DEDALE, Une solution facilitant l'accès aux aides médico-sociales par l'expression des besoins du patient et ses proches

Noe Redouin, Kevin Dethoor, Vincent Zalc, Theophile Cocquerez, Catherine Mallevaes, Jacques Vairon, Dan Istrate

DEDALE, Une solution facilitant l'accès aux aides médico-sociales par l'expression des besoins du patient et de ses proches

Noe Redouin[1], Kevin Dethoor[1], Vincent Zalc[1], Théophile Cocquerez[1], Catherine Mallevaes[2], Jacques Vairon[2] et Dan Istrate[1]

[1]Université de technologie de Compiègne, CNRS, Biomécanique et Bioingénierie, Centre de Recherche Royallieu – CS 60319 - 60203 Compiègne Cedex, France

[2]Association ARAMISE, 23 Rue des Vignerons, 45380 La Chapelle Saint Mesmin

kevin.dethoor@etu.utc.fr

Résumé - Les personnes souffrant de maladies rares présentent des handicaps de différents natures et nécessitent des aides médico-sociales. Trouver la bonne information en fonction des besoins de chacun est un vrai DEDALE. Nous proposons un chatbot permettant de répondre aux besoins des utilisateurs d'une manière simple en donnant les informations sur les institutions à contacter et sur les aides existantes. Les résultats d'identification de la question de l'utilisateur en utilisant des Transformers ainsi que la fusion de deux systèmes sont présenté dans cet article.

Mots-clés : chatbot, aides medico-sociales, maladies rares, IA, transformers, fusion de décision

I. INTRODUCTION

Les maladies rares conduisent à des situations de handicap complexes. L'accès à des aides adaptées aux besoins de chacun est un enjeu clé pour la construction d'un parcours de vie acceptable. En pratique toutefois, les patients et leurs familles sont rapidement confrontés à un « dédale » d'informations pour la plupart génériques.

Avec environ 8000 maladies rares, c'est près de 3 millions de personnes qui sont concernées sur le territoire français. Pour leur venir en aide, il existe de nombreux types d'aides médico-sociales mais leurs champs d'application, les conditions d'obtention et les démarches sont différents d'une aide à l'autre.

Pour les personnes atteintes de maladies rares, la recherche des possibilités d'aide peut devenir difficile, comme s'ils étaient perdus dans le labyrinthe que forme la foule d'informations disponibles (https://www.orpha.net/). D'autant que, pour atteindre ces informations, il faut généralement une idée déjà relativement précise des aides que l'on souhaite demander.

DEDALE propose un chatbot conversationnel inclusif, visant à orienter facilement chaque utilisateur touché par un ou plusieurs handicaps vers les aides médico-sociales correspondant à ses besoins spécifiques.

II. SOLUTIONS CHATBOT EXISTANTES

Un chatbot, aussi appelé assistant virtuel, est essentiellement un agent logiciel qui permet de simuler les conversations humaines par le biais du texte ou de la voix. Les utilisateurs s'adressent au chatbot dans leur propre langage ("en langage naturel") via l'interface de communication du chatbot (telle que Facebook Messenger, Twitter, site web) ou via une application personnelle.

Dans le cas idéal, le chatbot reconnaît le sens de nos phrases et nous fournit des réponses informatives, garde le contexte de la conversation et nous ne pouvons pas le distinguer d'un humain.

A. Types de chatbots

En pratique, il existe deux principales catégories de chatbots : les chabots simples ou scriptés et les chabots complexes ou apprenants.

Les premiers ne possèdent pas d'intelligence artificielle, ils suivent des schémas de dialogue prédéfinis aussi appelés "arbres de décision". Lorsqu'il parle à l'utilisateur, le robot donne des réponses à partir de ces scénarios préétablis. L'avantage est que ses phrases sont plus proches du langage naturel et qu'il y a peu d'erreurs grammaticales ; l'inconvénient est qu'il ne pourra pas répondre aux sujets non prédéfinis.

Les seconds, dotés d'une intelligence artificielle, sont capables d'effectuer des opérations contextuelles. L'intelligence artificielle leur permet de comprendre et d'interpréter le langage naturel et de personnaliser leur réponse à l'utilisateur. Le chatbot apprenant utilise l'apprentissage automatique pour se souvenir des conversations. Par conséquent, le chatbot apprendra de son expérience continuellement et améliorera sa réponse. Plus le robot interagit, plus il peut répondre à des demandes complexes et diverses.

