Sorbonne Université
Master ANDROIDE

Évaluation de débats en ligne par comparaison d'arguments

UE de projet M1

Simon RIGOLLIER – Michelle SONG – Aurélien CHAMBOLLE-SOLAZ

2025

Table des matières

Chapitre 1

Introduction

À une époque où les interactions sociales reposent majoritairement sur des plateformes numériques, les sites Internet et applications de débats en ligne ont vu leur intérêt croître fortement pour occuper aujourd'hui un rôle crucial dans la diffusion d'idées. Ils permettent à leurs utilisateurs d'accéder de manière simple à un espace de libre d'expression démocratique afin d'échanger sur des enjeux sociaux, environnementaux ou économiques. Ces plateformes facilitent également la communication entre des participants potentiellement éloignés géographiquement, tout en assurant un certain niveau de qualité du débat. Contrairement aux réseaux sociaux plus classiques, où de tels échanges apparaissent rapidement chaotiques, la mise en place d'un cadre structuré permet d'aboutir à des discussions organisées avec un taux de participation tout aussi élevé. Il est donc clair que ces outils occupent une place majeure dans un immense espace public numérique. [1]

Dans le cadre de l'UE PROJET du master ANDROIDE, nous souhaitions contribuer au système de débats en ligne. Sous la direction de M. Nicolas MAUDET, nous avons été familiarisés avec la plateforme Kialo [2] afin de comprendre et d'évaluer les limites des procédés déjà mis en œuvre. Ainsi, nous avons pu élaborer un dispositif et déterminer différentes stratégies complexes permettant une approche différente de l'évaluation de débats en ligne. Notre application de comparaison s'appuie sur les méthodes de débat classiques pour proposer une analyse précise du comportement des utilisateurs lorsque présentés à différents arguments.

L'ensemble du code, développé en langage Python, est disponible sur GitHub : `https://github.com/RIGSimon/debats-en-ligne`

Chapitre 2

État de l'art

2.1 L'approche hiérarchique : structuration logique des arguments

Certaines plateformes de débat en ligne s'inspirent de la logique argumentative classique, en structurant les échanges sous forme hiérarchique. Cette organisation repose sur un argument central auquel sont rattachés des arguments de soutien ou de contestation, eux-mêmes susceptibles d'être développés ou réfutés à leur tour. Ce modèle vise à rendre les débats lisibles, en explicitant les relations logiques entre les idées.

Une des plateformes qui repose sur ce système est Kialo. Chaque débat commence par une question principale, à laquelle sont rattachés des arguments pour et/ou contre. Pour chaque argument, les utilisateurs peuvent voter s'ils y sont pour ou contre, puis ajouter un argument qui justifie leur choix dans ce cas. Cela forme une hiérarchie logique qui permet d'examiner les débats à différents niveaux de profondeur.

Les utilisateurs peuvent aussi évaluer l'impact des arguments (sur une échelle de 1 à 5), ce qui influe sur leur position dans l'arbre. Ce mécanisme aide à faire émerger les arguments perçus comme les plus solides, tout en facilitant l'identification des zones de désaccord ou de consensus. Cependant, ce système présente certaines limites. L'évaluation des arguments repose principalement sur des votes d'utilisateurs, censés refléter leur appréciation de la pertinence ou de l'impact d'un argument. Or, cette méthode peut prêter à confusion : il n'est pas toujours clair si un vote exprime une adhésion au contenu de l'argument, ou une reconnaissance de sa qualité argumentative.

2.2 L'approche pair-à-pair : évaluation préférentielle par comparaison

Une autre manière d'aborder les débats en ligne consiste à privilégier une approche basée sur la comparaison directe d'idées. Plutôt que de demander aux utilisateurs d'articuler ou d'évaluer des arguments dans une arborescence logique, certaines plateformes reposent sur des systèmes de *wikisurveys*, qui permettent aux participants de juger entre deux propositions à la fois.Cette méthode s'appuie sur un principe simple mais pertinent : au lieu d'attribuer une note à chaque proposition de manière isolée, les utilisateurs choisissent

entre deux options présentées au hasard, de manière répétée. Au fil des comparaisons, un classement global des idées émerge, basé sur les préférences agrégées de l'ensemble des participants. [3]

La plateforme AllOurIdeas utilise ce mécanisme. Les utilisateurs peuvent voter à l'infini entre des paires d'idées, ou bien proposer leurs propres suggestions. Ce système hybride entre sondage et participation collaborative favorise l'émergence d'idées nouvelles et représentatives, sans exiger une structuration préalable du débat.

