



UNIVERSITÉ
CÔTE D'AZUR

Rapport du projet

Traitement automatique du texte en IA

Création des caractères et communication

Stefan Am Ende

Matthis Kuhl

UNIVERSITÉ CÔTE D'AZUR

DS4H - MASTER INFORMATIQUE

27 décembre 2021

Table des matières

1	Contexte du projet et description des tâches	2
2	Bases de données utilisées	2
2.1	Wikidata	2
2.2	Word2Vec	2
2.3	Jeu de données WebNLG Challenge 2020	2
3	Implémentations	3
3.1	Création des caractères	3
3.2	Entraînement du modèle T5	4
3.3	Génération des descriptions des personnes avec le T5	4
3.4	Implémentation du chat entre les IAs	5
3.5	Implémentation du chat interactif	5
4	Évaluation et analyse d’erreurs	7
4.1	Descriptions des personnes	7
4.2	Chat entre les IAs	8
4.3	Chat interactif	9
4.4	Idées non compris dans le projet final	11
5	Conclusion et améliorations possibles	11

1 Contexte du projet et description des tâches

L'objectif de ce projet était de créer des différentes personnes avec des propriétés personnelles. Celles-ci peuvent être par exemple des intérêts, des loisirs ou leur métier. En provenance de ces propriétés personnelles on voulait générer des descriptions sur les personnages d'une manière automatisée. D'ailleurs c'était le but de créer une interaction entre eux d'une façon comme un chat en ligne et ensuite l'étendre pour qu'un utilisateur humain soit capable d'y participer. Pour finir on a essayé d'extraire les informations écrites par l'utilisateur pour connaître ses propriétés personnelles, comme celles des personnages générées, et aussi générer une description sur la personnes avec.

2 Bases de données utilisées

2.1 Wikidata

Wikidata est une gigantesque base de donnée relié à travers un système compréhensible pour l'ordinateur. Pour avoir accès à *Wikidata* on utilise *SPARQLWrapper*. On y accède via internet a l'adresse <https://query.wikidata.org/sparql>. Cela nous permet plus tard de chercher dans *Wikidata* via des requêtes, par exemple pour générer les propriétés des bots.

2.2 Word2Vec

Comme notre projet est très global puisqu'il se repose sur *Wikidata*, un modèle spécialisé ne nous sert à rien. Donc il nous faut un grand *Word2Vec* avec une grande variété de sujets pour être réellement utile. C'est pourquoi on a cherché de le faire sur *Wikipedia dump file* comme base de donnée. Mais après quelque lecture du projet et explications dans cette direction on s'est rendu compte que le faire nous même nous demanderait plusieurs jours et beaucoup de mémoire d'ordinateur. Puis cherchant de modèle déjà pré-entraîné, on a trouvé un fichier *Word2Vec* sur Wikipedia 2017 qu'on utilise par la suite. (voir aussi [1])

2.3 Jeu de données WebNLG Challenge 2020

Pour la génération des descriptions un *modèle T5* a été utilisé. Ce modèle est déjà pré-entraîné par Google pour servir à plusieurs tâches différentes mais doit encore être finement entraîné pour un tâche spécifique, donc dans notre cas la génération des phrases à partir des mots clés. Dans ce but on a utilisé le jeu de données de la *WebNLG Challenge 2020* [2]. Ce jeu de données consiste à plusieurs triples de mots et leurs phrases correspondantes (voir figure 1). Grâce à ça on apprend au modèle quelles phrases sont formées par quels mots. L'entraînement est expliqué dans la section 3.2.

```

<entry category="University" eid="Id1" shape="(X (X))" shape_type="NA" size="1">
  <originaltriple>
    <otriple>1_Decembrie_1918_University,_Alba_Iulia | city | Alba_Iulia</otriple>
  </originaltriple>
  <modifiedtriple>
    <mtriple>1_Decembrie_1918_University | city | Alba_Iulia</mtriple>
  </modifiedtriple>
  <lex comment="good" lid="Id1">The "1 Decembrie 1918 University" is in the city Alba Iulia.</lex>
  <lex comment="good" lid="Id2">The city of Alba Iulia has a University called "1 Decembrie 1918 University".</lex>
  <lex comment="good" lid="Id3">The 1 Decembrie 1918 University is located in the city of Alba Iulia.</lex>
</entry>

