, les clients vont être des nouveaux apprenants ou bien des partenaires potentiels de l'école qui posent des questions précises sur l'école Micorosoft IA Brest+. Le premier défi est que le chatbot puisse répondre, automatiquement via des méthodes d'intelligence artificielle, à ces questions. Quant au deuxième défi, le chatbot doit, à travers ses réponses, convaincre les visiteurs soit de postuler (les apprenant) ou de devenir un partenaire officiel. Par exemple, il doit intégrer des phrases qui font la force de l'école : Pédagogie active, Travail en équipe, Learning by teaching, évaluation par compétence, Type de partenariat, ...

SOMMAIRE

I- Qu’est-ce qu’un chatbot ?

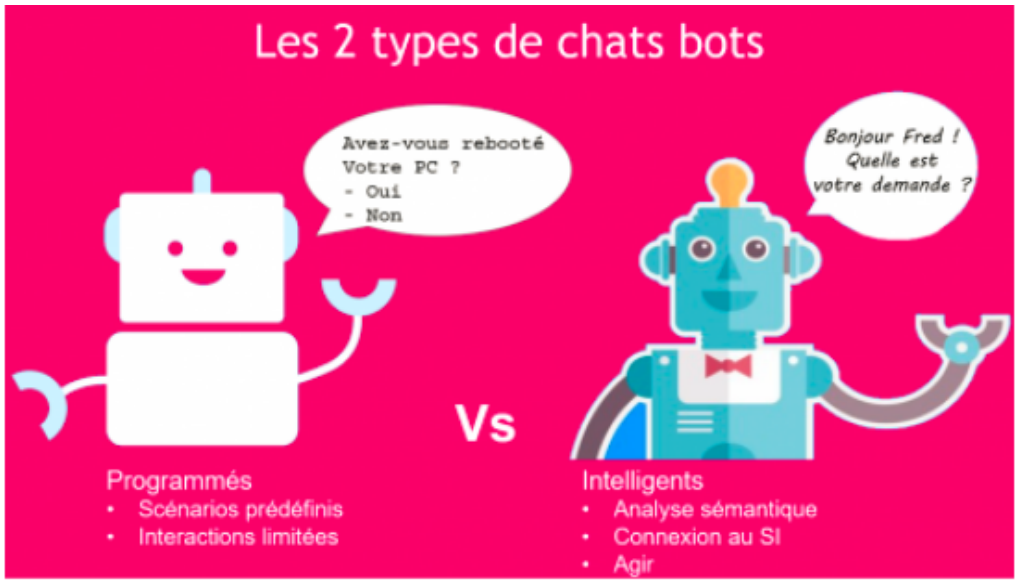
Contraction anglaise des mots
« chat » (discuter)
« bot » (robot)

Un chatbot est défini comme un logiciel permettant de comprendre un texte (ou une conversation vocale), de l’interpréter afin d’en déduire une ou plusieurs actions à mener.

comparatif entre Chat / chatbot / FAQ / Tél / Emails avec les avantages et inconvénients de ces différentes solutions:

	Chatbot	Chat	FAQ	Tél	eMails
Personnalisation messages	X	X		X	X
Multicanal	X				
Scalable	X		X		
Coût à l'interaction	Très Faible	Moyen	Très Faible	Fort	Moyen
Disponibilité 24h/24	X		X		
G° de Problèmes	Simples	Medium	Simples	Tous	Tous
Erreurs interprétations	Moyen	Faible	Moyen	Faible	Faible
Coût mise en place	Important	Moyen	Faible	Important	Moyen
Type de problèmes	Simples	Tous	Simples	Tous	Tous

Les différents types de chatbots :



1 - Les bots d'interaction textuelle sans IA :

Ils utilisent des questions prédéfinies et suivent une logique séquentielle, émulant une conversation à partir d'un menu d'options préalablement établi.

2 - Les chatbots dotés d'intelligence artificielle :
Ils ont la capacité de comprendre et de traiter le langage naturel pour fournir des réponses plus personnalisées.
Contextuels, ces chatbots sont capables d'interpréter l'intention de l'utilisateur et de formuler des réponses à partir de zéro, donnant un sens plus dynamique et authentique à la conversation.
Ces robots dits « cognitifs » sont en mesure de tirer des enseignements des interactions passées, ce qui permet aux dialogues de devenir de plus en plus fluides et précis.

A - Comment fonctionnent les chatbots AI / ML?
Les chatbots alimentés par l' intelligence artificielle **utilisent couramment trois méthodes de classification**. Les chatbots peuvent appliquer toutes ces méthodes dans différentes situations.

1. Correspondances de motifs:

Les chatbots utilisent la correspondance de modèles pour **classer le texte et produire une réponse appropriée pour les clients**.

Les chatbots de correspondance de modèles utilisent une base de connaissances qui contient des modèles de discours et des modèles particuliers. Lorsque le chatbot reçoit une entrée qui correspond à un modèle existant, il envoie le message stocké dans le modèle en tant que réponse.
Le modèle peut être une phrase du type "Quelle heure est-il?" ou un modèle «L'heure est *», où le «*» est une expression régulière. Un modèle unique doit être disponible dans la base de données pour chaque type de question.

2. Compréhension du langage naturel (NLU)

La compréhension du langage naturel (NLU) est un sous-ensemble de l'IA qui permet une interaction homme-machine. **NLU exploite une hiérarchie de modèles de classification pour analyser l'entrée de texte ou de parole**:

Classificateur de domaine - classe l'entrée dans l'un des groupes prédéterminé de domaines de conversation. Ce classificateur est utilisé uniquement pour les conversations sur plusieurs sujets nécessitant un vocabulaire spécialisé. Par exemple, d**es assistants virtuels comme Siri utilisent des classificateurs de domaine pour répondre à des questions sur les sports, la météo, la musique ou la navigation.**

Classificateurs de rôle - effectue un étiquetage des entités en fonction du contexte. Par exemple, **vous pouvez classer davantage l'heure en l'étiquetant «ouverte» ou «fermée».**

Classificateur d'intention - détermine ce que l'utilisateur essaie d'accomplir en affectant chaque entrée à l'un des intentions spécifiées dans votre algorithme NLP. Par exemple, **une intention peut être «trouver les heures d'ouverture», «trouver un produit», «trouver le magasin le plus proche».** Reconnaissance d'entité - extrait les phrases et les mots essentiels pour répondre à l'intention de l'utilisateur. Par exemple, si l'utilisateur tente de réserver une table dans un restaurant, l'intention inclura la date, l'heure et le nombre de personnes.

