Projet de Recherche :
Recommandation personnalisée de TV sociales de sport électronique.

Michalet Stefan et Moser Antoine

Février 2013

Résumé : Le sport électronique s'est développé dans la dernière décennie avec l'apparition des TVs sociales dont la plateforme la plus populaire pour les jeux vidéos est Twitch.tv. Actuellement les systèmes de recommandation sont utilisés dans l'aide à la décision de différents domaines comme l'e-commerce, les moteurs de recherche ou les sites de vidéos (Youtube) dont les TVs sociales se rapprochent en terme de services mais où le contenu évolue en temps réel. Le but de cet article est de proposer un système de recommandation personnalisée sur Twitch.tv qui s'adapte au changement du contenu et des TVs. Nous expliquons le terme TVs sociales et décrivons les systèmes existants. Un algorithme sera suggéré tout en réfléchissant aux améliorations possibles selon les données récupérables.

Mots-clés : Tv sociales, Twitch.tv, système de recommandation, sport électronique, jeux-vidéos

1 Introduction

Dans le cadre du Master 1 d'Informatique, nous avons participé à un projet de recherche pendant une durée de 5 semaines. Nous avons été encadrés par Marc Plantevit, Mehdi Kaytoue et Julien Mille, membres du LIRIS.

Aujourd'hui, la plupart des jeux proposent un mode multi-joueurs permettant à des joueurs du monde entier de s'affronter sur leur jeu favori. Certains de ces jeux, exclusivement multi-joueurs, en comptent des millions (Riot Games revendiquait 32 millions de joueurs en 2011 sur League of Legends [ref12c]). La compétition entre les joueurs s'est naturellement développée proposant à une petite communauté de joueurs de se départager dans des tournois.

L'aspect compétition dans les jeux vidéo, appelé eSport, existe depuis presque 10 ans. Il se développe mais est resté dans un milieu très réduit où seuls les joueurs recherchant la compétition sur leur jeu étaient interressés. De nombreux tournois de très grandes envergures comme la Dreamhack en Suède ou les *Intel Extreme Master (IEM)* (plusieurs étapes à travers le monde) organisés par Turtle Entertainment via leur filiale "Electronic Sports League" ont vu s'affronter les meilleurs joueurs du monde sur Counter Strike et Quake. Les dotations de certaines des compétitions peuvent atteindre les millions d'euros (2,525,775 dollars de récompense pour Starcraft 2 sur l'année 2011 [ref12a]). La grande différence entre le sport "traditionnel" et l'eSport est que celui-ci utilise les médias Web pour diffuser son contenu.

Le phénomène eSport contient, comme les sports traditionnels, des joueurs amateurs, professionnels et de simples spectateurs. Les équipes et les structures sont soutenues par des sponsors. Avec l'émergence des TVs sociales (TVs qui permettent l'interaction entre les spectateurs), l'eSport a passé un palier en élargissant et en diversifiant son public. Celui-ci n'est plus constitué exclusivement de joueurs cherchant la compétition mais de simples spectateurs préférant regarder plutôt que jouer [KSC+12b]. Elles ont interprété un rôle de catalyseur développant l'eSport de manière significative. Effectivement, l'augmentation de l'audience grâce à ce système de diffusion attirent des sponsors offrant la possibilité aux tournois d'augmenter les dotations et de proposer des shows toujours plus professionnels. En 2012, Riot Games organisait

les finales mondiales de League of Legends. Cet événement a vu les vainqueurs remporter 1 million de dollars et plus d'un million de spectateurs ont regardé simultanément l'événement sur les TVs mises en place par Riot [ref12b]. Les acteurs de l'eSport et les joueurs professionnels peuvent diffuser simplement leurs tournois et leurs sessions de jeu, certains joueurs étant même rémunérés grâce à des contrats publicitaires.

Twitch.tv est un site permettant (par la plateforme Justin.tv) le streaming de jeux video et la diffusion de compétitions vidéo ludiques. Le système de TVs sociales de Twitch permet aux utilisateurs enregistrés de diffuser leur propre contenu (en restant dans le domaine du jeu vidéo) et aussi de participer à la messagerie instantanée des autres TVs du site. Ils peuvent partager leurs TVs favorites sur les différents réseaux sociaux (Facebook, Twitter, Google+ ...). En 1 an, le site reçoit presque 20 millions de visiteurs chaque mois [ref13b]. Il n'est pas seulement dédié à l'eSport et diffuse souvent des jeux comme World of Warcraft, MineCraft ou encore Halo représentant une audience non négligeable [KSC+12b]. Certains de ces jeux sont populaires et des "marathons caritatifs" sont organisés [ref13a].

Aujourd'hui lorsqu'un utilisateur arrive sur un site proposant des TVs sociales liées à l'eSport, l'aide à la décision peut être bénéfique face à la multitude de contenu. Selon une étude réalisée sur Twitch.tv, il est montré, par l'analyse de la popularité des *streamers*, que seulement 10% d'entre eux concentrent 95% des téléspectateurs du site [KSC+12b]. Cela montre que l'audience est accaparée par une petite minorité de *streamers*. De même, les téléspectateurs sont plus enclins à regarder du contenu qui se rapproche de ce qu'ils regardent habituellement [KSC+12a]. L'audience et le contenu des TVs varient globalement peu d'où l'importance d'une recommandation permettant de diversifier les habitudes, disperser l'audience et rendre populaire d'autres types de contenu.