```

FIGURE 1 – Un exemple pour un triple du jeu de données

3 Implémentations

3.1 Création des caractères

« Caractères » ici veut dire ce que décrit une personne. Pour décrire les personnes ou dans ce cas les bots nous avons choisis ces catégories des caractéristiques suivantes : age, name, book, sport, music, job, film, game, food

Pour créer l'âge c'est assez facile, on utilise un simple random entre 18 et 80. (On exclue certains âges pour simplifier et rester réaliste. Si 5 personnes de 110 ans et 2 bébés à 2 ans t'écrivent via un chat cela serait très bizarre).

Mais pour les autres catégories comment faire ? Bien sûr on pourrait écrire soit même une liste de noms etc. et utiliser celle-ci mais pourquoi ne pas utiliser ce qui est déjà fait. C'est la raison pourquoi on a choisis ici de faire des requêtes avec *Wikidata* comme décrit dans le chapitre 2.1. On définit une fonction `research(Qid)` qui à partir d'une Qid renvoie la liste d'items appartenant au thème de Qid. La fonction `research(Quid)` fait une requête qui recherche les items `wdt:P31` de `Quid`, ce qui sont les items reliés à la Qid. Dans la requête on spécifie le langage de la requête des `itemlabels` :

```

SERVICE wikibase:label {
  bd:serviceParam
    wikibase:language "[AUTO_LANGUAGE],en".
}

```

On définit quand même une limite a 1000 items de recherche avec `LIMIT 1000`. Par la suite on transforme le résultat en fichier `json`. On parcourt les `itemlabels` et les sauvegarde dans une liste qu'on renvoie. On s'est rendu compte que si `itemlabel` n'est pas décrit, il nous renvoie la Qid de l'item, c'est pourquoi on exclut les « Q... ». Et voilà, on a grâce à un Qid la liste des `itemLabel`. On a mis les Qids pour nos sujets dans une liste comme suit :

```

listofidqHuman = [ 'Q22808320', 'Q571', 'Q31629', 'Q188451',
  'Q192581', 'Q11424', 'Q11410', 'Q2095', 'Q3624078' ].

```

Maintenant pour chaque personne on peut piocher un nom de la liste des noms générée grâce à la méthode `research('Q22808320')`. C'est ainsi que nos bots ont ses propres caractéristiques humaines.

3.2 Entraînement du modèle T5

Pour finement entraîner le T5 pour le tâche de la génération des phrases à partir des mots clés on a suivi un tutoriel en ligne [3]. Avant de pouvoir entraîner le modèle il faut d'abord pré-traiter le jeu de données. Ça veut dire, on itère à travers la structure des fichiers *xml* pour ensuite sauvegarder tous les exemple dans un seul fichier *csv* (voir figure 2). Après on utilise ce fichier pour l'entraînement du modèle. Veuillez noter que l'algorithme pour l'entraînement n'était pas notre travail et c'est pourquoi on ne va pas dans les détails ici (voir aussi [4]).