L'approche comparative présente plusieurs avantages : elle est simple à utiliser, facilement déployable, et encourage une participation massive, même de la part de publics peu familiers avec les débats structurés. Toutefois, elle ne permet pas de visualiser les relations logiques entre les idées, ni de distinguer les arguments de soutien de ceux de réfutation. Elle se concentre sur la perception relative de préférence, ce qui en fait un outil robuste pour établir une hiérarchie d'opinions, mais moins adapté à l'analyse du raisonnement argumentatif.

2.3 Vers une approche combinée

Le projet que nous proposons cherche à concilier les deux perspectives vues précédemment :

— conserver la structure logique d'un débat comme dans Kialo : pour préserver la structure logique et hiérarchique des débats. Cette structuration rend explicites les liens de soutien ou d'opposition entre les idées, ce qui favorise une meilleure compréhension du raisonnement global.

— tout en utilisant la comparaison préférentielle par paires pour l'évaluation des arguments, à la manière d'AllOurIdeas : pour obtenir des jugements plus nuancés. Ce type de comparaison est plus intuitif et améliore la qualité des préférences exprimées.

En combinant ces deux dimensions, la structure logique et l'évaluation préférentielle, notre approche vise à surmonter les limites propres à chacun des systèmes pris isolément. Elle permet de naviguer dans l'arborescence d'un débat tout en collectant des évaluations plus fiables, mieux ancrées dans des comparaisons directes, et potentiellement moins influencées par d'autres facteurs qui ne sont pas pertinents pour leurs choix.

Cette méthode ouvre la voie à une analyse plus fine des débats, en identifiant la cohérence des préférences exprimées par les participants, la stabilité de leurs jugements, ou encore les tensions entre différentes branches de l'argumentation.

À notre connaissance, peu de travaux ou d'outils existants combinent à ce jour ces deux dimensions, ce qui constitue l'originalité de notre approche.

Chapitre 3

Contributions

3.1 Présentation générale

L'application a été développée en Python en utilisant la bibliothèque Tkinter pour construire l'interface graphique. Ce choix nous a permis de créer rapidement une interface simple et interactive, adaptée à la tâche de comparaison par paires.

Nous avons structuré l'application autour d'une boucle principale qui gère l'affichage, les événements utilisateurs (clics, saisies), et la sélection des prochaines paires via les fonctions associées aux différentes stratégies.

3.2 Collecte et stockage des données

3.2.1 Débats

Conformément à l'approche présentée dans l'état de l'art, afin de garantir une structuration logique entre les arguments, nous avons exporté au format JSON des débats existants depuis Kialo, puis les avons représentés sous la forme d'un graphe orienté. Cette représentation permet de préserver les relations hiérarchiques entre les arguments, qu'ils soutiennent ou contredisent l'argument principal.

Le graphe est construit à l'aide de la bibliothèque NetworkX, à partir d'un parseur qui lit et traite les fichiers JSON. Chaque nœud du graphe correspond à un argument, enrichi de métadonnées récupérées depuis le fichier source, à savoir :

- `author_identity_id` : l'ID de la personne qui a rédigé l'argument ;
- `created` : la date de création de l'argument (long int) ;
- `text` : le contenu de l'argument ;
- `last_modified` : la date de la dernière fois où l'argument a été modifié (long int) ;
- `votes` : une liste de taille 5, où chaque position correspond à la valeur de l'échelle (elle va donc de 1 à 5) et chaque valeur correspond au nombre de personnes qui ont voté pour cette échelle.