Dans les magasins physiques, la recommandation se fait facilement car le contenu est limité par le stockage et on ne peut pas l'adapter à chaque client. La recommandation se fait alors par les meilleures ventes. La différence avec le monde on-line s'appelle le phénomène *long-tail*. Dans un magasin en ligne le contenu et l'espace ne sont pas limités. Il est possible de proposer un nombre illimité d'articles peu connus et non trouvables en magasin. L'idée est alors de proposer à un utilisateur en particulier une liste de produits qu'il lui est susceptible d'acheter, le rendant ainsi plus populaire. Par exemple, le livre d'abord inconnu *Touching the Void* sur le site de vente d'Amazon est devenu, grâce au système de recommandation, un best-seller [RU11]. Le livre a été recommandé pour quelques personnes ayant achetées, ou qui le considéraient, la suite du livre dénommé *Into Thin Air*. *Touching the Void* est finalement devenu plus populaire. Cela prouve qu'un bon système de recommandation peut avoir de l'influence sur les habitudes des utilisateurs.

Des systèmes de recommandation existent pour du contenu figé et à la popularité exclusivement ascendante comme par exemple les vidéos de Youtube [DLL+10]. Dans le cas d'une diffusion d'un contenu en direct celui-ci n'est pas figé et la popularité de la TV peut varier au cours du temps ou ne pas être disponible (hors-ligne). La recommandation doit s'adapter aux contraintes de la diffusion en direct pour proposer du contenu en adéquation avec ce que l'utilisateur souhaite regarder.

Dans ce papier, nous présenterons un état de l'art sur les systèmes de recommandation existants et sur l'étude des TVs sociales. Puis nous exposerons une méthode à la problématique de recommandation en direct de TVs sociales sur le site Twitch.tv. Nous discuterons des améliorations que l'on peut apporter et nous conclurons sur les perspectives de notre recherche.

2 Etat de l'art

2.1 Les systèmes de recommandation

Dans la thèse de Damien Poirier [Poi11], nous pouvons voir qu'il existe différentes formes de recommandation selon les données que l'on recommande :

- **la recommandation éditoriale** : elle a pour but d'attirer l'utilisateur vers les produits les plus populaires, les nouveautés, les articles les mieux notés, ...

- **la recommandation sociale** : les recommandations sont faites par les internautes pour les autres internautes (par exemple Youtube ou Amazon).
- **la recommandation personnalisée** : elle a pour but de déterminer, pour un utilisateur particulier, les contenus ou les services qui sont susceptibles de l'intéresser.

Parmi l'ensemble des types de recommandation, la recommandation personnalisée est le domaine le plus actif du moment. Il intéresse le monde industriel pour les possibilités marketing qu'offre ce type de système.

Nous ne pouvons pas tout voir et tout connaître sur Internet, c'est pour cela que les systèmes de recommandation personnalisés peuvent guider les utilisateurs dans ses choix. La recommandation personnalisée a pour objectif de filtrer des contenus ou des items afin de garder les plus pertinents pour un utilisateur donné. L'objectif est de prédire l'opinion qu'un utilisateur portera sur les items qu'il ne connaît pas et qu'il peut apprécier. On peut connaître les goûts de l'utilisateur grâce à l'appréciation qu'il a portée sur les contenus qu'il a déjà consultés. L'auteur indique que les informations peuvent être regroupées dans une **"matrice d'usages"**. On peut prendre par exemple une matrice binaire contenant des informations de type "utilisateur u a apprécié/n'a pas apprécié l'item i". Une fois que la matrice d'usage est construite, l'objectif du moteur de recommandation est de remplir les informations manquantes. Il y a 2 types d'approches utilisés :

- **le filtrage basé sur le contenu** : chaque utilisateur possède un profil et le système compare la représentation de l'item avec le profil de l'utilisateur afin de voir si l'utilisateur peut l'apprécier.
- **le filtrage collaboratif** se base sur les appréciations données par un ensemble d'utilisateur sur les items. On peut soit comparer les utilisateurs entre eux, soit rapprocher les items appréciés par des personnes communes.

Il existe des problèmes lorsque le système n'a pas assez de données pour effectuer un filtrage de bonne qualité. Notamment, le problème du **système débutant**. Le système ne possède aucune information sur les utilisateurs et sur les items. On ne peut pas utiliser les méthodes de filtrage collaboratif car ceux-ci ne fonctionnent pas avec une matrice d'usage vide. Une solution possible est de trouver des informations descriptives des items et inciter l'utilisateur à parcourir le catalogue pour remplir la matrice d'utilité. Un deuxième problème peut apparaître, c'est le **cas du nouvel utilisateur**. Le système n'a aucune information sur lui. Une solution possible est de faire de la recommandation éditoriale pour qu'il parcourt le catalogue et permettre au système d'acquérir des informations sur cet utilisateur. Un autre problème est le **cas du nouvel item**. Si l'on fait un filtrage basé sur le contenu, il faut trouver des descripteurs pour pouvoir le comparer aux profils que l'on possède déjà. Si l'on utilise un filtrage collaboratif et si l'item n'a jamais été noté, on ne peut pas le recommander. Il faut alors le rendre visible pour qu'il puisse obtenir une note.

2.2 TV Sociales

La télévision traditionnelle est généralement associée à la passivité devant laquelle les téléspectateurs la regardent. L'article "Understanding Social Tv : a survey" [CG11] nous explique qu'aujourd'hui de nouvelles télévisions dites sociales (TVs sociales) émergent. Celles-ci ont la particularité de permettre aux téléspectateurs d'interagir avec les autres, séparés par le temps ou l'espace, tout en regardant le même contenu. Nous pouvons le faire à l'aide de différents moyens physiques (smartphone, PC ou tablettes) et différents moyens techniques (chat, Twitter, Facebook, vidéoconférence, ...). Cela amène l'article à catégoriser les TVs sociales via six aspects.