B. *Architecture d'un chatbot*

L'architecture d'un chabot se compose de deux parties :

1. L'interface utilisateur, qui gère l'interaction avec l'utilisateur.
2. Le moteur conversationnel, qui est le cœur du chatbot. Il analyse la phrase de l'utilisateur soit en la comparant aux phrases de la base de données soit avec une analyse d'intention.

C. *Choix d'une solution*

Il existe plusieurs outils pour implémenter un chatbot, chacun avec ses caractéristiques :

- Botnation AI : basé sur une analyse des intentions, bien adapté au français, interface graphique pour web, fonctionnalités avancées. La version utilisable est payante.
- Botsify : avec apprentissage automatique donc adaptable au contexte. Version payante.
- Chatfuel : analyse des phrases, interface intuitive. Version à mettre en place payante.
- Team Brain : basé sur une base de données personalisable. Version payante.
- ChatterBot : basé sur apprentissage de phrase et calcul de similitude, bibliothèque open source en Python.
- Mobile Monkey : analyse des intentions. Version payante.
- Motion.ai : analyse des intentions, interface intuitive. Version payante.
- Surveybot : pour Facebook et seulement pour des sondages.
- Crisp : éditeur du chatbot, des API. Version plus d'un utilisateur payant.
- Wit.ai : chatbot de Facebook.

Les contraintes de notre application sont : possibilité d'adaptation à un domaine particulier (celui des aides médico-sociales), interface d'administration simple, version gratuite car l'association ne peut pas payer un abonnement.
Tenant compte de tout cela, notre choix s'est porté vers Chatterbot.

III. SOLUTION PROPOSÉE

Comme précédemment indiqué, nous avons basé notre solution sur la bibliothèque Chatterbot qui est conçue en Python et disponible en sources ouvertes. A cela, nous avons rajouté Django pour l'interface utilisateur et SQlite pour la base de données (voir figure 1).

La première version de DEDALE a été basée sur plusieurs corpus de questions-réponses spécifiques à des thèmes : adaptation au logement, adaptation du véhicule, aide financière, aide humaine, aide ménagère, aide technique.

Figure 1. Schéma de l'application DEDALE

Ces corpus ont été rédigés par l'association ARAMISE avec notre aide.

Une fois la question de l'utilisateur reçue via l'interface du chatbot, elle est pré-traitée par l'outil Spacy, dans des objets appelés « tokens » correspondant à chaque mot. Ainsi nous avons comme information l'entité nommée, les dépendances, la fonction grammaticale et la racine du mot. Ensuite la distance de Levenshtein entre la requête de l'utilisateur et les questions de la base de données est calculée (voir figure 2).

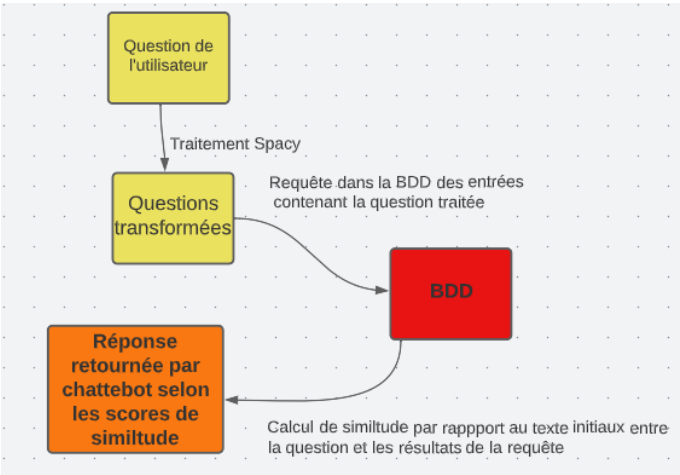

Figure 2: Schéma de traitement d'une question

La distance de Levenshtein calcule la similarité syntaxique (similarité basée sur la ressemblance des chaînes de caractères et non sur le sens des mots) entre deux phrases ou textes. Elle

est égale au nombre minimal de caractères qu'il faut supprimer, insérer ou remplacer pour passer d'une chaîne à l'autre [1].

Dans cette version, l'interaction avec l'utilisateur a lieu seulement au niveau d'une question et il n'y a pas de mémoire du contexte. Pour améliorer l'aspect conversationnel, nous avons introduit après chaque question, une réponse avec plusieurs propositions (sous forme de boutons) pour continuer le dialogue. Cela a nécessité la modification des corpus. Un exemple d'échange est présenté dans la figure 3.