3. Traitement du langage naturel

La NLP fait référence à la synthèse et à l'analyse des langues humaines. Les algorithmes NLP utilisent une combinaison de techniques d'analyse statistique, prédictive, d'exploration de données et de modélisation de données pour générer des informations de manière proactive. **La NLP permet au chatbot d'apprendre et d'imiter les modèles et les styles de conversation humaine. Cela vous donne le sentiment que vous parlez à un humain, pas à un robot.**

B - Chatbots génératifs ou basés sur la récupération :

Décider de la meilleure technique pour traiter les entrées de dialogue et générer des réponses est l'une des premières décisions à prendre. La plupart des systèmes actuels utilisent des méthodes basées sur la recherche, tandis que les méthodes génératives sont encore à l'étude.

Basé sur la récupération :

Les chatbots basés sur la récupération fonctionnent sur le principe des graphiques ou des flux dirigés. Le chatbot est formé pour **fournir la meilleure réponse possible à partir d'une base de données de réponses prédéfinies. Les réponses sont basées sur des informations existantes.**

Les chatbots basés sur la récupération utilisent des techniques telles que la correspondance de mots-clés, l'apprentissage automatique ou l'apprentissage en profondeur pour identifier la réponse la plus appropriée. Quelle que soit la technique, **ces chatbots ne fournissent que des réponses prédéfinies et ne génèrent pas de nouvelle sortie.**

Un exemple de chatbot basé sur la récupération est Mitsuku . Il contient plus de 300 000 modèles de réponse prédéfinis et une base de connaissances de plus de 3 000 objets. Ce chatbot peut construire des chansons et des poèmes en fonction de sa base de connaissances.

Génératif

Les systèmes basés sur l'extraction sont **limités à des réponses prédéfinies**. Les chatbots qui utilisent des méthodes génératives peuvent générer un nouveau dialogue basé sur de grandes quantités de données d'apprentissage conversationnel.

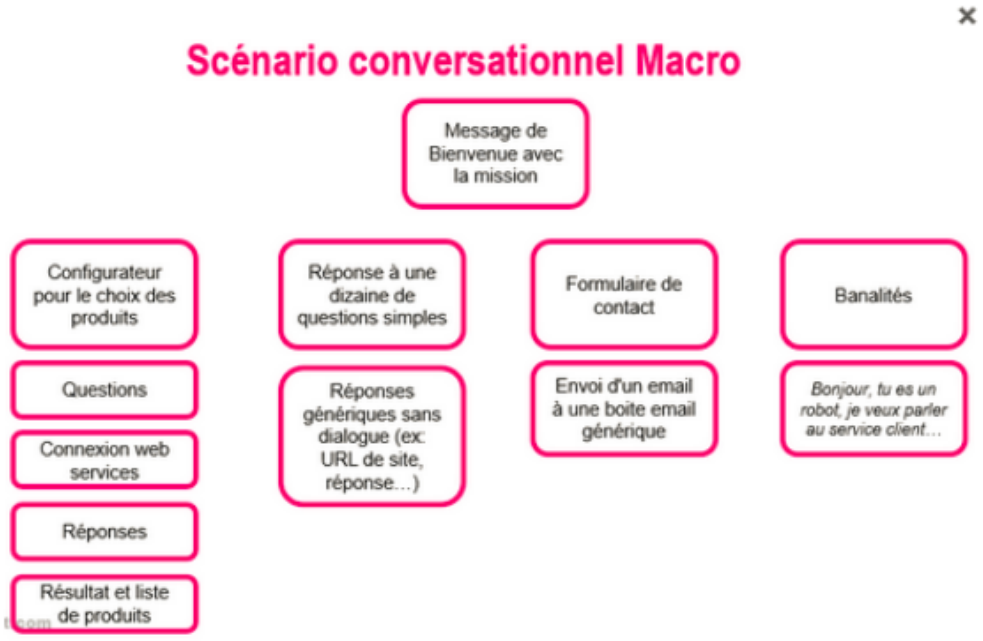
Les chatbots génératifs utilisent une combinaison d'apprentissage supervisé, d'apprentissage non supervisé , d'apprentissage par renforcement et d'apprentissage contradictoire pour une formation en plusieurs étapes.

L'apprentissage supervisé structure une conversation comme un problème séquentiel. L'apprentissage de séquence à séquence mappe les entrées de l'utilisateur à une réponse générée par ordinateur. Cependant, ce type d'apprentissage a tendance à donner la priorité aux réponses à forte probabilité comme «je sais». Les systèmes d'apprentissage supervisé ont également du mal à inclure les noms propres dans leur discours, car ils apparaissent moins dans le dialogue que d'autres mots. En conséquence, les chatbots d'apprentissage supervisé semblent répétitifs et ne peuvent pas favoriser une conversation humaine stable.

Pour résoudre ce problème, les développeurs tirent parti de l'apprentissage par renforcement pour enseigner aux chatbots comment optimiser les dialogues pour une récompense cumulative.

3 - **À chaque objectif, son bot conversationnel**

- Les chatbots d'interactions clients
- Les chatbots d'accompagnement client
- Les agents conversationnels à but marketing
- Les bots destinés aux réseaux sociaux
  un message envoyé via Facebook Messenger peut obtenir
  - un taux d'ouverture de 88 % (contre 33 % en moyenne pour un email)
  - un taux de clic de 56 % !
- Les chatbots internes



4 - Comment mettre en place un chatbot ?

- Listez les cas d'usages que pourrait traiter le bot
- Sélectionnez 1 à 4 uses cases très précis à traiter par le bot.
- Validez que les cas d'usage sélectionnés peuvent être répondus par le bot.
- Modélisez le flux des conversations
- Modélisez le Bot
- Optimisez les questions et les réponses
- Faites des tests avec des opérationnels
- Faites de l'amélioration continue jusqu'à atteindre le niveau de qualité suffisant
- Validez définitivement la pertinence du bot
- Peaufinez la communication sur le bot et corrigez le tir avec les premiers retours

Technologie du chabot retenue

Nous optons pour **un chabot doté d'intelligence artificielle basé sur la récupération**

II - DESCRIPTION DU MODÈLE :

1- Vision globale de notre modèle :

Import des librairies
Les données
Prétraitement des données
Longueur d'entrée, longueur de sortie et vocabulaire
Réseau de neurones
Analyse du modèle
Test

2- Explication de notre modèle

A - LES LIBRAIRIES :

- tensorflow
- numpy
- pandas
- json
- nltk
- matplotlib

Tensorflow :

bibliothèque open source de Machine Learning, créée par Google, permettant de développer et d'exécuter des applications de Machine Learning et de Deep Learning.