- **activité** : but de l'interaction sociale et principales tâches de l'application
- **environnement** : web, télévision traditionnelle ou smartphone
- **forme des interactions** : interaction entre les téléspectateurs (texte, audio ou vidéo)
- **représentation des utilisateurs** : liste de contacts plus ou moins évoluée

- **synchronisation des interactions** : synchrone ou asynchrone
- **portée sociale** : cercle familial, amical ou relation avec des inconnus

Une TV sociale contient différents moyens de communiquer, d'interagir et parfois de modifier son aspect. Par exemple, la plupart d'entre elles permettent une mise à jour de son statut sur les réseaux sociaux ou encore une communication direct entre les téléspectateurs souvent par le biais d'un "chat" écrit. Mais ce qui nous intéresse ici est la faculté des TVs à proposer du contenu personnalisé à un utilisateur via un système de recommandation. C'est à dire l'utilisation des informations (notes, commentaires, messages, favoris ou encore historique) des téléspectateurs pour aider à la sélection d'une chaîne.

L'article "Watch me playing, I am professionnal : a First Study on Video Game Live Streaming" [KSC+12b] nous présente une étude sur le phénomène du *Live Streaming* (e.g la diffusion en direct de contenu sur le Web) de jeux vidéo. Elle se focalise sur les jeux vidéo qui représentent un intérêt pour le sport électronique (E-sport). A l'aide des données récupérées sur le site Twitch.tv, les auteurs étudient l'audience des *streams*. La plupart sont localisés en Amérique, en Europe et en Asie de l'Est. L'analyse des données a permis de mettre en évidence le fait que le nombre de spectateurs augmente de façon significative avec les événements sportifs diffusés sur le site. Elle montre aussi que les *streams* les plus populaires (10%) contiennent 88% des spectateurs. L'étude propose deux causes qui peuvent être liées à ce phénomène : la rareté des bons *streams* ou la basse qualité du système de recommandation.

Ensuite l'article nous explique que l'on peut prédire la popularité d'un *stream* à l'aide d'un modèle de régression linéaire. L'étude montre, par l'analyse des jeux diffusés (375 sur la période d'analyse), que les plus populaires sont ceux joués de manières compétitives et présents dans l'eSport. Leur audience est régulière tandis que six, apparaissant dans la liste des 20 jeux les plus regardés et étant nouvellement sortis voient leur audience baissée au fil du temps. Il l'explique par l'envie du téléspectateur de se faire une idée avant d'acheter un jeu et par le fait que seuls les jeux dont les parties sont intéressantes à regarder ont une audience régulière.

L'article nous montre comment classer les *streamers* par popularité à l'aide de la méthode Condorcet. Elle consiste à prendre les *streamers* par paire et à regarder lequel les spectateurs préfèrent regarder. Un classement est ensuite effectué entre les paires. Il faut toutefois ne pas confondre le nombre de vues et le score de popularité d'un *streamer*. Les auteurs appliquent un pré-traitement sur le nombre de vues pour y soustraire la "popularité incorrect". Ils estiment que les *streamers* du top 100 ont 5% de leur audience qui font partie de leur "popularité incorrect".

L'article "Witnessing the Digitization of Sport through Social TV" [KSC+12a] rajoute une caractérisation de la communauté (spectateur et diffuseur) de Twitch.tv par les aspects conventionnels (qui, où, quoi, combien de fois, ...). Les auteurs concluent par l'analyse des données du chat liée à une TV que l'audience "concernée" regarde les TVs tous les jours alors que la moins "concernée" préfère regarder le week-end lorsque les événements liés à l'eSport sont diffusés.

Cet article pose alors la question de savoir si la langue est une barrière pour l'utilisateur. Il apparaît que la vaste majorité des messages sont en anglais. Grâce aux TVs sociales les frontières sont déjà brisées alors que l'eSport n'en est qu'à ses débuts. Dans le cas d'une interaction audio (commentaires du/des diffuseurs) le seul moyen de parler au monde entier est l'anglais.

Les auteurs expliquent avoir réalisé une simple analyse sur la nature des messages dans le chat. La plupart des messages sont des questions qui montrent que le chat est utilisé pour avoir des informations sur le contexte et le contenu de la vidéo.

Les favoris sont ensuite analysés. L'article précise la différence entre l'apparition en favori et l'audience. L'utilisateur ajoute facilement une TV en favori sans pour autant la suivre régulièrement. A l'inverse l'utilisateur peut suivre régulièrement une TV sans l'avoir en favori.

Pour en revenir au système de recommandation, les jeux les plus regardés, la prédiction de popularité et le système de classement peuvent nous être réellement intéressant dans notre recherche d'une solution

adéquate aux questions : quels jeux le système doit-il recommander en priorité ? Quels *streams* peuvent être plus intéressants au fil des heures ? Comment savoir si un *stream* sera plus populaire qu'un autre ? Dans quelles langues devons nous recommander un *stream* ? Comment améliorer le système de recommandation en analysant les messages des utilisateurs ?

3 Un système de recommandation de tv sociales

Notre méthode se base sur le système de recommandation utilisé par Youtube. Dans cette section, nous allons présenter le problème lié à la recommandation en direct et les données utilisées. Nous proposerons ensuite une méthode de recommandation et son implémentation. Puis nous finirons par discuter des améliorations que l'on peut apporter.