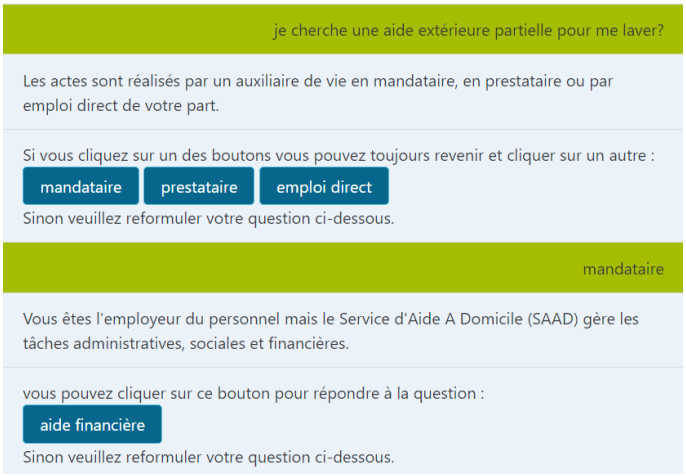

Figure 3: Exemple de conversation DEDALE

IV. UTILISATION DES TRANSFORMERS

Les Transformers représentent les outils de traitement naturel du langage les plus complets et efficaces aujourd'hui. Ils peuvent être appliqués dans un grand nombre de tâches telles que : la traduction ou l'élaboration de résumés automatiques, la classification, la génération de texte (ex : chatGPT, Open-AI). Ils sont aussi capables de réaliser beaucoup de tâches dans le domaine de la reconnaissance audio-visuelle. Le principe d'utilisation des Transformers est de partir sur de grands modèles pré-entraînés proposés en ligne puis de faire du "fine-tuning", c'est-à-dire de les affiner pour une tâche spécifique.

Le fonctionnement des Transformers est basé sur des réseaux de neurones récurrents avec notamment des réseaux à Long-Short-Term-Memory [Vaswani, 2017].

Initialement, les avancées ont été majoritairement réalisées en anglais, par Google et Open AI. Google AI a prouvé sa force avec son modèle BERT en 2018, modèle ayant été entraîné sur la totalité des pages Wikipédia en anglais. Il a fallu attendre 2019 pour que les équipes de Facebook AI Research associées aux chercheurs de l'INRIA rendent public CamemBERT, un modèle de type BERT, pré-entraîné sur des corpus en français. Le modèle CamemBERT est maintenant en open-source, hébergé sur le site internet de Hugging-face.

Cependant le modèle transformer de CamemBERT n'est pas directement construit pour évaluer la similarité des phrases, il existe pour cela une bibliothèque spéciale de Transformers appelée SBERT.

Les transformateurs de phrases (SBERT) constituent l'état de l'art actuel en matière d'intégration de phrases en langage naturel. Ils utilisent BERT et ses variantes comme modèle de base et sont pré-entraînés à l'aide d'un type d'apprentissage métrique appelé apprentissage contrastif.

A. Création d'un modèle Sentence-Transformer

Nous avons trouvé un seul modèle en français mis à disposition sur le site internet de Hugging-face : *french_semantic*, modèle qui répertorie exactement le même nombre de mots (un peu plus de 32000) que le modèle CamemBERT simple. Il faut savoir que la plupart des dictionnaires français répertorient le nombre de mots de vocabulaire à environ 60000, même si Le Grand Robert de la langue française compte autour de 100000 mots. On peut supposer que 32000 mots sont suffisants car on estime que le vocabulaire fondamental d'un adulte contient de 3000 à 5000 mots.

Nous avons créé notre propre modèle sur la base du modèle CamemBERT avec une adaptation à notre tâche, réalisée sur les corpus existants déjà dans DEDALE. Nous avons rajouté des acronymes spécifiques comme MDPH ou SSIAD.

B. Evaluation des différents systèmes proposés

Afin de choisir la meilleure méthode pour calculer la similarité entre la question de l'utilisateur et les questions des corpus, nous avons réalisé une évaluation sur un corpus de test.

Les méthodes que nous avons décidé de prendre en compte dans le test sont :

- La « Spacy similarity » directement proposée par Chatterbot.
- Le modèle *french_semantic* de sentence-transformer trouvé en open-source sur Hugging-face.
- Notre propre modèle de sentence-transformer appelé Dédale-scratch basé sur le modèle CamemBERT affiné.
- La distance de Levenshtein.