Pour faire simple, il s'agit d'une boîte à outils permettant :

- de résoudre des problèmes mathématiques extrêmement complexes avec aisance.
- aux chercheurs de développer des architectures d'apprentissage expérimentales et de les transformer en logiciels.
- d'entraîner et d'exécuter des réseaux de neurones pour la classification de chiffres écrits à la main, la reconnaissance d'image, les plongements de mots, les réseaux de neurones récurrents, les modèles sequence-to-sequence pour la traduction automatique, ou encore le traitement naturel du langage.

Numpy :
bibliothèque pour langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.

Pandas
bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles.

Json
(JavaScript Object Notation) est un langage léger d'échange de données textuelles.
Pour les ordinateurs, ce format se génère et s'analyse facilement. Pour les humains, il est pratique à écrire et à lire grâce à une syntaxe simple et à une structure en arborescence

Nltk
Boîte à outils permettant la création de programmes pour l'analyse de texte.

tensorflow Tokenizer
tokenizer =>outil qui segmente un texte en balises, ou tokens. Chaque token correspond à une étiquette linguistiquement unique propre à la langue de l'analyse.
Tokeniser un texte, c'est le nettoyer et le normaliser afin de mieux le manipuler et en extraire le sens.

Modèles de Keras

Keras est une API d'apprentissage en profondeur écrite en Python, exécutée sur la plate-forme d'apprentissage automatique TensorFlow. Il a été développé dans le but de permettre une expérimentation rapide. Être capable de passer de l'idée au résultat le plus rapidement possible est la clé pour faire de bonnes recherches.

Matplotlib
Matplotlib est devenu une librairie puissante, compatible avec beaucoup de plateformes, et capable de générer des graphiques dans beaucoup de formats différents.

B - LES DONNÉES :

Nous devons constituer le dataset pour alimenter notre modèle.

Le **choix du format** a été rapide : un fichier json contenant 3 entrées :

- **input** : messages que l'utilisateur va envoyer au bot.
- **tag** : utilisé pour catégoriser les 'input' et les corréler à un type particulier de 'réponse'.
- **réponses** : quand le tag et l'input sont associés, permet de donner la réponse à l'utilisateur.

2 "bases" sont constituées:
- une base apprenant
- une base entreprise

Pour incrémenter les entrées, plusieurs essais ont été nécessaires pour aboutir aux solutions suivantes :

- pour les inputs, d'un mot unique nous sommes passés à des phrases complètes
- pour les réponses, nous avons décidé de garder une réponse unique pour que les informations soient toutes présentes, à l'exception des salutations pour lesquelles nous avons gardé différentes formules.

Avant :

```
"tags": "employabilité",
"input":["alternance","partenaires" ,"entreprise","stage", "pae", "CDI", "choix entreprise"],
"responses":[
    "oui, chaque apprennant a une alternance .",
    "3 périodes de PAE sont prévues en cours du cycle de formation intensif",
    "l'alternant peut être seul dans l'entreprise partenaire ou avec un ou deux collègues",
    "à l'issue de la période d'alternance, un CDI peut être proposé à l'alternant mais cela n'est pas une obligation",
    "lors du process de recrutement, une rencontre est organisée entre chaque apprenant et chaque entreprise."
```

Après :

```
{
    "tags": "emploi",
    "input":[
        "Le contrat en alternance est-il systématique?",
        "Des périodes de stages sont-elles prévues lors de la période intensive?",
        "Comment sont faites les attributions des entreprises ?",
        "Est-on seul dans son entreprise ?",
        "Sommes-nous à plusieurs dans les entreprises ?",
        "rencontrons-nous les entreprises avant le début de la formation ?",
        "Combien de fois, allons nous en entreprise avant le début de l'alternance?",
        "Le tuteur est-il obligatoirement un expert métier ?"
    ],
    "responses":[
        "Chaque apprenant aura une alternance trouvée par l'école. Trois périodes de présence en
entreprise de quelques jours sont prévues au cours du cycle de formation. Il y a parfois plusieurs
alternants par entreprise. Les tuteurs ne sont pas forcément des experts IA."
    ]
}
```

Voilà le format officiel de notre dataset, avec un tag, plusieurs questions en input, et une seule réponse en output. Après divers essais c'est la formule qui fonctionne le mieux (modifié)

Les données sont stockées dans la base de données MONGO Atlas qui est compatible avec les données de format Json. Elle est gratuite et en ligne, ce qui nous permettait à tous d'y avoir accès en permanence pour des essais ou des modifications. Visualisation de notre base de données :

```
_id: ObjectId("6093a263228a0b47333a5ffe")
tags: "bonjour"
responses: Array
  0: "Bonjour, ravi de vous renconter. En quoi puis-je vous aider ?"
  1: "Bonjour, comment puis-je vous aider ?"
  2: "Bonjour, avez-vous une question ?"
  3: "Bonjour, avez-vous besoin d'un renseignement ?"
```

Notre dataset se décompose ainsi :

```
nombre de questions :  262
nombre de tags :  23
questions/tag :  11.391304347826088
```

ci dessous notre liste de tags :

{0: 'actionnaires',
1: 'alternance',
2: 'au revoir',
3: 'bonjour',
4: 'certification',
5: 'chatbot',
6: 'confinement',
7: 'contact',
8: 'coût',
9: 'durée',
10: 'emploi',
11: 'entretiens',
12: 'evaluation',
13: 'formateurs',
14: 'inclusion',
15: 'localisation',
16: 'logement',
17: 'matériel',
18: 'presentation',
19: 'pédagogie',
20: 'recrutement',
21: 'teamwork',
22: 'technologie'}

C - LE PRÉTRAITEMENT DES DONNÉES

L'objectif est de "nettoyer" le texte afin de faciliter l'apprentissage.