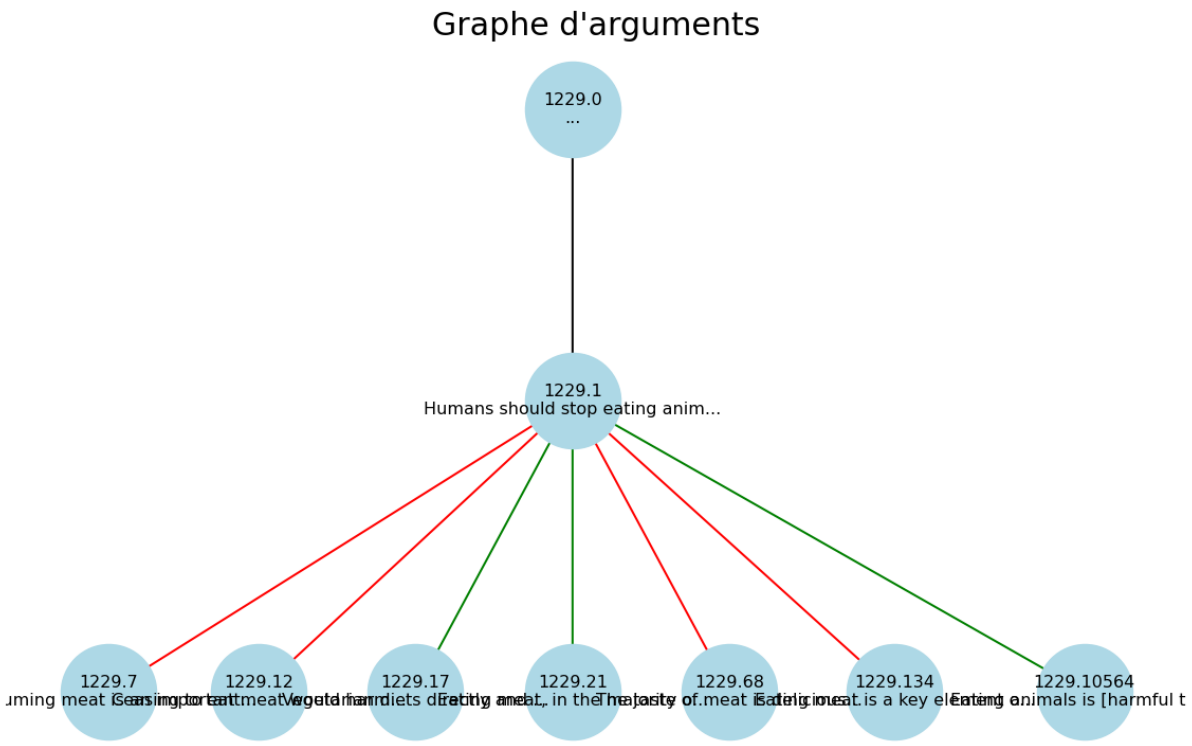

FIGURE 3.1 – Représentation du graphe d'un débat issu de Kialo

Les liens entre les arguments sont ajoutés en tant qu'arêtes, avec un attribut `relation` indiquant s'il s'agit d'un argument favorable (relation = 1, en vert) ou défavorable (relation = −1, en rouge).

Dans la figure 3.1 (ci-dessus), par exemple l'argument 7 : `"Consuming meat is an important part of many cultures and religions"` est un argument contre (relié en rouge à) l'argument 1 : `"Humans should stop eating animal meat"`.

On remarquera, par ailleurs, que la profondeur de l'argument dans le graphe donne un indice sur la spécificité de l'argument : plus la profondeur est élevée, plus l'argument est spécifique. Cette propriété va être étudiée et utilisée dans notre analyse des résultats.

3.2.2 Utilisateurs

Afin de permettre à de multiples utilisateurs de profiter de notre application, nous avons intégré à notre système une simple base de données au format JSON (fichier *user_ db.json*), mémorisant les noms d'utilisateurs et les mots de passe. Ces données sont stockées localement sur chaque machine et ne sont pas transmises à des partis tiers. Ainsi, les participants peuvent s'inscrire ou se connecter via l'interface de connexion avant d'initialiser une séquence d'arguments.
Le fichier est structurée sous forme de dictionnaire, où chaque nom d'utilisateur est associé à un mot de passe, par exemple :

```
{"misoop": "ms112233"}
```

Pour enrichir l'interaction, nous avons également ajouté la possibilité pour l'utilisateur de laisser un feedback après chaque comparaison. Cette fonctionnalité permet de collecter

des informations qualitatives, comme les raisons d'un choix ou des remarques sur la clarté des arguments, ce qui pourra être utile pour une analyse plus fine ou pour améliorer les futures itérations de l'interface.
Le format du fichier JSON associé (fichier *feedback_ db.json*) est la suivante : chaque clé représente le nom d'utilisateur, qui est associée à une liste de chaînes de caractères. Ces chaînes de caractères contiennent les identifiants des deux arguments et le contenu du retour que l'utilisateur a rédigé (format spécial). Voici un exemple :

```
{ "misoop": [ "1229.786, 1229.3315 : L'argument du haut est trop général alors que celui du bas est trop spécifique.", "1229.3832, 1229.731 : L'argument du haut est pertinente, mais des recherches récentes ont montrées que c'est faux." ] }
```

Pour pouvoir analyser avec plus de précisions, après chaque séquence d'arguments effectuée, on enregistre la coloration de l'utilisateur vis-à-vis du sujet principal (un score global) aux différentes profondeurs du graphe. Cela permet d'avoir une analyse globale sur la position de l'utilisateur sur chaque niveau de l'arbre (et donc, en fonction de la spécificité des arguments).
Le format du fichier JSON associé (fichier *user_ stats.json*) est le suivant :