Pour éviter d'avoir un test biaisé, nous l'avons construit autour de nouvelles questions n'étant pas présentes dans la base de données mais dont la réponse attendue correspond à une réponse dans la base de données. Différentes métriques ont été utilisées pour évaluer la performance des modèles.

Pour cela, nous avons tiré 20 réponses au hasard dans la base de données et mis dans un fichier les questions liées à ces réponses dans la base de données. Par la suite, nous avons rempli le même fichier avec des questions menant aux mêmes réponses (environ 10 supplémentaires par réponse). Ces nouvelles questions devaient varier en terme de niveau de

langage, de fautes d'orthographes, de formulation… Nous avons finalement 174 questions pour le test.

C. *Les métriques utilisées pour le test*

La première métrique que nous avons décidé d'utiliser est d'ajouter 2 au score de la ligne si la première réponse est bonne puis d'ajouter 1 et 1 si les deuxième et troisième réponse sont bonnes (voir figure 4).

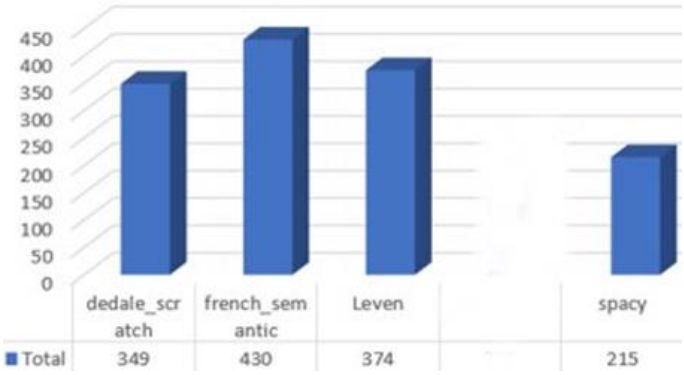

Figure 4: Scores des modèles

Le score maximum que les modèles pouvaient atteindre est 615. Le modèle ayant le score le plus élevé est *French_Semantic* (Transformers) avec 70 % de bonnes réponses suivi de la distance de Levenshtein avec 61 % puis de notre modèle Dédale scratch avec 58 %.

Le score original de CamemBERT sans apprentissage pour ce test est de 211, l'apprentissage fourni a donc augmenté ses performances de 64 %, lui permettant d'atteindre un score de 349.

La deuxième métrique que nous avons proposée est le score obtenu par chacun des modèles sur seulement la première réponse. Nous avons accordé +1 si la réponse à la question la plus similaire était bonne. Les scores obtenus sont dans la figure 5.

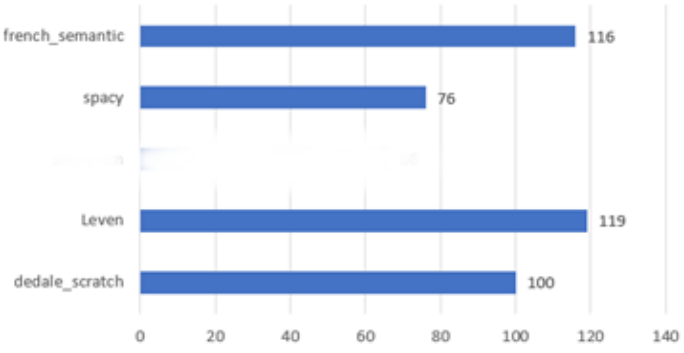

Figure 5: Scores des modèles, évaluation sur la première réponse

Le test ayant 174 questions, le score maximal pour cette métrique est de 174. Nous pouvons observer que la distance de Levenshtein avec un score de 119 (68 % de bonnes réponses) est la meilleure sur cette métrique, suivie de très près par le Transformer *french semantic* avec un score de 116 (67 % de bonnes réponses).

La troisième métrique que nous avons utilisée est le score maximum obtenu en regardant si les algorithmes de la distance de Levenshtein ou du Transformer *french_semantic* répondent bien à la question dans la première réponse qu'ils proposent (cela correspond à une fusion idéale entre les deux modèles). Si les deux ou seulement l'un d'entre eux a bon, on donne +1 au score. Le résultat obtenu fut alors de 146, soit 84 % de bonnes réponses.

Enfin la quatrième métrique que nous avons étudiée est le score maximum obtenu en regardant si l'algorithme de Levenshtein ou le Transformer *french_semantic* a bon à la première, deuxième ou troisième réponse. Le score obtenu est alors de 166 (95 % de bonnes réponses). Il y a donc seulement 8 questions pour lesquelles les deux modèles ont des résultats totalement faux sur leurs trois meilleurs choix de réponses.