Ci-dessous, quelques exemples de phrases avant le prétraitement des données.

```
je vous remercie
numéro de téléphone
quel est votre numéro de téléphone?
où puis-je vous appeler ?
comment puis-je vous contacter
je souhaite m'adresser à un humain
je veux parler à une personne
à qui puis-je m'adresser ?
```

Afin de nettoyer les données, nous utilisons le Process ci dessous :

- Suppression des accents
- Suppression des ponctuations
- Tokenization des données
- Appliquer un padding
- Encoder les outputs

résultat obtenu après suppression des accents, et des ponctuations

```
je vous remercie
numero de telephone
quel est votre numero de telephone
ou puisje vous appeler
comment puisje vous contacter
je souhaite madresser a un humain
je veux parler a une personne
a qui puisje madresser
```

Le tokenizer Tensorflow permet de **convertir les mots d'une phrase en nombres**. Chaque mot reçoit un identifiant et permet ainsi d'effectuer une grande variété de tâches NLP, de l'analyse des sentiments à la similitude des phrases.
Un Token unique est attribué à chaque mot distinct et le remplissage est effectué pour obtenir des arrays de même longueur afin de les envoyer à une couche RNN. les variables cibles sont également codées en valeurs décimales

résultat obtenu après le tokenizer

```
[17, 9, 44, 18, 120]
[18, 120, 4, 211]
[13, 9, 121, 44, 18, 212]
[17, 9, 44, 18, 28]
[13, 14, 122, 5, 213]
[6, 4, 5, 35, 214]
[6, 4, 5, 35, 215, 216]
[12, 39, 24, 217]
[40, 218, 219, 220]
```

résultat obtenu après le padding :

La fonction transforme les données en vecteurs de même longueur.

```
[  0   0   0   0   0   0   0   0   0  17   9  44  18 120]
[  0   0   0   0   0   0   0   0   0   0  18 120   4 211]
[  0   0   0   0   0   0   0   0  13   9 121  44  18 212]
[  0   0   0   0   0   0   0   0   0  17   9  44  18  28]
[  0   0   0   0   0   0   0   0   0  13  14 122   5 213]
[  0   0   0   0   0   0   0   0   0   6   4   5  35 214]
[  0   0   0   0   0   0   0   0   6   4   5  35 215 216]
[  0   0   0   0   0   0   0   0   0   0  12  39  24 217]
```

D - LONGUEUR D'INPUT, LONGUEUR OUTPUT ET VOCABULAIRE :

Cette partie, concerne la forme d'entrée (longueur des vecteurs) et la forme de sortie du réseau neuronal et nous permet de connaître le nombre de mot uniques. En output, le nombre de tags différents.

```
#input length
input_shape = x_train.shape[1]
print(input_shape)
#define vocabulary
vocabulary = len(tokenizer.word_index)
print("number of unique words : ",vocabulary)
#output length
output_length = le.classes_.shape[0]
print("output length: ",output_length)
```

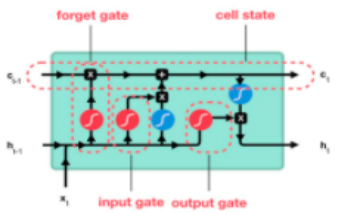
```
14
number of unique words :  478
output length:  23
```

E - RÉSEAU DE NEURONES : LSTM

*Un réseau de **neurones classique** permet de gérer les cas où **les expériences sont indépendantes les unes des autres**. Lorsque **les expériences sont sous la forme de séquences temporelles**, une nouvelle structure a été inventée : les réseaux de neurones récurrents. Cette nouvelle structure introduit un mécanisme de mémoire des entrées précédentes qui persiste dans les états internes du réseau et peut ainsi impacter toutes ses sorties futures. Le réseau de neurones Long Short Term Memory (LSTM) est l'un des plus connus.*

LSTM, qui signifie *Long Short-Term Memory*, est une cellule composée de trois "portes" : ce sont des zones de calculs qui régulent le flot d'informations (en réalisant des actions spécifiques). On a également deux types de sorties (nommées états).

- Forget gate (porte d'oubli)
- Input gate (porte d'entrée)
- Output gate (porte de sortie)
- Hidden state (état caché)
- Cell state (état de la cellule)



Cellule LSTM (crédit : image modifiée de Michaël Nguyen)

Le fonctionnement global d'un LSTM peut se résumer en 3 étapes :

1. Détecter les informations pertinentes venant du passé, piochées dans le cell state à travers la **forget gate** ;
2. Choisir, à partir de l'entrée courante, celles qui seront pertinentes *à long terme*, via l'**input gate**. Celles-ci seront ajoutées au cell state qui fait office de mémoire longue ;
3. Piocher dans le nouveau cell state les informations importantes *à court terme* pour générer le hidden state suivant à travers l'**output gate**.

Porte d'oubli (forget gate)

Cette porte décide de quelle information doit être conservée ou jetée : l'information de l'état caché précédent est concaténé à la donnée en entrée (par exemple le mot "des" vectorisé) puis on y applique la fonction sigmoïde afin de normaliser les valeurs entre 0 et 1. **Si la sortie de la sigmoïde est proche de 0**, cela signifie que l'on doit **oublier l'information** et si on est proche de 1 alors il faut **la mémoriser** pour la suite.

Porte d'entrée (input gate)

La porte d'entrée a pour rôle d'extraire l'information de la donnée courante (le mot "des" par exemple) : on va appliquer en parallèle une sigmoïde aux deux données concaténées (cf porte précédente) et une tanh.

- Sigmoïde va renvoyer un vecteur pour lequel une coordonnée proche de 0 signifie que la coordonnée en position équivalente dans le vecteur concaténé n'est pas importante. A l'inverse, une coordonnée proche de 1 sera jugée "importante" (i.e. **utile pour la prédiction** que cherche à faire le LSTM).
- Tanh va simplement normaliser les valeurs (les écraser) entre -1 et 1 pour éviter les problèmes de surcharge de l'ordinateur en calculs.
- Le produit des deux permettra donc de ne garder que les informations importantes, les autres étant quasiment toutes remplacées par 0.

Etat de la cellule (cell state)

On parle de l'état de la cellule avant d'aborder la dernière porte (porte de sortie), car la valeur calculée ici est utilisée dedans. L'état de la cellule se calcule assez simplement à partir de la porte d'oubli et de la porte d'entrée : d'abord on multiplie coordonnée à coordonnée la sortie de l'oubli avec l'ancien état de la cellule. Cela permet d'**oublier certaines informations** de l'état précédent qui ne servent pas pour la nouvelle prédiction à faire. Ensuite, on additionne le tout (coordonnée à coordonnée) avec la sortie de la porte d'entrée, ce qui permet d'enregistrer dans l'état

de la cellule ce que le LSTM (parmi les entrées et l'état caché précédent) a **jugé pertinent**.