3.1 Données et problème

Même si un algorithme de recommandation existant peut être utilisé dans notre cas, il est important de souligner qu'il doit être amélioré par rapport à son environnement (ici Twitch.tv) et à la problématique énoncé ci-dessus. Effectivement nous avons vu qu'il existe différentes catégories de TVs sociales caractérisées par les interactions qu'elles proposent. Nous proposons alors une instance de catégorisation de Twitch.tv [KSC+12a]

- **activité** : Twitch.tv propose aux utilisateurs enregistrés de créer leur TV personnalisée et alors diffuser du contenu lié aux jeux vidéo.
- **environnement** : Twitch.tv est disponible sur le web et principalement utilisé sur PC. Des applications smartphones (iOS et Android) existent mais ne permettent pas les interactions entre les utilisateurs.
- **forme des interactions** : Twitch.tv permet principalement l'interaction écrite entre les téléspectateurs (mais aussi le diffuseur) via un "chat" (IRC) pour chaque TV. Ils peuvent aussi faire une mise à jour de leur statut sur les réseaux sociaux Twitter et Facebook et envoyer des messages privés aux autres utilisateurs.
- **représentation des utilisateurs** : Twitch.tv ne propose pas une liste de contact personnalisée pour chaque utilisateur. Mais pour chaque TV, nous avons accès à la liste des utilisateurs enregistrés la regardant.
- **synchronisation des interactions** : nous avons ici deux types d'interactions : les messages écrits via le chat sont synchrones alors que les messages privés et la mise à jour du statut étant asynchrones.
- **portée sociale** : sur Twitch.tv la portée sociale a tendance à privilégier les relations entre les inconnus.

Il convient alors de répondre à la problématique suivante : Comment, à l'aide des informations issues des interactions, peut on améliorer (ou créer) un algorithme de recommandation qui soit en direct, personnalisé, efficace et de qualité sur Twitch.tv ?

Le système de recommandation de Youtube [DLL+10] est conçu pour recommander du contenu figé et dont la popularité est toujours ascendante. Plusieurs problèmes peuvent émerger si l'on utilise ce système sur des éléments en direct comme sur Twitch.tv. Le contenu est variable et alors qu'à un moment l'utilisateur s'est intéressé à du contenu d'une TV particulière, il se peut qu'à un autre moment les TVs recommandées ne proposent plus ce même contenu, rendant la recommandation obsolète. Cela entraîne qu'une TV n'est pas forcement recommandable à un moment donné mais qu'elle peut l'être à un autre moment, sans pour autant changer les préférences et les habitudes de l'utilisateur. Le problème est que la recommandation ne doit plus se faire seulement sur la variation des préférences de l'utilisateur mais aussi sur la variation en continu des TVs, d'où la nécessité d'un système de recommandation en direct.

Nous avons accès à des jeux de données collectés sur Twitch.tv. Il convient de différencier les différentes entités composant des TVs sur le site : Le contenu ou l'objet qui représente le contenu de la vidéo et qui est diffusé en direct, aussi appelé *stream* associé à un outil de messagerie instantanée. Le producteur qui est le diffuseur, aussi appelé *streamer*. Et les consommateurs qui sont les téléspectateurs, aussi nommés *viewers*. Il n'est pas nécessaire d'être enregistré pour pouvoir regarder une TV, mais être connecté permet d'utiliser la messagerie, les favoris et les fonctionnalités de partage des réseaux sociaux. Dans notre cas, seul les utilisateurs enregistrés apparaissent dans les jeux de données collectés car ils sont identifiées par leur pseudo.

Nous avons en tout 5 fichiers permettant de récupérer des informations différentes. Le premier est composé de 213 n-uplets de la forme (*tv*, *lang*) et définissant (si définissable) pour chaque tv la langue du contenu diffusé. Le deuxième fichier représente les signaux IRC récupérés (*message*, *connexion* et *déconnexion*) de chaque tv formé par les champs (*user*, *tv*, *date*, *type*, *message*) et constitué d'environ 29 millions de n-uplets Table 1. Le suivant nous permet de récupérer la langue parlée par 66000 utilisateurs (toujours si définissable grâce à une API qui examine les messages postés) et est de la forme (*user*, *lang*). Ensuite nous avons, pour chaque utilisateur, les tvs qu'ils ont en favoris (comprenant les TVs de justin.tv) avec 7,5 millions de n-uplets sous la forme (*user*, *tv*). Le dernier fichier nous présente les sessions (phase entre la connexion et la déconnexion) de chaque utilisateur, triées par celui-ci. C'est à dire que pour chaque session nous avons le temps (date de début et de fin) et la ou les TVs qu'il a regardée(s). Ce fichier est composé d'environ 5,8 millions de n-uplets sous la forme (*user*, *date_ begin*, *date_ end*, *time_ ms*, *time_ m*, *tvs*) Table 2.

Table 1 – Fichier signals.clean.csv

champs	description
#tuples	29 millions
user	Nom de l'utilisateur
tv	Nom de la TV
date	Date d'emission
type	Connexion(INC), Déconnexion(OUT), Message(MSG)
message	Contenu du message, si type vaut MSG

Table 2 – Fichier session.timeout.csv

champs	description
#tuples	5,8 millions
user	Nom de l'utilisateur
date_begin	Date de début de la session
date_end	Date de fin de la session
time_ms	Temps de la session en millisecondes
time_m	Temps de la session en minutes
tvs	liste des tvs regardées pendant la session

3.2 Notre méthode

A l'aide des données récoltées, nous avons un ensemble de signaux Sig=$\{(u,t,v) \mid t \in T, v \in V, u \in U\}$ tel que U=$\{u \mid u$ est un utilisateur enregistré$\}$, T=$\{t \mid$ 2012-10-01 $\leq t \leq$ 2012-11-05$\}$ et V=$\{v_i \mid v_i$ est un *stream*$\}$ avec c_{it}=$|\{$u $\mid \exists\ (u, t, v_i) \in Sig\}|$.