prefix	input_text	target_text
webNLG	Bionico dishVariation Cottage_cheese && Bionico course Dessert	Bionico is a dish served for dessert that can be varied by using cottage cheese.
webNLG	Bionico dishVariation Cottage_cheese && Bionico course Dessert	The desert Bionico, can be varied using cottage cheese
webNLG	Bionico region Jalisco && Bionico ingredient Condensed_milk	The ingredient condensed milk is used in Bionico and this derives from the region of Jalisco.
webNLG	Bionico region Jalisco && Bionico ingredient Condensed_milk	Condensed milk is used in bionico which comes from the region of Jalisco.
webNLG	Bionico region Jalisco && Bionico ingredient Condensed_milk	Bionico, made using condensed milk, is found in the region of Jalisco.
webNLG	Catalonia leaderName Carles_Puigdemont && Arros_negre region Catalonia	Arros negre is found in the region of Catalonia and the leader of the country is Carles Puigdemont.
webNLG	Catalonia leaderName Carles_Puigdemont && Arros_negre region Catalonia	Arros negre originates from Catalonia (led by Carles Puigdemont)
webNLG	Catalonia leaderName Carles_Puigdemont && Arros_negre region Catalonia	Arros negre is a region of Catalonia and Carles Puigdemont is it's leader.
webNLG	Celery genus Apium && Bakso ingredient Celery	Celery is a member of the genus Apium and is an ingredient of Bakso.
webNLG	Celery genus Apium && Bakso ingredient Celery	Bakso contain celery, which is from the genus apium.

FIGURE 2 – Extrait du fichier *csv* avec des exemple pour l'entraînement

Comme le modèle est maintenant entraîné pour la tâche on peut lui donner plusieurs triples afin de faire générer une phrase à partir de ces mots. Un exemple est représenté dans la figure 3.

```
Entrée [36]: generate('Université Côte d'Azur | city | Nice && Nice | city | France', model, tokenizer)
Out[36]: "The city of Nice in France is the location of the Université Côte d'Azur."
```

FIGURE 3 – Un exemple pour la génération d'un texte à partir des mots clés

3.3 Génération des descriptions des personnes avec le T5

Comme on a maintenant créé des personnages avec des propriétés personnelles on peut les donner au modèle de génération afin d'obtenir une phrase qui décrit une propriété d'une personne. Pour obtenir une description on lance alors plusieurs générations avec différentes propriétés. La première phrase parle du nom, l'âge et le pays d'une personne (voir figure 4). Ensuite on forme une phrase qui décrit le son métier et après des phrases concernant son livre préféré, son sport préféré, etc.

```
string=self.name+' | age | '+str(self.age)+' && '+self.name+' | country | '+self.country
print(generate(string, model, tokenizer))
```

FIGURE 4 – Implémentation de la génération d'une phrase qui décrit le nom, l'âge et le pays d'un personnage

3.4 Implémentation du chat entre les IAs

Comme le modèle entraîné dans le chapitre précédent a seulement vu des phrases qui sont écrites dans la troisième personne on ne peut pas générer des phrases dans la première personne avec. En plus les descriptions obtenues ne sont pas assez bien pour construire une discussion entière qui fait du sens (voir chapitre 4.1). C'est pourquoi on s'est décidé d'implémenter le mécanisme du chat nous même.

Dans la première partie de la discussion chacun des caractères doit se présenter en disant son nom, son âge et son pays. Pour chaque partie de la phrase plusieurs manières possibles sont implémentées pour dire la même chose dont une est choisie aléatoirement. Avec toutes ces combinaisons il y a pas mal de différentes phrases pour se présenter. Par exemple « *Hello, my name is Alexander Campbell, I'm 34 years old and I am from Israel.* »

Après la présentation c'est à une personne de commencer un sujet. La méthode *startTopic()* est appelée. Dans cette méthode on choisi aléatoirement un sujet parmi musique, métier, nourriture, jeux, sport, films et livres. Seuls les sujets desquels on n'a pas parlé encore sont concernés. Pour chaque sujet il existe une méthode *startTopicX()* où *X* est un des sujets mentionnés. Pour donc commencer un sujet la méthode correspondante est appelée. Là dedans on commence par exprimer sa propriété personnelle concernant le sujet sélectionné en appelant la méthode *tellYourX()*. Ensuite on demande les autres concernant leur valeur de la propriété de laquelle on vient de parler. Dans chaque de ces méthodes ils existent toujours plusieurs possibilités de dire la même chose pour avoir plus de variabilité dans la discussion. Si l'IA choisit par exemple le sujet « *jeux* » une phrase pourrait être : « *I like to play Fault Line. Are there games that you like to play ?* »