Porte de sortie (output gate)

Dernière étape : la porte de sortie doit décider de quel sera le prochain état caché, qui contient des informations sur les entrées précédentes du réseau et sert aux prédictions.

Pour ce faire, le nouvel état de la cellule calculé juste avant est normalisé entre -1 et 1 grâce à tanh. Le vecteur concaténé de l'entrée courante avec l'état caché précédent passe, pour sa part, dans une fonction sigmoïde dont le but est de décider des **informations à conserver** (proche de 0 signifie que l'on oublie, et proche de 1 que l'on va conserver cette coordonnée de l'état de la cellule).

Détail de notre modèle :

```
Model: "model_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, 14)]              0
_________________________________________________________________
embedding_4 (Embedding)      (None, 14, 10)            4790
_________________________________________________________________
lstm_4 (LSTM)                (None, 14, 10)            840
_________________________________________________________________
flatten_4 (Flatten)          (None, 140)               0
_________________________________________________________________
dense_4 (Dense)              (None, 23)                3243
=================================================================
Total params: 8,873
Trainable params: 8,873
Non-trainable params: 0
_________________________________________________________________
```

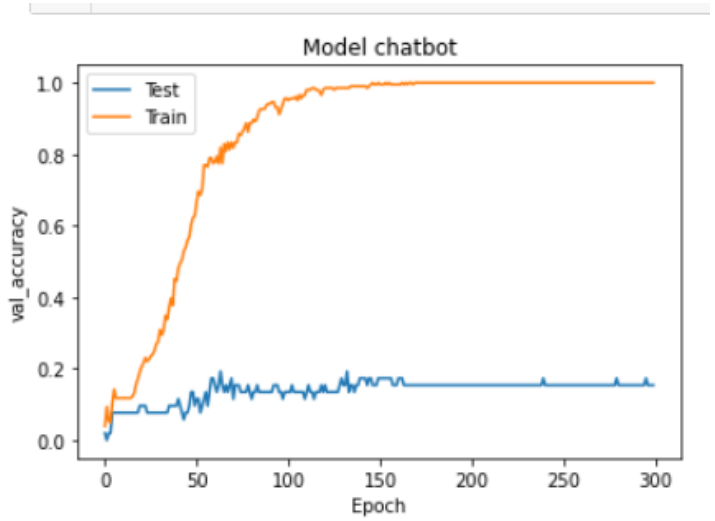
Le réseau se compose :
d'une couche d'intégration (embedding layer) qui est l'une des choses les plus puissantes en NLP.,
c'est celle qui donne un vecteur correspondant pour chaque mot de l'ensemble de données
les sorties de l'embedding layer sont
les inputs de la couche récurrente lstm gate.
ensuite, l' output est aplatie (flatten) et une couche dense régulière (dense layer) est utilisée avec une fonction d'activation softmax.

F - ANALYSE DU MODÈLE :

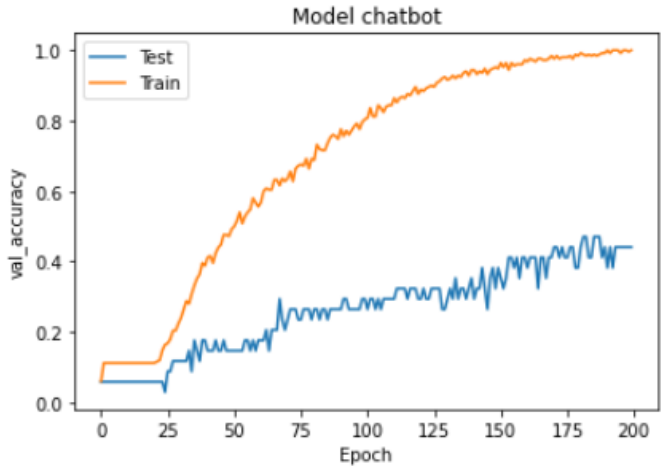
Pour analyser le modèle, on calcule la valeur de la précision sur l'échantillon de test. Nous avons créé un dataset spécifique au test.

Dans notre cas, le dataset est le facteur influent .

Avec un dataset peu complété (**127 questions**) nous obtenons les courbes ci dessous avec une précision sur les données de validation de seulement 15% :

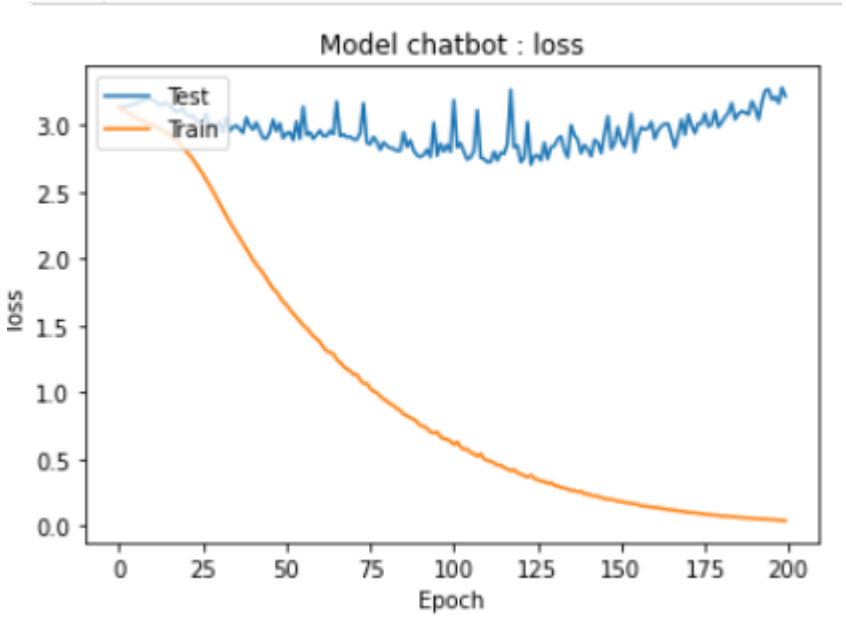


En augmentant le dataset à **250 questions** nous arrivons au résultat ci dessous. La précision a été démultipliée (jusqu'à environ 44%).