Nous créons un ensemble de graphe G=$\{G_t \mid G_t$=$(V, E_t), \forall\ t \in T\}$ tel que E_t=$\{(v_i, v_j) \mid v_i, v_j \in V\}$ représente l'ensemble des arcs e_{ijt} avec une valeur w_{ijt} tel que w_{ijt}=$|\{u \mid \exists\ (u,t,v_i)$ et $(u,t,v_j) \in Sig$ et v_i

$\neq v_j\}|$.

Notre méthode s'appuie sur l'algorithme du système de recommandation de Youtube. Il est divisé en trois parties : la création des TVs "related" (Méthode Related), la création des TVs recommandables et le classement de celles-ci (Méthode recommandation).

TVs "related"

Dans cette partie, le but est de déterminer un ensemble R_i de n TVs apparentées à une TV v_i passée en paramètre (Méthode Related). On commence par créer l'ensemble des TVs v_j reliées directement à v_i par l'arc (v_i, v_j) dans le graphe (ligne 1). Pour chacune de celles-ci nous voulons calculer un score k_{ijt} de parenté. On calcule d'abord c_{ijt} qui est le nombre de co-visitation de l'arc (v_i, v_j), c'est à dire la visitation par un même utilisateur des deux TVs (ligne 5). Nous avons accès par le graphe à la valeur, qui est le nombre de vue total d'utilisateurs uniques sur une journée, des deux nœuds v_i et v_j respectivement c_{it} et c_{jt} . On a alors :

$$k_{ijt} = \frac{c_{ijt}}{f(v_i, v_j, t)} \tag{1}$$

avec f(v_{it}, v_{jt}) une fonction de normalisation qui calcule la popularité des deux vidéos sur la session de 24h. Nous utilisons une des plus simples fonctions de normalisation possible qui consiste à diviser le score de parenté à la multiplication des deux scores de popularité des TVs : $f(v_i,v_j,t) = c_{it} \cdot c_{jt}$. Ainsi plus c_i et c_j sont petits (c'est à dire que la TV n'a pas beaucoup de visiteurs au cours de la session), plus le nombre de co-visitation influe sur le score de parenté.

Nous stockons ensuite, pour chaque TV liée, son score de parenté dans un tableau associatif et on retourne par une fonction annexe le top N de l'ensemble. On n'impose pas dans notre algorithme de score minimal pour figurer dans l'ensemble final R_i renvoyé mais nous imposons, en testant par une requête à l'API Justin.tv, que la TV soit en ligne. Effectivement nous ne voulons proposer que des TVs actuellement en ligne et nous verrons plus tard les autres possibilités de restrictions possibles.

Nous pouvons alors créer un graphe orienté avec l'ensemble de ces TVs. Pour chaque paire de TV (v_i, v_j) il existe un arc e_{ij} de v_i à v_j avec $v_j \in R_i$ tel que la valeur de l'arc est donnée par (1).

TVs recommandables

Pour obtenir un système de recommandation personnalisé, nous combinons les TVs apparentées de la section précédente avec les TVs de l'ensemble S, c'est à dire les TVs que l'utilisateur a vues (avec potentiellement un oubli selon la date), "liked" ou mises en favorites. On appelle les TVs de l'ensemble les *graines*.

Pour chaque TV appartenant aux *graines*, nous cherchons donc les TVs apparentées. Nous avons alors un ensemble R_i pour chaque TV $v_i \in S$. Nous indiquons par C l'union de tous les ensembles de TVs apparentées :

$$C_1(S) = \bigcup_{v_i \in S} R_i \tag{2}$$

Nous pouvons alors, si C_1 n'est pas suffisant pour générer une recommandation suffisante et diversifiée, élargir la recherche en augmentant la profondeur. Effectivement C_1 ne permet généralement pas de faire découvrir à l'utilisateur des TVs qu'il ne connaît pas.

Pour augmenter la profondeur on utilise les TVs du précédent C_i créé pour rechercher leur TVs apparentées. Nous dénotons alors C_n qui recherche les TVs apparentées à la distance n des *graines* :

$$C_n(S) = \bigcup_{v_i \in C_{n-1}} R_i \tag{3}$$

Avec cette définition on a C_0 qui est égal à S (distance 0). Nous définissons alors l'ensemble C_{final} comme l'union de tous les C_n calculés précédemment :

$$C_{final} = (\bigcup_{i=0}^{\mathbb{N}} C_i) \setminus S \tag{4}$$

Il faut noter que nous enlevons à l'ensemble final les *graines* pour ne pas recommander les TVs qu'il a récemment vues, "liked" ou mises en favoris. Nous utilisons ensuite C_{final} pour classer les TVs dans la prochaine partie.

Classement

Dans notre méthode, il est noté qu'il faut ensuite classer les TVs du C_{final} et recommander un nombre n de TVs. Dans notre implémentation nous n'appliquons pas de classement final mais nous pourrons voir qu'il est possible d'utiliser d'autres descripteurs (par exemple la langue) pour augmenter la qualité de la recommandation.

3.3 Implémentation

Notre système de recommandation peut être découpé en plusieurs parties : le pré-traitement des données, la création du graphe et la recommandation à l'utilisateur. Il sera accessible via un site web.

Pré-traitement

A l'aide des données des 5 fichiers extraits de Twitch.tv nous réalisons un pré-traitement pour extraire deux types de profils : l'utilisateur et le graphe.

Pour faire le pré-traitement sur le graphe, il a été nécessaire de couper le fichier contenant les signaux IRC par date à l'aide d'un script AWK. Nous générons les profils des graphes contenant les nœuds et les arcs correspondant à un jour. Nous pouvons faire varier les valeurs suivantes selon les paramètres passés au script : le nombre de vues minimal d'un nœud et le nombre de co-visitation minimal d'un arc. Ces fichiers nous servirons à la création du graphe lors de la recommandation.