- La clé de premier niveau (dans l'exemple, "random") indique la stratégie de sélection des paires utilisée par l'utilisateur (expliqué dans la suite) ;
- À l'intérieur, on trouve une seconde clé (dans l'exemple, "misoop") correspondant au nom d'utilisateur ayant réalisé les comparaisons avec cette stratégie ;
- Cette clé est associée à une liste de listes. Si l'utilisateur a effectué la séquence 3 fois en stratégie "random", alors il y aurait 3 sous-listes. Dans ces sous-listes, chaque élément est classé par ordre croissant de la profondeur de l'arbre, et correspond au score global associé à ce niveau de profondeur. Ce score est calculé de la façon suivante : pour chaque niveau, initialement ce score est nul. Ensuite, on ajoute 1 (si l'argument est pour) ou -1 (argument contre) au score associé à la profondeur de l'argument. Ainsi, si le score est strictement positif (à la profondeur associée), on pourrait conclure que l'utilisateur est globalement *pour* à cette profondeur.

Un exemple :

```
{ "random":  "misoop": [ [ 3.0, 2.0, 5.0, 2.0, 6.0, -1.0 ] ]  }
```

3.3 Comparaison par paires

Pour implémenter ce mécanisme, nous avons développé une interface utilisateur qui génère des paires d'arguments. L'utilisateur doit alors choisir l'argument qu'il juge le plus convaincant, ou, le cas échéant, cliquer sur le bouton "Je ne peux pas décider". Il a également la possibilité de rajouter un feedback sur la comparaison courante, qui sera ensuite stocké dans un fichier JSON.

La génération des paires d'arguments se fait différemment selon la stratégie choisie, chacune étant implémentée sous la forme d'une fonction distincte. En effet, en fonction de la stratégie choisie, l'analyse des résultats pourrait différer : notre but final étant de conclure si l'utilisateur est orienté *pour* ou *contre* la question principale, nous pourrons comparer les différentes stratégies à l'aide de vrais tests plus tard dans la section 3.4.

Par ailleurs, nous veillons à vérifier la cohérence des choix de l'utilisateur quelque soit la stratégie choisie, à l'aide du test de la transitivité :
Si l'utilisateur juge que :

$$\text{un argument } A \text{ plus convaincant à } B$$

et,

$$\text{un argument } B \text{ plus convaincant à } C$$

Alors, la transitivité permet de confirmer que A doit également être jugé plus convaincant que C.
Pour implémenter ce système, déterminer B tel que $A \succ B$ et $B \succ C$ n'est pas toujours garanti et dépend fortement de ce qu'on aura arbitrairement choisi pour A. Pour déterminer ces arguments, nous avons procédé de la façon suivante : Choisir N_1, N_2, N_3 et N_4 tels que,

$$N_1 \succ N_2$$

$$N_3 \succ N_4$$

Puis, en comparant les paires (N_1, N_3) ou (N_2, N_4), on peut obtenir un test de transitivité. Par exemple, si on choisit la paire (N_1, N_3), et que l'utilisateur juge que $N_1 \succ N_3$, alors on peut faire le test "$N_1 \succ N_3$?" ($A = N_1, B = N_3, C = N_4$). Et cela fonctionne pour n'importe quelle comparaison entre (N_1, N_3) et (N_2, N_4) au départ.

3.3.1 Stratégie aléatoire

La première stratégie que nous avons mise en place consiste à générer des paires d'arguments de manière entièrement aléatoire, indépendamment de leur position dans le graphe.

Cependant, cette stratégie ne garantit pas que les arguments comparés aient un lien sémantique ou logique fort. Ainsi, les résultats obtenus à partir de cette méthode peuvent être plus bruyants, et l'interprétation de l'orientation globale de l'utilisateur (pour ou contre la question principale) peut s'avérer moins précise que dans les stratégies plus structurées.

Les comparaisons ainsi obtenues sont néanmoins soumises au test de transitivité, ce qui permet de vérifier la cohérence interne des choix de l'utilisateur, même en l'absence de structure explicite.

3.3.2 Stratégie BFS

Cette stratégie repose sur un parcours en largeur du graphe des arguments. À partir de l'argument principal, les arguments sont explorés niveau par niveau, en respectant l'ordre hiérarchique du débat.