Epoch 200/200
8/8 [==============================] - 0s 15ms/step - loss: 0.1001 - accuracy: 1.0000 - val_loss: 3.0755 - val_accuracy: 0.4412

La précision de nos résultats est presque multipliée par X3 pour une augmentation de notre jeu de données, passage de 127 à 250 questions.



G- TESTS :

Au préalable, notre dataset était plutôt susceptible .

```
tag:  contact
question : je refuse de parler à un robot
tag:  au revoir
```

et au final , la tendance s'est inversée

```
question : Je refuse de parler à un robot
tag:  contact
```

EXemple plus complet :

```
question : Bonjour
tag:  bonjour
question : etes-vous un chatbot
tag:  chatbot
question : qui puis-je appeler?
tag:  contact
question : combien ça coute ?
tag:  cout
question : ou puis-je me loger
tag:  logement
question : ou est l'école?
tag:  recrutement
question : qu'estce que Simplon?
tag:  presentation
question : dois-je savoir coder avant?
tag:  recrutement
question : combien coute la formation?
tag:  cout
question : je suis neuroatypique
tag:  inclusion
question : comment se passent les cours?
tag:  cout
question : comment se deroulent la classe?
tag:  pédagogie
question : le travail se fait-il en groupe?
tag:  teamwork
question : y a t il une évaluation?
tag:  logement
question : est-ce qu'on a des notes?
tag:  evaluation
```

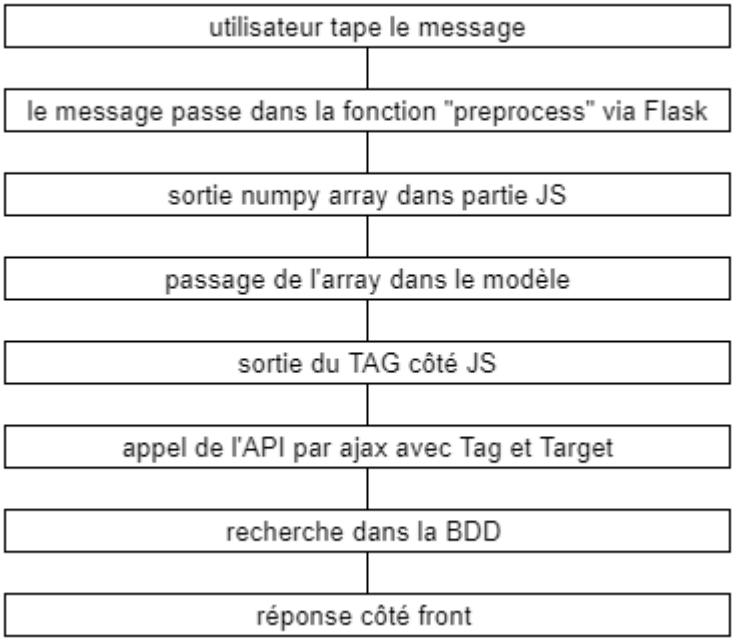
avec les mots magiques :

```
question : les cours sont ils à distance?
tag:  confinement
question : comment se passent les cours?
tag:  pédagogie
question : peut-on travailler en groupe?
tag:  teamwork
question : est ce que je dois trouver seul mon alternance?
tag:  alternance
question : Qu'est ce que Simplon?
tag:  presentation
question : y a t il un concours d'entrée?
tag:  entretiens
```

III - ARCHITECTURE DU MODÈLE :

```
PS C:\Users\utilisateur\Google Drive\microsoft_ia\Google Drive\projets\ia\ChatBot_Microsoft_Simplon_Brest> tree ./
Structure du dossier pour le volume Windows
Le numéro de série du volume est 000000B9 0EAC:DE3F
C:\USERS\UTILISATEUR\GOOGLE DRIVE\MICROSOFT_IA\GOOGLE DRIVE\PROJETS\IA\CHATBOT_MICROSOFT_SIMPLON_BREST
├───Api
│   ├───dataset
│   └───__pycache__
├───model
├───static
│   ├───css
│   ├───images
│   └───js
├───templates
└───__pycache__
```

IV - CHOIX TECHNIQUES :



API :

Pour la mise en place de l'API, nous avons fait le choix de FastAPI. Il s'agit d'un framework utilisé sous python qui est assez simple d'utilisation.
L'installation se fait très simplement via l'invit de command.
Dans un premier temps on va installer FastAPI:

```
$ pip install fastapi
```

Dans un second temps, on va installer un serveur ASGI. Ca sera Uvicorn:

```
$ pip install uvicorn
```

Trois choses vont être "nécessaire" ensuite pour le bon fonctionnement:

- Nous allons stocker l'appel à FastAPI dans une variable pour simplifier son utilisation dans le code.

```
app = FastAPI()
```

- Un problème est souvent rencontré également au niveau des navigateurs. Le CORS ou Cross-Origin Resource Sharing est un système qui permet de récupérer des ressources depuis plusieurs origines (des ports différents dans notre cas). Hors les navigateurs refusent souvent les accès lorsque le serveur et le client ont des ports différents. Nous avons donc ajouté cette autorisation au niveau de l'API.

```
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

- Enfin nous assignons un port spécifique à notre API.

```
if __name__ == '__main__':
    uvicorn.run("api:app", port=8000, reload=True)
```

FastAPI ne servant ici qu'à récupérer des réponses dans le corpus, elle n'aura qu'un seul endpoint :