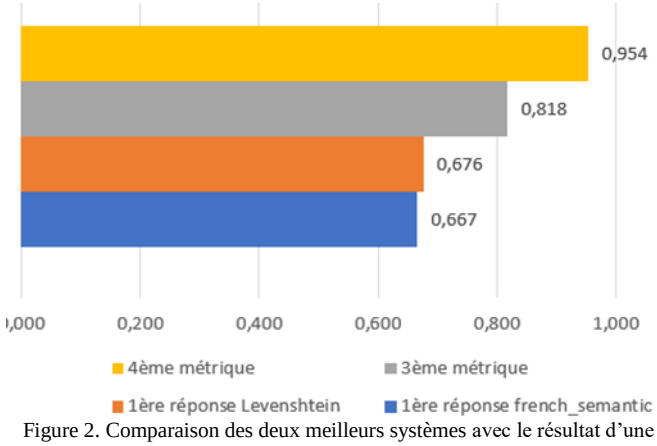

Figure 2. Comparaison des deux meilleurs systèmes avec le résultat d'une fusion idéale

La comparaison des algorithmes de Levenshtein et du Transformer *french_semantic* en fonction des métriques est présentée en figure 6.

V. FUSION ENTRE LES DEUX MEILLEURS SYSTEMES

Dans un premier temps, pour réaliser la fusion de décision, nous avons implémenté plusieurs algorithmes de calcul simples se basant sur la récurrence de proposition des réponses et leurs valeurs de similarités. Les algorithmes proposés sont les suivants :

- La pondération par multiplication : consiste à affecter un score à chaque réponse selon leurs classements (1ère place = 3, 2ème place = 2 et 3ème place = 1)

puis de multiplier ces scores selon le classement de la réponse par les deux comparateurs et enfin choisir la réponse ayant le score le plus élevé ;

- La pondération par addition : utilise la même méthodologie que la pondération par multiplication mais les scores sont additionnés dans cette fonction ;
- La standardisation : le choix de la réponse est effectué par rapport à la similarité la plus élevée après standardisation des données, suivant la formule suivante : $z = \frac{x - mean(x)}{stdev(x)}$ où z correspond aux nouvelles valeurs de similarités standardisées ;
- Les écarts entre les réponses : consiste à affecter à chaque réponse un score correspondant à son écart de similarité par rapport à la 3ème réponse proposée par son comparateur et puis à choisir la réponse avec le plus grand écart, correspondant donc à la réponse dont les comparateurs sont plus « sûrs » d'eux qu'à la normale.

A chacun de ces algorithmes s'ajoute également la priorisation du choix d'une réponse proposée plusieurs fois par un même comparateur, à condition qu'aucune autre réponse ne possède d'écart à la moyenne supérieur à l'écart-type puisque le comparateur sera plus sûr que cette dernière réponse, bien que proposée une seule fois, soit la bonne réponse.

L'algorithme simple le plus performant est l'algorithme de calcul d'écarts entre les réponses avec plus de 78 % de choix de bonnes réponses par rapport à 73 % de bonnes réponses si l'on ne choisit que la première réponse proposée par le comparateur basé sur le modèle *french_semantic*.

Dans un second temps, afin d'obtenir une performance plus proche des 95 % idéaux, nous avons mis en place un réseau de neurones artificiel, et plus précisément un réseau de neurones denses multicouches (DNN) avec rétropropagation et ajustements des poids par la méthode de descente des gradients. Le corpus de questions de test a été augmenté dpur passer de 174 à 321 questions. Pour le DNN, en partant des 321 questions, nous avons un total de 1090 réponses différentes subdivisées en 719 réponses utilisées pour l'apprentissage du modèle (2/3) et 371 réponses utilisées pour tester la performance du modèle (1/3).

Ces 13 paramètres suivants sont fournis en entrée du réseau de neurones :

- Sim_french_stand : score de similarité standardisé attribué à la réponse par le modèle *french_semantic* ;
- Sim_leven_stand : score de similarité standardisé attribué à la réponse par la distance de Levenshtein ;
- F1 : booléen égal à 1 si la réponse est proposée en première position par le modèle *french_semantic* ;
- F2 à L3 : même fonctionnement que F1 selon le classement et le modèle proposant la réponse ;
- Nb_proposition : nombre de fois où la réponse est proposée par les deux comparateurs ;
- Nb_french : nombre de fois où la réponse est proposée par le modèle *french_semantic* ;
- Nb_leven : nombre de fois où la réponse est proposée par le modèle de Levenshtein ;
- Ecart_french : écart mesuré entre la similarité de la question proposée par le modèle *french_semantic* et la 3ème réponse proposée par ce comparateur ;
- Ecart_leven : écart mesuré entre la similarité de la question proposée par le modèle de Levenshtein et la 3ème réponse proposée par ce comparateur.