Contrairement à la stratégie aléatoire, cette méthode assure que les arguments comparés soient relativement proches dans la structure du débat, ce qui favorise des comparaisons plus pertinentes et informées. L'analyse des préférences de l'utilisateur gagne ainsi en cohérence, et permet une meilleure estimation de son orientation globale vis-à-vis de la question principale.

Cependant, cette stratégie reste sensible à la structure locale du graphe : certains arguments profonds dans l'arbre peuvent être négligés, ce qui peut biaiser l'évaluation de l'utilisateur en la focalisant sur une partie du débat.

3.3.3 Stratégie DFS

Cette stratégie repose sur un parcours en profondeur du graphe. A partir de l'argument principal, cette méthode explore une branche du graphe jusqu'à atteindre une feuille, avant de revenir en arrière pour explorer une autre branche.

Cette stratégie permet à l'utilisateur de se concentrer sur des chaînes argumentatives complètes, en suivant le fil d'un raisonnement jusqu'à son aboutissement. Cela peut aider à évaluer plus finement certaines lignes argumentatives, en mettant en évidence la logique interne d'un enchaînement d'arguments.

Cette méthode présente aussi des limites. Comme pour le parcours BFS, en suivant qu'une seule branche à la fois, elle peut négliger certaines zones du graphe. Cela peut biaiser l'analyse globale si l'analyse ne permet pas de couvrir équitablement l'ensemble des arguments favorables et défavorables qui ont été rédigés.

3.3.4 Stratégie "priority"

La stratégie dite "priority" exploite un parcours en largeur du graphe représentatif du débat, en ajoutant une contrainte sur les arguments proposés à l'utilisateur. Lors de la séquence, les comparaisons impliqueront toujours, lorsque cela est possible, une thèse ainsi qu'une antithèse de la question précédente dans le graphe représentatif.

Cette stratégie propose une approche similaire mais plus dichotomique de la stratégie BFS présentée plus tôt : en un nombre de comparaisons restreint, elle permet d'aborder un éventail statistiquement plus large de points de vue concernant l'argument principal.

Elle souffre néanmoins des mêmes contraintes que la stratégie de parcours en largeur, en n'exploitant pas entièrement les nœuds situés en profondeur du graphe. La stratégie "priority" peut également échouer dans son approche si le débat ne présente pas un nombre proche d'arguments jugés "pour" et "contre", imitant simplement la stratégie BFS.

3.3.5 Stratégie tournoi

La stratégie *tournoi* sélectionne les fils d'un nœud de manière à obtenir un nombre de comparaisons aussi proche que possible du nombre de questions choisi par l'utilisateur. Par exemple, si l'utilisateur sélectionne 10 questions, la fonction `tournament_order(graph, root, nb)` dans `graph.py` renvoie une structure de tournoi générant environ 10 comparaisons.

Nous avons également développé une fonction de simplification du graphe, qui permet d'ignorer certaines branches où un tournoi n'est pas pertinent — par exemple, lorsque tous les fils d'un nœud soutiennent unanimement (ou s'opposent unanimement à) l'argument du parent.

Le tournoi s'effectue en deux étapes intermédiaires suivies d'une finale : on commence par comparer entre eux tous les arguments « contre », afin de désigner le meilleur représentant

de cette position ; on fait de même pour les arguments « pour ». Ensuite, les deux gagnants s'affrontent dans un duel final, afin de déterminer si l'utilisateur est globalement plutôt favorable ou opposé à l'argument principal du nœud parent.

3.4 Analyse des résultats

Afin de fournir une analyse des choix effectués par les utilisateurs, nous avons implémenté un score associé à chaque argument de la séquence. Ce score est calculé par la formule suivante :

$$score(arg) = \frac{relation(arg)}{profondeur(arg)}$$

où *relation(x)* désigne la coloration de l'argument x vis-à-vis du sujet principal du débat (pour ou contre et symbolisé par une fonction à valeurs binaires dans 1, -1), et *profondeur(x)* désigne sa position relative dans l'arbre caractéristique du débat. Le score global de l'utilisateur correspond à la somme pondérée des scores des arguments qu'il aura sélectionnés au cours de la séquence :

$$score(user) = \sum score(arg)$$

Ainsi, un argument répondant directement à la thèse initiale possède un impact plus important qu'une réponse à un sous-débat construit au sein de la discussion générale [4].

Pour une meilleure visualisation, l'interface d'analyse propose un diagramme en camembert (Fig. B.4) de la répartition des choix d'arguments. De plus, l'utilisateur sera présenté avec un taux de cohérence, évalué en fonction de son respect par rapport à la règle de transitivité présentée en 3.3.