```
@app.get("/api/target/{target}/{tags}")
async def reponse(target:str, tags:str):
    return co.get_response(target, tags)
```

Les réponses pour les apprenants et les entreprises ayant quelques petites différences, nous avons fait le choix de créer 2 collections distinctes dans notre base de données. Ainsi nous devons lui renseigner 2 arguments :

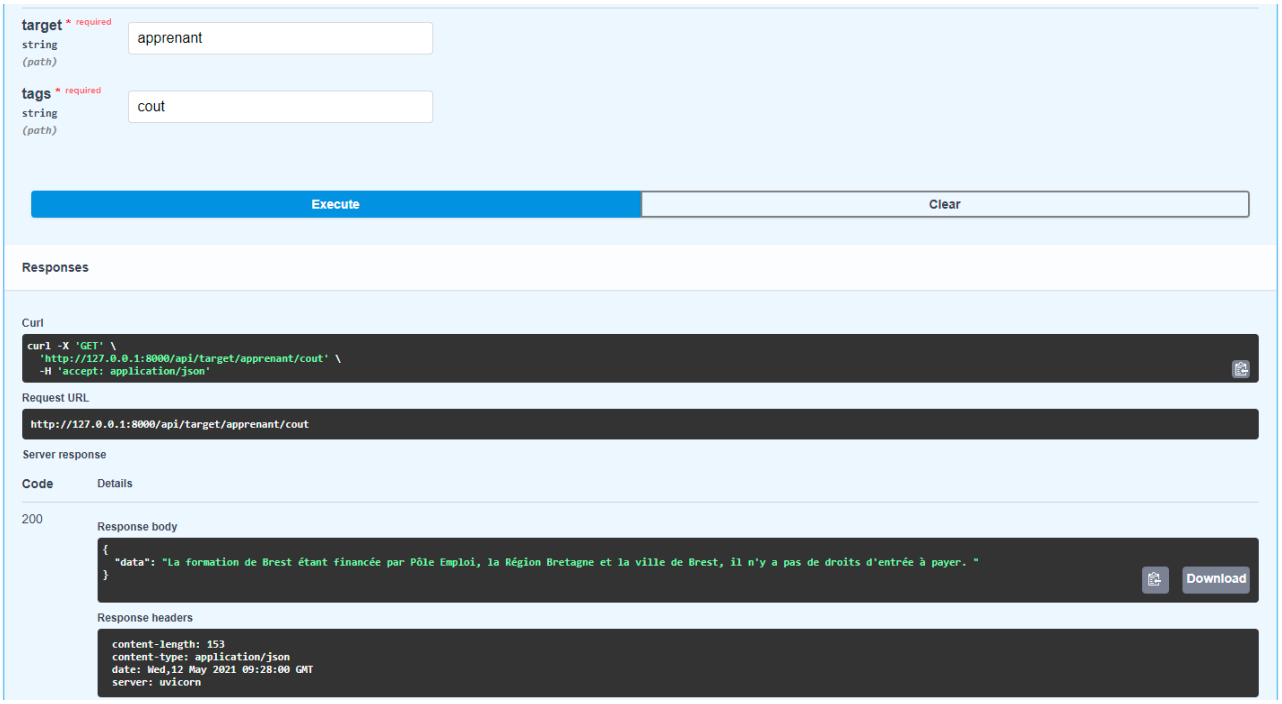
- la target (apprenant ou entreprise)
- le tag (bonjour, cout, emploi, duree, …)

Ainsi l'endpoint peut faire appel à la fonction qui permet d'aller chercher dans la BDD.

```python
    @classmethod
    def get_response(cls, target, tags):
        cls.open_db()
        if target == "apprenant":
            cls.data = list(cls.chat_app.find({"tags": tags}, cls.filter))
        else:
            cls.data = list(cls.chat_ent.find({"tags": tags}, cls.filter))

        cls.close_db()
        print(random.choice(cls.data[0]["responses"]))
        return {"data": random.choice(cls.data[0]["responses"])}
```

Enfin pour pouvoir tester les appels, nous avons utilisé un outils de FastAPI qui permet de vérifier si l'url renseigné renvoie bien une réponse (code 200 + réponse) ou une erreur (code erreur)



PARTIE WEB :

Pour le Front nous sommes partis sur Flask qui, comme FastAPI, est aussi un microframework utilisé sous python afin de faire du développement WEB. Flask dépend du moteur de modèle Jinja et de Werkzeug WSGI toolkit pour gérer les erreurs.

```
$ pip install Flask
```

Installer Flask installe aussi tout ce qu'il faut pour son bon fonctionnement.

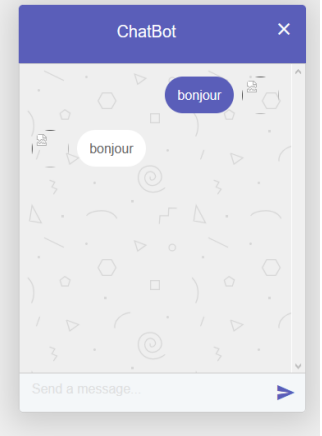
FLASK :

Flask va nous servir à 2 choses :

- Faire passer le message écrit par l'utilisateur dans une fonction de preprocessing afin de récupérer un numpy.array (représentation de la phrase sous forme de tableau).
    - bonjour => [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 195]
- Faire appel à l'API en lui renseignant une target et un tag pour aller récupérer une réponse dans la base de données.

JAVASCRIPT :

La fenêtre de chatbot en elle même est codée en JavaScript, et plus particulièrement en utilisant la bibliothèque JavaScript Jquery



Le code original du chatbot est issu d'un site de code en ligne appelé codepen.

Ce bot répond à ce qu'écrit l'utilisateur comme un "perroquet". Il a fallu adapter le code une première fois pour le faire marcher tel quel.

La communication avec l'API se fait grâce à des requêtes AJAX qui sont nécessaires afin de pouvoir faire fonctionner le chatbot sans avoir à recharger la page

Ouverture du ChatBot:

Il nous a été demandé que le chatbot s'ouvre quand on clique sur un bouton en bas à droite de la page Web

Ici le code qui permet l'ouverture de la fenêtre de chat:

```
$("#chat-circle").click(function() {
  $("#chat-circle").toggle('scale');
  $(".chat-box").toggle('scale');
  var intro = 'Bonjour, je m\'appel Tayouste et je suis là pour répondre à vos questions
  setTimeout(function() {
    generate_bot_message(intro, 'user');
  }, 1000)
})

$(".chat-box-toggle").click(function() {
  $("#chat-circle").toggle('scale');
  $(".chat-box").toggle('scale');
})
```

La première fonction ouvre la fenêtre du Chat et envoie le message d'accueil du Bot.
La deuxième fonction ferme la fenêtre du Chat.

La fonction qui affiche le message de l'utilisateur se présente comme ceci :