La sortie du DNN est 0 si la réponse est considérée comme étant fausse et 1 si la réponse est considérée comme étant correcte. Les paramètres optimaux du DNN afin d'obtenir une performance maximale avec les données dont nous disposons sont les suivants : 10 couches cachées de 64 neurones chacune, un pas d'apprentissage de 0,001 et un nombre d'itérations d'actualisation du modèle fixé à 50000. Si l'on réduit ces paramètres, la performance du réseau de neurones diminue, et si on les augmente dans l'optique d'améliorer la performance du réseau, du surapprentissage est détecté grâce au jeu de données de test.

Finalement, la performance obtenue s'élève à 86 % pour le jeu de données de test par rapport à 94 %, la limite maximale d'une fusion idéale. Une synthèse des résultats est présenté dans la figure 7.

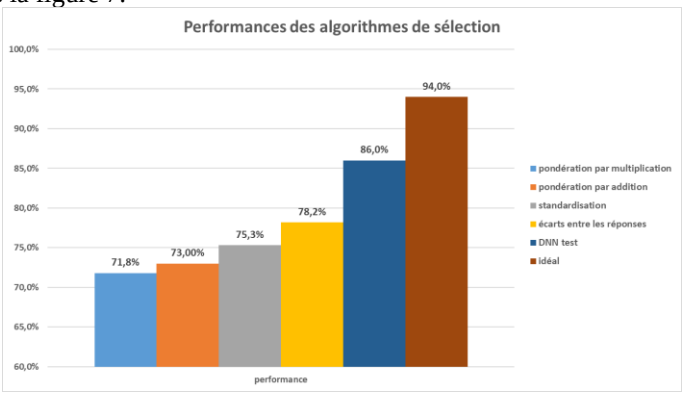

Figure 3. Synthèse des résultats des différents algorithmes de fusion

VI. CONCLUSIONS ET PERSPECTIVES

Cet article propose un chatbot permettant de répondre à la problématique de l'existence de plusieurs aides médico-sociales pour les personnes en situation de handicap et surtout avec une maladie rare. L'information concernant ces aides est présente et répartie à différents endroits et n'est pas présentée en rapport avec les besoins des personnes. Nous proposons un

chatbot permettant d'aider les personnes à identifier les aides adaptées à leurs besoins à travers un outil d'utilisation simple.

Plusieurs développements ont permis, en partant d'une solution simple open-source, de proposer un outil basé sur les technologies de l'état de l'art (les Transformers) mais adapté à la problématique.

Nous envisageons de continuer l'optimisation en travaillant plus spécialement sur l'aspect conversationnel par le rajout d'une mémoire de contexte.

REMERCIEMENTS

Nous remercions la Fondation des Maladies Rares ainsi que la filière BrainTeam qui ont financé plusieurs stages de fin d'étude et nous ont aidé dans la définition des réponses possibles. Nous remercions aussi le laboratoire BMBI CNRS UMR 7338 qui a financé le dernier stage de fin d'études.

REFERENCES

[1] Léonard Dumas Milne Edwards. Conception de formes de relecture dans les chaînes éditoriales numériques. Autre [cs.OH]. Université de Technologie de Compiègne, 2016. Français. ⟨NNT : 2016COMP2254⟩. ⟨tel-01562039⟩

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, Attention Is All You Need, https://doi.org/10.48550/arXiv.1706.03762.

[3] Lewis Tunstall, Leandro von Werra, Thomas Wolf – Natural Language Processing with Transformers Building Language Applications with Hugging Face, May 2022, O'Reilly Media, Inc., ISBN: 9781098136796

[4] Delip Rao, Brian McMahan - Natural Language Processing with PyTorch_ Build Intelligent Language Applications Using Deep Learning (2019, O'Reilly Media). ISBN: 9781491978238

[5] B. K. Jha, C. M. V. Srinivas Akana and R. Anand, "Question Answering System with Indic multilingual-BERT," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2021, pp. 1631-1638, doi: 10.1109/ICCMC51019.2021.9418387.