Maintenant c'est aux autres de comprendre la phrase et d'y répondre correctement. Pour ceci la méthode *respond(message)* est appelée qui reçoit comme paramètre le message de la personne qui avait commencé le sujet. Pour la compréhension on commence par distinguer les parties du message en phrases et en questions à l'aide des signes de ponctuation. Après on devine le sujet de chaque phrase et de chaque question en utilisant *gensim* avec *word2vec*. La distance entre chaque mot de la phrase et chaque sujet est calculée. Le sujet avec la distance la plus petite à un mot dans la phrase est considéré être le bon. Maintenant l'IA connaît le sujet de la question et peut y répondre en appelant une des méthodes *tellYourX()*. Cette boucle continue jusqu'à ce qu'il n'y a plus de sujets ou quand la nombre de sujets donnée comme argument est atteinte.

3.5 Implémentation du chat interactif

Dans ce chapitre on étend les fonctionnalités décrites dans le chapitre précédent. On montre comment on a réalisé l'interaction entre un utilisateur et les IAs, comment ils peuvent communiquer et comment on extrait des informations sur l'utilisateur pour ensuite générer une description de lui.

Dès qu'on lance le chat interactif l'utilisateur est invité d'entrer son nom afin de pouvoir participer au chat. Ensuite les autres personnages se présentent comme auparavant.

Après une première personne va demander l'âge de l'utilisateur et une deuxième son pays. Ces informations qui ne font pas partie d'un des sujet implémentés sont nécessaires pour la première phrase de la description plus tard. Une personne va maintenant commencer un sujet comme dans le chat sans interaction et l'utilisateur est invité d'écrire une réponse. Puis les IAs reçoivent le message de l'utilisateur, essaient de comprendre le sujet et y répondent par rapport au sujet deviné. Chaque IA gère une propre liste avec les sujets sur lesquels elle a déjà parlé. Comme l'IA qui a commencé le dialogue a déjà parlé du sujet concerné elle ne va pas répondre à sa propre question. En général une IA ne parle pas deux fois sur le même sujet. Une autre possibilité du côté de l'utilisateur est de répondre à la question et de poser une nouvelle question aux autres. Maintenant les IAs qui n'ont rien dit encore vont répondre aux deux questions posées et celle qui a posé la première question ne va que répondre à la deuxième (voir figure 5). Le mécanisme est exactement le même comme décrit au chapitre 3.4 parce qu'en vrai l'IA devine tous les sujets des phrases et tous les sujets des questions et les sauvegarde dans des listes séparées. S'il y a plusieurs phrases ou plusieurs questions l'IA en choisit une aléatoirement pour y répondre. Dans ce cas là il est possible que les différentes listes des IA avec les sujets traités peuvent se distinguer ce qui influence si une IA va répondre à une question plus tard ou pas.

```
Edward: I usually listen to ciranda. What is your favourite music?
Stefan: My favourite music is rap. What is your favourite food?
Frederick: My favourite dish is United States military chocolate. I like Gregorian chant.
Alexander: My favourite dish is asinan Bogor. My favourite music genre is folk metal.
Barbu: My favourite dish is Chiroti. I like listening to empfindsamkeit.
Edward: I really enjoy Manisa kebab.
```

FIGURE 5 – Un exemple où des IA répondent à deux questions sauf celle qui a posé la première.

Si l'utilisateur pose une question que les IAs ne comprennent pas un personnage va exprimer qu'elle ne l'a pas compris et lui expliquer les sujet sur lesquels on peut discuter. Dans le cas où aucune IA peut répondre, soit parce qu'elles ne comprennent pas le message soit qu'elles aient déjà parlé sur le sujet concerné, un caractère va commencer un nouveau sujet s'il y en a encore (voir figure 6).

```
Stefan: Lorem ipsum.
Edward: Sorry, I didn't get your question. We can talk about sport, music, work, games, food, films and books.
Edward: I like films like Long Live the Lady!. Which films do watch?
```

FIGURE 6 – Le comportement d'une IA quand elle ne comprend rien.