Dans le cas où l'utilisateur sélectionne la stratégie tournoi, l'analyse des résultats est différente : elle retourne l'argument gagnant du tournoi et conclut si l'utilisateur est plutôt en faveur ou contre l'argument père.

Chapitre 4

Conclusion

Notre outil propose donc une approche combinatoire des méthodes de structuration et d'évaluation des arguments d'un débat, et permet une analyse plus élaborée des préférences des participants par la différenciation de stratégies d'élicitation. En récupérant un débat au format correspondant, il est possible d'utiliser ce programme sur une grande variété de sujet, multipliant les domaines d'application. Les algorithmes développés s'appuient sur les travaux antérieurs de parcours de graphe tout en apportant certaines innovations afin d'élargir les possibilités de stratégies de séquence. Néanmoins, bien que robuste, notre code paraît technique et peu présenter un obstacle à la compréhension de ses mécanismes internes. De ce fait, l'amélioration ou l'ajout de nouvelles fonctionnalités par un parti-tiers peut s'avérer complexe.

Il nous sera possible ultérieurement d'apporter une refonte visuelle de l'interface afin de la rendre plus agréable, et de corriger des troubles mineurs spécifiques à des systèmes d'exploitations : par exemple, certaines versions de macOS présentent des problèmes d'affichage du texte (pour le moment, l'affichage en couleur ne fonctionne pas, donc nous affichons les informations sous forme de texte). En cas de découverte de nouvelles stratégies de séquençage présentant un intérêt quant à l'étude du comportement des utilisateurs, nous envisageons d'implémenter ces dernières dans notre code pour renforcer notre capacité d'analyse.

Bibliographie

[1] Francesca Cerutti, Madalina Cramer, Marc Guillaume, Eyke Hadoux, Anthony Hunter et Sylwia Polberg. « Empirical Cognitive Studies About Formal Argumentation ». In : *Handbook of Formal Argumentation*. 2021. Chap. 14 (page 1).

[2] A. P. Young, S. Joglekar, G. Boschi et N. Sastry. « Ranking comment sorting policies in online debates ». In : *Argument & Computation* 12 (2021), p. 265-285 (page 1).

[3] Matthew J. Salganik et Karen E.C. Levy. « Wiki surveys : Open and quantifiable social data collection ». In : *PLoS ONE* 10.5 (2015), e0123483 (page 3).

[4] Jordan Thieyre, Aurélie Beynier, Nicolas Maudet et Srdjan Vesic. « Reassessing the Impact of Reading Behaviour in Online Debates Under the Lens of Gradual Semantics ». In : *Proceedings of the Fifth International Workshop on Systems and Algorithms for Formal Argumentation (SAFA-24)*. 2024 (page 9).

Annexe A

Cahier des charges

Contexte

Les débats en ligne sont souvent influencés par la manière dont les arguments sont formulés, présentés et évalués. Sur des plateformes comme Kialo, les utilisateurs votent pour ou contre des arguments, mais ces votes ne permettent pas toujours de clarifier l'intention exacte des participants. Il est souvent difficile de savoir si un vote exprime un soutien à l'argument ou s'il reflète simplement une évaluation de sa pertinence.

L'idée est donc de développer une méthode d'analyse plus précise en remplaçant le vote traditionnel par une approche de comparaison directe entre arguments.

Objectifs

Le principal objectif du projet est de créer une application permettant de charger des débats sous forme de graphes d'arguments et de faire des comparaisons pair-à-pair entre ces arguments.

L'application proposera deux arguments à comparer et permettra aux utilisateurs de choisir celui qu'ils jugent le plus convaincant. Ce processus sera répété pour affiner les résultats.

Elle devra aussi enregistrer les réponses des utilisateurs et permettre une analyse des résultats afin de tirer des conclusions sur la manière dont les différents arguments soutiennent ou attaquent la question principale du débat.

Fonctionnalités

L'interface du prototype sera organisée en trois principaux modules :

Chargement d'un débat et lancement de la séquence de questions

Le système devra être capable d'importer un débat au format JSON, préalablement structuré et extrait de Kialo. Cette base de données servira de support pour toutes les inter-

actions et analyses menées par l'application.

Les métadonnées associées aux arguments pourront également être prises en compte pour enrichir les comparaisons.

Une fois le débat chargé, l'utilisateur pourra initier une séquence de questions permettant d'évaluer les arguments de manière comparative. Plusieurs stratégies seront mises en place pour poser ces questions :

— Présenter deux arguments en faveur d'une même position et demander lequel est le plus convaincant ;
— Comparer deux arguments opposés (un pour et un contre) afin de mieux cerner l'impact des points de vue divergents.