Pour les profils des utilisateurs, nous créons, pour chacun d'entre eux, un fichier texte incluant les favoris et l'historique avec, pour chaque TV, le pourcentage du temps qu'il y a passé par rapport au temps total. Nous verrons dans la partie discussion nous pouvons nous servir de ce pourcentage de fidélité. Ces fichiers nous serviront, pour la recommandation, à créer l'ensemble S.

Création du graphe

Le graphe est créé dans une application web Java à l'aide de la lecture d'un des fichiers texte pré-traités des dates. La création du graphe se fait régulièrement sur des sessions de 24h et non pas en temps réel comme peut l'être la recommandation. Le graphe est mémorisé du côté du serveur et est recréé à la fin de la session avec les nouvelles données récupérées. Nous générons un fichier JSON pour afficher le graphe avec la bibliothèque Javascript d3.js (Data-Driven Document).

Génération de la recommandation

Contrairement au graphe, la recommandation doit se faire en temps réel. En réalité nous choisissons un pas de temps de l'ordre de quelques minutes pour qu'elle se rafraîchisse. Nous devons rafraîchir régulièrement les TVs à recommander car celles-ci peuvent avoir changées de statut. De nouvelles TVs peuvent alors être recommandées à l'utilisateur. Notre application génère la recommandation côté serveur. Nous pouvons faire varier le top N des TVs que l'on souhaite conserver lors de la création des TVs "related".

Nous pouvons aussi changer la valeur de la profondeur pour avoir une plus grande diversité dans le choix des TVs à recommander.

3.4 Discussions

Il est possible d'améliorer la définition et l'implémentation de notre algorithme pour correspondre à la problématique de la recommandation en direct. Effectivement nous pouvons, en ajoutant et en faisant varier différents paramètres, jouer sur l'apparence du graphe et des résultats de la recommandation à l'utilisateur.

Lors du pré-traitement des données, pour créer les fichiers liés au graphe, nous réalisons deux choix : la valeur e_{ijt} d'un arc(v_i, v_j) est calculée par le nombre de co-visitation d'utilisateurs uniques sur les deux TVs grâce à l'analyse des connexions entrantes à la messagerie d'une TV. Nous pouvons nous poser la question de savoir si l'analyse des messages laissés par un même utilisateur sur deux TVs différentes lors d'une même session peut avoir un poids supérieur à une simple connexion sur ces deux TVs. Effectivement on peut dire qu'un utilisateur est plus "concerné" lorsqu'il laisse un message. Dès lors, la pondération de l'arc qui permet de déterminer la force de connexion entre deux TVs et qui est importante pour la suite de la recommandation doit être discutée et testée. Pour éviter de recommander des TVs avec un petit nombre de vues mais fortement connectées avec d'autres, il faut les filtrer par une valeur minimale de vues. De même il faut se poser la question de savoir à combien de vues une TV peut être jugée trop peu populaire.

Comme vu dans la méthode, il existe beaucoup de paramètres qui modifient le résultat de la recommandation. Lors de la création de l'ensemble S (les *graines*) qui décrit l'activité de l'utilisateur, son historique peut se réduire selon la date à laquelle la TV a été regardée. Pour aller plus loin, nous pouvons calculer un degré de "fidélité", c'est à dire un pourcentage de temps passé sur la TV selon le temps total. Ainsi on peut plus facilement "oublier" des TVs qui ont un degré faible et au contraire mettre plus de temps à supprimer une TV avec un grand pourcentage. Les *graines* permettent alors de mieux décrire l'utilisateur et son activité, rendant ainsi la recommandation plus précise.

D'autres paramètres peuvent être changés comme la profondeur de la recommandation, le calcul du score de parenté ou encore le top N lors de la création des TVs *"related"*. La profondeur permet de diversifier le contenu par rapport aux *graines* et donc par rapport à l'activité de l'utilisateur. Elle est donc importante pour ne pas toujours recommander des TVs que l'utilisateur connaît et ne doit pas être trop grande pour ne pas défavoriser des TVs plus connectées. Le score de parenté k_{ijt} entre deux TVs est quant à lui calculé par une simple fonction de normalisation qui multiplie le nombre de total de vues des deux TVs. Comme le souligne l'article sur Youtube [DLL+10], il existe beaucoup d'autres fonctions de normalisation et l'on peut aussi exiger un score minimal. On prend alors le top N qui permet de restreindre les TVs apparentés. La question de combien de TV faut-il prendre est donc légitime.

L'autre facette de la recommandation est l'utilisation de descripteurs par rapport au contenu. Il y a deux choix à faire : quels descripteurs et à quel moment ? Les descripteurs comme la langue parlée de la TV ou le jeu transmis doivent être comparés à l'utilisateur. On peut alors ne recommander que des TVs anglaises ou dans la langue de l'utilisateur mais élargir la recommandation à d'autres jeux qu'il n'a pas l'habitude de regarder. Choisir à quel moment de la recommandation il faut restreindre un ensemble de TV par les différents descripteurs est important. Faut-il, par exemple, filtrer les TVs anglaises au moment de la création de l'ensemble des TVs apparentées, au moment de l'union des TVs recommandables ou au moment du classement ? Étant donné que le contenu est variable et que le temps de *stream* d'un *streamer* varie, nous pouvons choisir de ne pas recommander une TV dont on suppose qu'elle va se terminer ou qu'une partie en cours est trop avancée pour qu'elle soit intéressante pour l'utilisateur. La recommandation doit se faire en temps réel et donc le système doit se rafraîchir dans un pas de temps de l'ordre de quelques minutes (on peut dire qu'il est raisonnable de recommander un *stream* hors ligne sur une période de 2-3 min). On se pose alors la question de savoir quelles sont les données que l'on peut récupérer en temps réel sur une TV et comment peuvent-elles influencer la recommandation. Dans notre implémentation la création du graphe et la recommandation se font tous les deux sur le serveur. Il est facile d'imaginer que

la recommandation à l'utilisateur puisse se faire côté client.