Comme les sports, les plats, les métiers, etc. qu'on récupère de *Wikidata* sont souvent très exotique on peut aussi demander à une IA d'expliquer par exemple son sport préféré. Le mécanisme est plutôt simple : Pour chaque question demandée par l'utilisateur on vérifie d'abord s'il y a un caractère qui possède une valeur d'une propriété qui se trouve dans la question. Si c'est le cas on récupère la description de cette valeur de *Wikidata* et forme une phrase avec à l'aide de la méthode *answerSpecificQuestion()* (voir figure 7).

```

Robert: I work as a Museum Technician. What do you do?
Stefan: What is a Museum Technician?
Robert: The job Museum Technician can be described as position or role in a museum environment concerned with the technical and scientific aspects of museum work, including but not limited to conservation.

```

FIGURE 7 – La réponse à une question spécifique.

Pour pouvoir sauvegarder les caractéristiques de l'utilisateur on lui crée une instance de la classe **Person** sans propriétés qu'on nomme **chater**. Au début du chat on récupère **chater.name**, **chater.age** et **chater.country**. Par la suite de la discussion on récupère le sujet de chaque phrase. Pour cela nous utilisons notre *Word2Vec* pré-entraîné avec lequel on calcule la distance des mots de la phrase à nos catégories. Le sujet est celui qui a la plus petite distance aux mots de la phrase.

Par la suite, pour récupérer le mot clé dans une phrase et pouvoir le placer dans **chater.<sujet>** on parcourt cette phrase et on cherche le mot qui est le plus proche à ce sujet sans être trop proche.

Un « bye » finit le chat et les propriétés extraites de l'utilisateur avec une description générée sont affichées.

4 Évaluation et analyse d'erreurs

4.1 Descriptions des personnes

La figure 8a montre des exemples des descriptions générées par notre modèle de génération à partir des propriétés des caractères présentées dans la figure 8b. La première phrase est censée exprimer le nom entier, l'âge et le pays d'une personne. On peut voir que ceci a bien marché chez la troisième personne. Par contre, chez les autres les mots clés ont été utilisés pour former des phrases grammaticalement correctes mais souvent l'algorithme déclare l'âge de la personne comme sa taille. De plus, il écrit par exemple que « Zhang Lu » était le nom d'une entreprise ou que les Saint-Kitts-et-Nevis étaient une sorte de groupe au lieu d'un pays. D'ailleurs, il arrive que quelques mots clés sont malformés par le modèle comme par exemple le pays « Lesotho » qui est devenu « Leotho ». Ce sont alors surtout des genres de noms ou des pays que le modèle n'a pas vus pendant son entraînement où il fait des erreurs.

Dans cet exemple là, ça marche bien pour les catégories livre et sport (lignes 3 et 4 de chaque paragraphe) mais plutôt moyen pour les autres. Par contre on ne peut pas dire ça en général, ça dépend toujours des propriétés avec lesquelles on génère les phrases, ça veut dire si le modèle a déjà vu un mot comme ça dans une phrase pendant l'entraînement. Pour conclure on peut dire que les phrases générées sont toutes grammaticalement correctes et pour la majorité on peut comprendre ce que l'IA voulait dire même si les phrases n'expriment pas toujours exactement ce qu'on voulait décrire.

Samuel Hood, who is from the country of Leotho, has a height of 70.
 Samuel is a member of the crew of the meal, the name of the employee of the meal.
 Harenberg Verlag is the favourite book of Samuel.
 The favourite sport of Samuel is Frisian handball.
 The favourite music for Samuel is indietronica.
 The favourite book of Ramuel is The Monster.
 The favourite football team of Samuel is Pilolo.
 Cethyl is a favourite food of Samuel.