```
  generate_message(msg, 'self'); // messages de l'utilisateur
})

function generate_message(msg, type) {
  INDEX++;
  var str="";
  str += "<div id='cm-msg-"+INDEX+"' class=\"chat-msg "+type+"\">";
  str += "          <span class=\"msg-avatar\">";
  str += "            <img src=\"static/images\/humain.png\">";
  str += "          <\/span>";
  str += "          <div class=\"cm-msg-text\">";
  str += msg;
  str += "          <\/div>";
  str += "        <\/div>";
  $(".chat-logs").append(str);
  $("#cm-msg-"+INDEX).hide().fadeIn(300);
  if(type == 'self'){
   $("#chat-input").val('');
  }
  $(".chat-logs").stop().animate({ scrollTop: $(".chat-logs")[0].scrollHeight}, 1000);
}
```

La variable "str" permet de rajouter du HTML aux templates et de faire ainsi défiler les messages dans la fenêtre du ChatBot, une animation est ajoutée afin de fluidifier ce défilement.

La fonction de réponse du bot est fortement inspirée de celle-ci, quelques changements ont été effectués, notamment au niveau de l'image de profil.

La fonction chatBot qui utilise AJAX:

La communication avec l'API se fait à cet endroit
cette fonction prend en argument le message écrit par l'utilisateur

```
// fonction qui récupère le message et charge le modèle puis fait la prediction
function chatBot(message) {
let inputMessage = message;
userType = 'entreprise'

if (inputMessage.lenght !== 0) { // je vérifie que le message n'est pas vide

    async function loadModel(tags) {  // fonction de chargement du modèle

        tag = tf.tensor2d(tags, [1,14]) //transformation du tag en tensor
        const modelURL = 'static/model.json';   // url de chargement du modèle

        const model = await tf.loadLayersModel(modelURL);  //chargement du modèle
        var prediction = await model.predict(tag);
        var label = await prediction.argMax(-1).data();

        var tagsIndex = {0: 'teamwork',
        1: 'actionnaires',
        2: 'bonjour',
```

La fonction loadModel prend en argument la phrase preprocessée par l'API et la passe dans le modèle afin de faire la prédiction, ensuite elle choisit le bon tag dans le dictionnaire des tags.
Puis elle appelle la fonction runApi() et stock la réponse dans la variable "message" et à la toute fin elle génère la réponse du robot avec la fonction generate_bot_message()

```
        22: 'localisation',
        23: 'participation'}

        var tagValue = tagsIndex[label]

    var message = runApi(tagValue, userType)
    setTimeout(function() {
      generate_bot_message(message, 'user');
    }, 1000)
```

La requête AJAX ci-dessous envoie le message texte à l'API, qui le retourne transformé, et en cas de succès la fonction loadModel() est lancée.

```
$.ajax({
    url:"/chat",  // url vers l'application
    data:{'message':inputMessage},
    type:"POST",
    dataType:"json",
    success : function(tags) {
        loadModel(tags)
    }
});
```

à la fin de la fonction loadModel(), la fonction runApi() est lancée, elle envoie avec une requête AJAX: input qui est le tag et user qui est “apprenant” ou “entreprise”. En retour, la fonction récupère le message correspondant au tag.

```
function runApi(input, user){
var jqXHR = $.ajax({
    type: "POST",
    url: "/api",
    async: false,
    data: { mydata: input,
    mydata2: user}
});
return jqXHR.responseText;
}
```

Résultat:

V - SOLUTION DE DÉPLOIEMENT, COÛT ET SES AVANTAGES :

La solution est avantageuse pour le client car les calculs se font côté client et la solution est moins onéreuse pour le client.

TEAM DEPLOIEMENT

20 Rue Cuirassé

29 200 BREST

mail : CPSERVICES@ORANGE.FR

Tel : 06 12 34 56 78

DEVIS N°200178

devis valable 1 mois

ISEN

20 rue Cuirassé

29200 BREST

France

Date	CODE CLIENT	N° tva	TYPE
01/05/2021	411116 N	FR 12 345 678 91	prestation

DESIGNATION	nbr de jours	Prix forfaitaire	Total HT
Déploiement du chatbot TAYOUSTE Déplacement	[illegible]	[illegible]	11 550,00€

Total Prix HT	11 550,00 €
TVA 20 %	2 310,00 €
Total Prix TTC	13 860,00 €

CONDITIONS DE PAIEMENT

50 % à l'acceptation du devis

50% à la livraison

REFERENCES BANCAIRES :

Bénéficiaire: BANQUE de Brest

Place liberté - 29200 BREST

COMPTE : 12345 12345 11111111111 05

IBAN : FR76 1234 5123 4511111111111 05

SWIFT : CCBPFRPXXXX

Le devis n'engage le client qu'à partir du moment où il a exprimé sa volonté de faire exécuter les travaux, par une signature au bas du devis « B

Pénalité en délai : 1,20 %

signature du client :

date :

10 000 € - SIRET : FR 12 345 678 90 – APE 123 -

Date	CODE CLIENT	N° tva		TYPE
03/05/2021	4111ISEN	FR 12 345 678 91		prestation
DESIGNATION		nbe de jours	Prix forfaitaire	Total HT
Déploiement du chatbot TAYOUSTE				
		33	350,00€	11 550,00€
Déplacement				
		Total PriX HT		11 550,00 €
		TVA 20 %		*2 310,00 €*
		Total PriX TTC		13 860,00 €

VI - AXE D'AMÉLIORATION.

Le premier est certainement le DATASET.

Nous avons pu le constater lors de nombreux tests. Hormis la constitution du dataset qui est un élément fondamental, le nombre de données est essentiel. Ce fait est mis en évidence dans les graphiques précédents. Lorsque nous doublons le nombre de données (de 127 à 250 questions), les résultats de précisions sont presque multipliés X3. Nous n'aurions certes pas obtenu la même corrélation à chaque augmentation du dataset, mais on peut penser que plus notre dataset aurait été fourni, plus la précision aurait augmenté.

Nous nous sommes, également, rendu compte que la façon dont on posait les questions avait une grosse influence sur les résultats. S'il n'y a qu'une personne

qui crée le dataset, le jeu de test et d'entraînement, le modèle s’adapte à sa logique et est plus fiable avec cette personne.

Le deuxième est peut-être le choix du modèle.

Des recherches sur les architectures RNN pour les tâches de traduction ont démontré que le sens de lecture classique des séquences n’est pas optimal. En effet, ils ont montré empiriquement que lire les séquences à l’envers améliore significativement la performance du modèle dans ce genre de problèmes. La combinaison des deux sens de lecture (bidirectional RNN) donne des résultats encore meilleurs. Ce dernier résultat est prévisible, vu qu’une représentation bidirectionnelle permet au modèle d’accorder autant d’importance aux premiers caractères de la séquence qu’aux derniers. La version unidirectionnelle, quant à elle, « dilue » mécaniquement les premiers caractères, même si ce phénomène est moins prononcé dans un LSTM que dans un RNN simple.