Pour permettre l'évaluation du système de recommandation, il faut savoir si les utilisateurs sont intéressés par les *streams* recommandés. On peut aussi savoir si ils les connaissent déjà ou si ils trouvent le nombre de *streams* proposés suffisant. On peut ainsi jouer sur les paramètres et améliorer le système. L'évaluation peut se faire sur un simple vote d'utilisateurs sur une période donnée.

4 Conclusion et perspectives

Nous avons vu, dans ce papier, et malgré l'existence de plusieurs systèmes de recommandation, qu'il est difficile de les implémenter dans le cadre de contenu en direct et donc variable. De même il n'est pas aisé de concevoir une application sur des données et un environnement extérieures (ici Twitch.tv). Nous proposons dans cet article une méthode et une implémentation qui fonctionne mais qui nécessite des améliorations pour être efficace (comme l'ajout de certains descripteurs et le test de plusieurs paramètres). Aussi, une évaluation aurait permis de sonder le système pour savoir si il est de qualité et si il peut avoir une influence sur les utilisateurs dans le cadre de contenu en direct. En effet nous avons réalisé l'impact que pouvait avoir un système de recommandation sur la diversité et sur l'aide à la décision alors qu'il faut rappeler que Twitch.tv comporte le problème du *long-tail* (c'est à dire que les *streamers* les plus populaires contiennent la majorité des spectateurs du site).

Ce projet de recherche nous aura apporté de la rigueur dans plusieurs domaines. La nécessité d'écrire de manière formelle une méthode ou un algorithme est plus difficile que l'on ne le croit. De même la rédaction d'un article scientifique demande beaucoup de rigueur et de recul sur le travail effectué. La manipulation d'un important volume de données est aussi compliqué. Nous remercions les encadrants Mr Plantevit, Mr Kaytoue et Mr Mille pour l'aide, le temps et les ressources qu'ils nous ont accordés.

Références

[CG11] Pablo Cesar and David Geerts. Understanding social tv : a survey. In *in Proceedings of the Networked and Electronic Media Summit (NEM Summit 2011), Torino, Italy*, September 27-29, 2011.

[DLL+10] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. The youtube video recommendation system. In *RecSys*, pages 293–296, 2010.

[KSC+12a] Mehdi Kaytoue, Arlei Silva, Loïc Cerf, Wagner Meira Jr., and Chedy Raïssi. Internal report. 2012.

[KSC+12b] Mehdi Kaytoue, Arlei Silva, Loïc Cerf, Wagner Meira Jr., and Chedy Raïssi. Watch me playing, i am a professional : a first study on video game live streaming. In *WWW (Companion Volume)*, pages 1181–1188, 2012.

[Poi11] Damien Poirier. *Des textes communautaires à la recommandation*. PhD thesis, Université d'Orléans et Université Pierre et Marie Curie - Paris 6, 2011.

[ref12a] Dotations des tournois sur les principaux jeux vidéos esport en 2011. http ://www.progamingtours.net/index.php ?/StarCraft-2/best-esports-2011.html, 2012.

[ref12b] Informations de riot games au sujet de la saison 2 du jeu league of legends. http ://euw.leagueoflegends.com/news/thank-you-summoners, 2012.

[ref12c] Riot games parle du nombre de joueurs sur league of legends. http ://euw.leagueoflegends.com/fr/news/la-communaut%C3%A9-atteint-32-millions-de-joueurs, 2012.

[ref13a] Comment twitch s'est intégré dans les jeux-vidéos. http ://www.gamasutra.com/view/news/182281, 2013.

[ref13b] Evolution de l'audience de twitch.tv. http ://www.businessinsider.com/one-year-later-after-twitchtv-2012-8, 2013.

[RU11] A. Rajaraman and J.D. Ullman. *Mining of Massive Datasets*. Mining of Massive Datasets. Cambridge University Press, 2011.

Annexes

Algorithmes

Algorithme du système de recommandation de TVs sociales

Algorithm 1 Méthode recommandation_system(Utilisateur u) : C_{final}

Construire graphe G_t de TVs avec les règles d'association
$S \leftarrow \{v_i \mid$ vues, likes, votées, favori par $u\}$
$C \leftarrow S$
$C' \leftarrow \emptyset$
$R \leftarrow \emptyset$
$C_{final} \leftarrow \emptyset$
for profondeur $n \in \mathbb{N}$ **do**
  $C' \leftarrow \emptyset$
  for each $v_i \in C$ **do**
    $R \leftarrow$ related_tv(v_i, G_t)
    $C' \leftarrow C' \cup R$
  end for
  $C \leftarrow C'$
  $C_{final} \leftarrow C_{final} \cup C$
end for
$C_{final} \leftarrow C_{final} \setminus S$
$C_{final} \leftarrow$ ranking(C_{final})
return C_{final}