Zhang Lu is a 40 year old company from the Republic of the Congo.
 Zhang's occupation is a sacred Prostitution.
 Det sovande folket is a favourite book of Zhang.
 Wu Dao is the favourite sport of Zhang.
 Nakasi is the favourite of Zhang.
 Czlowiek z M-3 is a favourite of Zhang.
 The favourite football team of Zhang is the Midway.
 The frog cake is a favourite of Zhang.

The 29-year-old Brian Nielsen is from the country of Malawi.
 Brian is a course of course.
 The book Systema Mycologicum is a favourite book of Brian.
 The favourite sport of Brian is water skiing.
 African American is the favourite of a Brian.
 The favourite of a Brian is the point of No Return.
 The favourite of a Brian is a tag.
 The rolex dish of a Brian is also a favourite food.

Eric Newman, who is a member of the Saint Kitts and Nevis, has a height of 55.
 Eric's occupation is a toll collector.
 The favourite book of Eric is Lady Jane Grey.
 Ericans' favourite sport is football codes.
 mento is a favourite of Erican.
 The favourite of Eric is Farewell to Enrico Berlinguer.
 Eric's favourite games are the Humankind.
 keskul is a favourite of Erican.

name: Samuel Hood
 age: 70
 country: Lesotho
 job: employee of the Diet
 book: Harenberg Verlag
 sport: Frisian handball
 music: indietronica
 film: The Monster
 game: Pilolo
 food: Cethyl

name: Zhang Lu
 age: 40
 country: Republic of the Congo
 job: sacred prostitution
 book: Det sovande folket
 sport: Wu Dao
 music: Nakasi
 film: Czlowiek z M-3
 game: Midway
 food: frog cake

name: Brian Nielsen
 age: 29
 country: Malawi
 job: course grader
 book: Systema Mycologicum
 sport: water skiing
 music: African-American music
 film: Point of No Return
 game: tag
 food: rolex

name: Eric Newman
 age: 55
 country: Saint Kitts and Nevis
 job: toll collector
 book: Lady Jane Grey
 sport: football codes
 music: mento
 film: Farewell to Enrico Berlinguer
 game: Humankind
 food: keskul

(a) Quatre exemples pour des descriptions générées.

(b) Les propriétés personnelles utilisées pour la génération des description de la figure 8a

FIGURE 8 – Exemples pour la génération des description à partir des propriétés personnelles

4.2 Chat entre les IAs

La figure 9 montre un chat entres des IAs sans participation d'un être humain. Comme ici les phrases possibles sont pré-codées il n'est pas un miracle qu'ils sont correctes et font du sens. La compréhension des questions avec *gensim* marche très bien aussi, surtout parce que les sujets possibles se trouvent souvent directement dans les questions. En revanche la question « What is your preferred dish ? » ne contient pas directement le mot « food ». Mais comme « dish » et le mot le plus proche à « food » dans la question les autres caractères comprennent quand même. Le seul problème qu'on a dans cette partie c'est qu'on ne peut pas distinguer entre les sports qu'on fait (p.e. le ski nautique) et les sports qu'on joue (p.e. le foot). C'est pourquoi la phrase « I like playing water skiing » est générée.