Chaque utilisateur devra répondre à plusieurs comparaisons successives, permettant ainsi d'établir une hiérarchie des arguments en fonction de leur pouvoir de persuasion.

Il sera également envisageable d'ajouter un module de retour utilisateur pour recueillir leur feedback sur un argument, ainsi qu'un module permettant de remonter dans l'arborescence des arguments s'ils souhaitent connaître le contexte de l'argument courant.

Enregistrement et gestion des votes

Les utilisateurs devront se connecter via un identifiant afin de garantir la traçabilité des réponses. Chaque vote sera enregistré et associé à un utilisateur pour permettre une analyse approfondie des préférences et tendances argumentatives. Le système devra permettre l'accumulation des votes et leur récupération pour une exploitation ultérieure.

Pour favoriser l'engagement, une représentation visuelle des tendances argumentatives pourra être intégrée, permettant aux participants de situer leurs choix par rapport à l'ensemble des réponses collectées.

Analyse des résultats

Une fois les votes collectés, un module d'analyse proposera plusieurs axes d'interprétation :

— Étudier la répartition des préférences et mettre en évidence les arguments les plus convaincants,
— Évaluer dans quelle mesure la question principale du débat est soutenue ou contestée.

On pourra explorer différentes méthodologies pour quantifier l'impact des comparaisons par paires par rapport aux méthodes traditionnelles de pondération des arguments.

Annexe B

Manuel utilisateur

Une fois l'archive récupérée via le dépôt distant GitHub, il vous suffira de compiler et d'exécuter le fichier Python *app.py*.

Vous aurez alors accès à l'interface d'inscription et de connexion. Lors de la première utilisation, il faudra s'inscrire dans la base de données avant de pouvoir se connecter les fois suivantes.

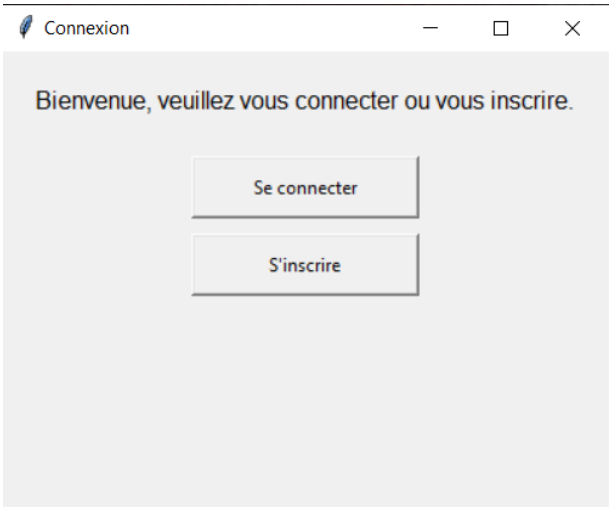

FIGURE B.1 – Aperçu de l'interface de connexion/inscription

L'interface de chargement de débats apparaîtra à la suite de votre identification : vous pourrez sélectionner les fichiers JSON fournis avec l'archive, ou choisir un débat au format correspondant. Les paramètres de la séquence pourront être définis dans l'interface suivante, au travers des menus déroulants : le nombre de questions et la stratégie de comparaison, avant de lancer le débat.

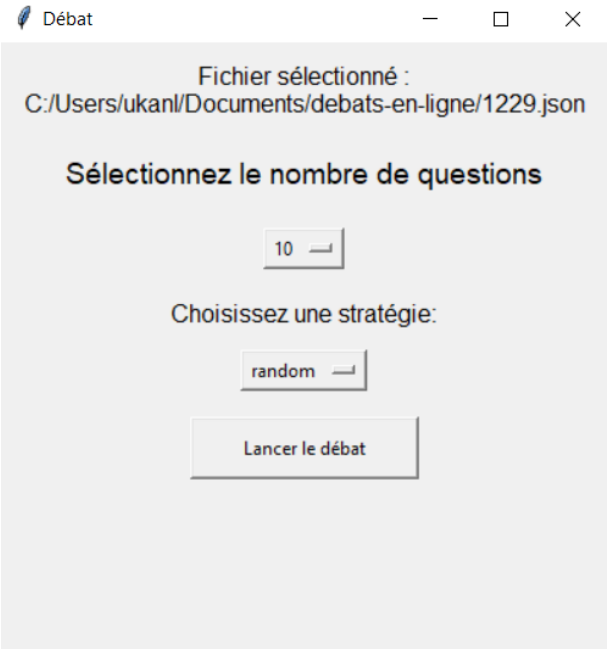

FIGURE B.2 – Aperçu de l'interface de configuration de la séquence

En cliquant sur "Lancer le débat", vous voici au cœur de l'application : la séquence d'évaluation par comparaison d'arguments.