Henk: Good day, my name is Henk Visser, I'm 74 years old and I live in Cyprus.
 Frederick: Hello, I am Frederick II, I'm 60 years old and I'm from Trinidad and Tobago.
 John: Good day, my name is John Harrison, I'm 63 years old and I'm from Laos.
 Richard: Good day, my name is Richard III, I am 71 years old and I come from Syria.
 John: I work as a aviation pioneer. What do you work as?
 Richard: I'm electrician.
 Frederick: My job is Nōgaku artist.
 Henk: I am Museum Technician.
 Richard: I like to eat Sambal pepaya. What do you like to eat most?
 Frederick: I like to eat Vigna radiata.
 John: I like to eat Haşhaşlı çörek.
 Henk: I like to eat Frutas de Aragón.
 John: I really like the book Al-Amali. What do you like to read?
 Richard: I really like the book Lepsius-Projekt.
 Henk: I find Vrikshasastramu is a pretty good book.
 Frederick: I am currently reading Garshaspname and i really like it.
 Frederick: I like watching rugby. What is your preferred sport?
 John: I'm a fan of memory sport.
 Richard: I am doing Nouka Baich.
 Henk: I'm a fan of boule lyonnaise.
 John: I like playing Russian Railroads. Are there games that you like to play?
 Henk: I really enjoy playing Fascination.
 Richard: I like Renegades.
 Frederick: I like Catch Phrase.

FIGURE 9 – Exemple d'un chat entre des IAs

4.3 Chat interactif

La figure 10 représente un exemple d'un déroulement du chat interactif. Là on peut voir que les IAs comprennent bien les sujets des phrases pour y répondre. Ils changent de sujet et créent un chat assez varié pour être intéressant et on apprend des choses inconnues quand on pose des questions spécifiques.

Please tell us your name: Stefan

Pyotr: Good day, my name is Pyotr Shuvalov, I am 28 years old and I live in Lesotho.

Carl: Good day, my name is Carl Andersen, I'm 29 years old and I live in Seychelles.

John: Hi, I'm John Hall, I'm 56 years old and I live in Bolivia.

Billy: Good day, my name is Billy the Kid, I'm 68 years old and I'm from Yemen.

Billy: How old are you?

Stefan: I am 21 years old.

Pyotr: From which country are you?

Stefan: Germany

Carl: The Sisterhood of the Traveling Pants was the film that I liked most. Which film did you like most?

Stefan: I liked the films of Harry Potter.

John: I really enjoyed watching the film Map of the Human Heart.

Pyotr: I really liked the film Furtivos.

Billy: My favourite film is Summer Snow.

Stefan: What is your favourite sport?

Billy: I like playing pallina.

John: I'm a Rossall hockey fan.

Pyotr: I am doing capoeira.

Carl: I often do AcroCap.

Stefan: I like skiing.

Pyotr: I like hands up. What is your favourite music?

Stefan: I often listen to rap.

Billy: My favourite music genre is melodic death metal.

John: My favourite music genre is minimal house.

Carl: I like Swing revival.

Stefan: What is melodic death metal?

Billy: The music genre melodic death metal is a subgenre of death metal.

Stefan: I like to play computer games.

John: I like Outpost.

Pyotr: I really enjoy playing Notakto.

Carl: I like Rack-0.

Billy: I like to play Army of Zero.

Stefan: When I was young I liked to read Die drei Fragezeichen.

John: I find Quai d'Orsay is a pretty good book.

Billy: I enjoyed reading the book Birds of South Asia. The Ripley Guide.

Carl: I find Mus'haf is a pretty good book.

Pyotr: I like the book Kleines Destillierbuch a lot.

Stefan: What are you doing for a living?

Pyotr: I work as a directing.

Billy: I work as a Master chef.

John: I'm electrician.

Carl: I work as a Fliegel-Adjutant.

Stefan: I want to become a software engineer.

Pyotr: I like to eat Gaplek. What is your preferred dish?

Stefan: I love crêpes.

John: My favourite dish is salted duck egg.

Carl: I like to eat milk pilaf.

Billy: I love stuffed intestines.

Stefan: bye

FIGURE 10 – Exemple pour un chat interactif

La figure 11 montre les information que l'IA pouvait extraire des messages écrites par l'utilisateur. Le problème ici c'est que les mots clés sont difficile à trouver puisque le mot le plus proche n'est pas forcément le mot intéressant. Un autre problème est bien sûr que dès qu'il s'agit d'une composition de mots il ne peut que récupérer un seul mot.

```
name: Stefan
age: 21
country: Germany
book: read
sport: skiing
music: software
film: Potter
game: I
```

FIGURE 11 – Les propriétés extraites des messages de l'utilisateur

Dans la figure 12 on peut voir la description générée avec les données de la figure 11. Comme il y a aussi des mauvaises données les phrases générées font pas beaucoup du sens non plus. Par contre pour le sport cela a bien fonctionné quand même.

```
Stefan, who was born in the country of Germany, has 21 years of age.
The book "Stefan" is a favourite book of Stefan.
The favourite sport of Stefan is Ski.
Stefan's favourite music is the computer.
Potter is the favourite for Stefan.
The favourite games of Stefan are I.
```

FIGURE 12 – La description de l'utilisateur avec les données présentées dans la figure 11

Néanmoins, il n'est pas possible de commencer une vraie discussion sur un sujet concret puisque les bots sont limités à « j'aime <sujet> » et la définition du sujet. Donc on n'a pas une longue durée de vie sur un sujet.

4.4 Idées non compris dans le projet final

On avait essayé d'avoir plus de profondeur dans les discussions sur une propriété d'un bot ou de l'utilisateur. Pour cela on a envoyé des requêtes pour récupérer la description d'un objet par exemple la date de création et le réalisateur d'un film. On a aussi tenté d'y arriver par *Word2Vec* en travaillant avec des additions et soustractions des mots vectorisés. Mais on a eu des résultats beaucoup trop médiocre, voir incompréhensible c'est pourquoi on ne le retrouve pas dans le chat interactif.

5 Conclusion et améliorations possibles

Donc on est capable de créer des bots semblants dans ses descriptions à un humain avec ses activités, préférences etc. Puis de comprendre le sujet d'une phrase et donc répondre en fonction avec les caractéristiques des bots. On peut aussi donner la définition des caractéristiques définis à travers *Wikidata*. Récupérer les mots clés et ainsi définir les caractéristiques de la personne écrivant et générer une description avec.

Pour y arriver on a utilisé *Wikidata*, *Word2Vec* et le modèle T5 pour la génération. Le projet est assez complet par rapport au temps et à ce qu'on a vu en cours.

Tout cela marche assez bien, bien qu'ils existent beaucoup d'améliorations globales pour affiner cette méthode. Voici des pistes plus concrètes d'amélioration qui pourraient rendre ce chat plus intéressant :

- Améliorer le modèle T5 en l'entraînant avec une base de donnée plus dédiée à cette tâche.
- Comprendre l'ironie, les mots clés de l'émotion, la négation pour savoir ce qu'il pense réellement du sujet et donc ne pas partir sur l'idée, ce qu'il écrit il aime bien.
- De la même manière savoir si une phrase est personnelle ou adressée à une autre personne.
- Avoir plus de profondeur dans la discussion.
- Créer des caractéristiques moins aléatoires donc plus réalistes, par exemple : Un italien aime de la nourriture italienne ou européenne et non mondiale. Pour cela on peut faire des requêtes beaucoup plus avancées qui cherchent que la nourriture relié à un pays.
- Faire évoluer les personnes : Pouvant changer de caractéristiques et l'écrire. S'approprier une caractéristique d'une autre personne.

Références

- [1] "Google's trained word2vec model in python." <https://mccormickml.com/2016/04/12/googles-pretrained-word2vec-model-in-python/>. Accédé : 27/12/2021.
- [2] "Webnlg challenge 2020." https://gitlab.com/shimorina/webnlg-dataset/-/tree/master/release_v3.0/en/train. Accédé : 26/12/2021.
- [3] "Fine tuning t5 transformer for data-to-text." <https://colab.research.google.com/drive/1TaPqHQ2CRHINFCRSWqSmdfF0U0F49yuj?usp=sharing>. Accédé : 26/12/2021.
- [4] "Ai text generation for seo : a practical example." <https://wordlift.io/blog/en/ai-text-generation-for-seo/>. Accédé : 27/12/2021.