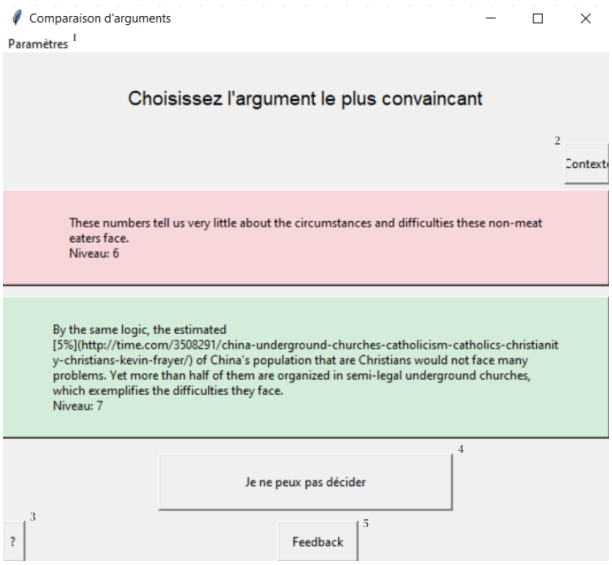

FIGURE B.3 – Aperçu de l'interface de la séquence

À chaque paire d'arguments sélectionnés selon la stratégie choisie, vous devrez choisir l'argument le plus convaincant. Cliquer sur l'option "Paramètres" (1) vous indiquera pour chacune des propositions si elle soutient ou non la thèse principale à travers un code couleur : vert (resp. rouge) pour un argument en faveur (resp. en désaccord) du point apporté par son message (nœud) parent. Pour des arguments trop spécifiques (correspondant à des feuilles du graphe), vous pourrez obtenir du contexte à l'aide du bouton correspondant (2) pour l'un ou l'autre des deux arguments. À tout moment, il vous sera

possible d'afficher à nouveau la question centrale du débat via le bouton " ?" (3). Dans le cas où vous ne pourriez pas décider, un bouton dédié (4) vous permettra de passer à la paire suivante, en ignorant la comparaison actuelle. De plus, vous aurez la possibilité d'apporter un retour (*feedback*) à chaque paire d'arguments, pour justifier vos choix ou expliquer votre incapacité à décider (5).

Une fois que la séquence est terminée, il est possible d'accéder à une interface d'analyse de vos résultats.

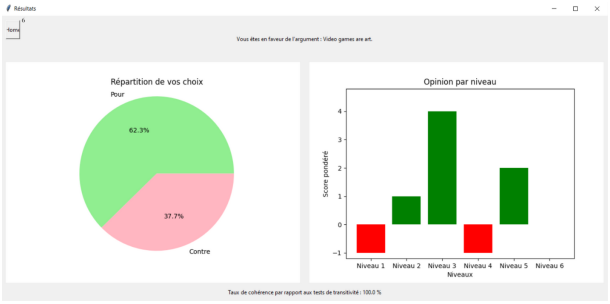

FIGURE B.4 – Aperçu de l'interface d'analyse des résultats

Cette interface vous indique votre position générale vis-à-vis du problème posé par le débat, un diagramme en camembert de la répartition de vos choix, ainsi qu'un diagramme en barre de votre opinion vis-à-vis du sujet en fonction de la profondeur des arguments. Une évaluation de votre cohérence est traduite par le taux de réussite à la transitivité. Si vous avez opté pour une stratégie tournoi, l'analyse des résultats vous proposera alors l'argument que vous avez estimé le plus convaincant et vous serez attribué sa coloration vis-à-vis du débat.

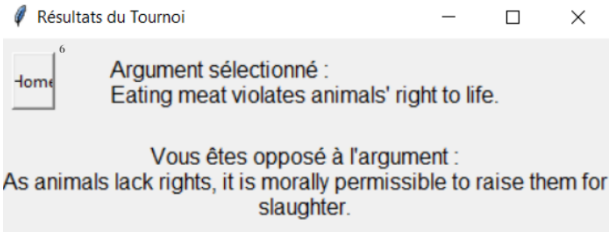

FIGURE B.5 – Aperçu de l'interface d'analyse des résultats

Si vous le souhaitez, il vous est possible de revenir à l'interface d'accueil en cliquant sur le bouton "Home" (6), et ainsi lancer la prochaine séquence de comparaisons.