Algorithm 2 Méthode related_tv(TV v_i, Graphe G_t) : R

$R' \leftarrow \{v_j \mid v_j \in G_t, \exists\ (v_i, v_j) \in E_t\}$
$c_{it} \leftarrow |\{u \mid \exists\ (u, t, v_i) \in Sig\}|$
$R'' \leftarrow \{v_j, k_{ijt} \mid v \in R'$ et $k_{ijt}{=}c_{ijt}$ / $(c_{it}.c_{jt})\}$
for each $v_j \in R'$ **do**
  $c_{ijt} \leftarrow |\{u \mid \exists\ (u,t,v_i)$ et $(u,t,v_j) \in Sig$ et $v_i \neq v_j\}|$
  $c_{jt} \leftarrow |\{u \mid \exists\ (u,t,v_j) \in Sig\}|$
  $R''[v_j]$.second $\leftarrow c_{ijt}$ / $(c_{it}.c_{jt})$
end for
$R \leftarrow$ top_score(R'')
return R

Algorithm 3 Méthode ranking(Utilisateur u, C) : C_{final}

1: **return** Classe les TVs $v_i \in C$

Algorithm 4 Méthode top_score(R'') : R

1: **return** Renvoie les n TVs $v_i \in R''$ avec le meilleur score

Déroulement de l'algorithme

Soit 9 vidéos et un utilisateur, nous devons recommander pour celui-ci 3 vidéos. On construit le graphe suivant :

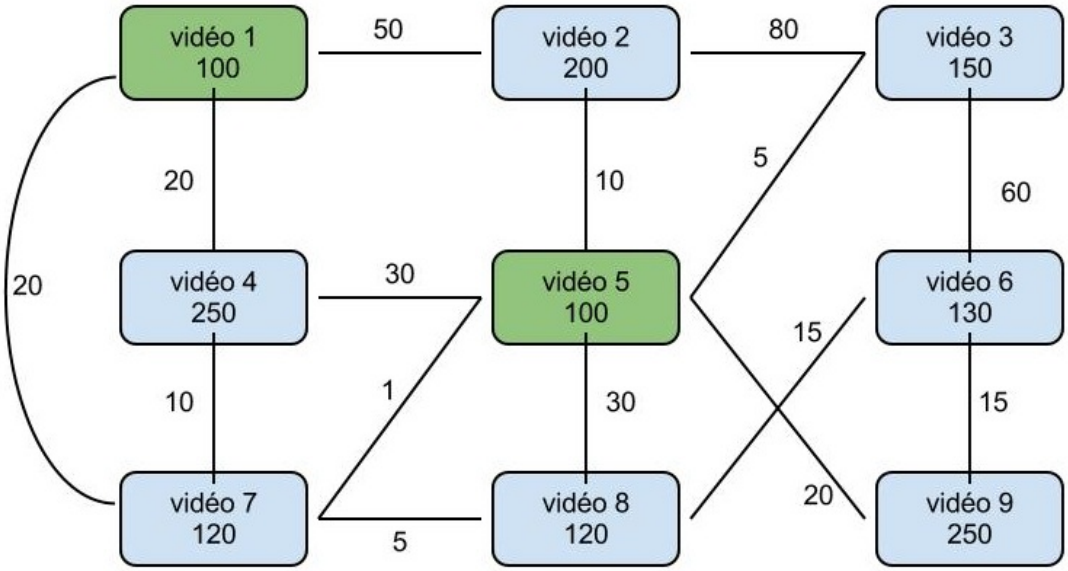

FIGURE 1 – Graphe des vidéos

Chaque nœud représente une vidéo avec le nombre de vue total. Les arcs (non orientés) représentent le nombres de vues des deux vidéos (par un même utilisateur) dans une période de 24 heures. Les vidéos avec lesquelles l'utilisateur a interagi sont en vert.

On déroule l'algorithme avec une profondeur de 2 :

- L'ensemble $S = \{1, 5\}$
- On a alors $C_0 = \{1, 5\}$
- pour chaque vidéo v_i dans C_0 on construit son ensemble R_i (nous ne voulons que 2 vidéos pour chaque R_i) :
  - On construit R_1 :
    r(1,2) = 50/ (100x200) = 0,0025
    r(1,4) = 20/ (100x250) = 0,0008
    r(1,7) = 20 / (100x120) = 0,0016
    On prend les 2 meilleurs scores ce qui nous donne $R_1 = \{2,7\}$
  - Idem pour R5 = {4, 8}
  - On fait alors l'union de tous les Ri ce qui nous donne : $C_1 = R_1 \cup R_2 = \{2,7,4,8\}$
- pour chaque vidéo v_i dans C_1 on construit son ensemble R_i

  - $R_2 = \{3, 5\}$
  - $R_7 = \{8, 1\}$
  - $R_4 = \{5, 1\}$
  - $R_8 = \{5, 6\}$
  - On a alors $C_2 = R_2 \cup R_7 \cup R_4 \cup R_8 = \{3, 5, 8, 1, 6\}$
- On fait l'union de tous les C_i. Ce qui donne $C_{final} = C_0 \cup C_1 \cup C_2 = \{2, 7, 4, 8, 3, 5, 1, 6\}$
- On soustrait C_{final} de S, $C_{final} = C_{final} \setminus S = \{2, 7, 4, 8, 3, 6\}$
- On a enfin notre ensemble final qu'il nous suffit de classer pour recommander les 3 vidéos.

Implémentation de l'algorithme

```
/**
     * Génére la recommandation pour un utilisateur
     * @param nomUtilisateur nom de l'utilisateur
     * @param fichier fichier de profil de l'utilisateur
     * @param g Graphe
     */
    public void genererRecommandation(String nomUtilisateur, String fichier, Graphe g) {
        Utilisateur utilisateur = _utilisateurs.getUtilisateur(nomUtilisateur);
        if(utilisateur == null) {
            utilisateur = new Utilisateur(nomUtilisateur);
        }
        //Lecture du fichier de profil utilisateur
        lireFichierUtilisateur(fichier, utilisateur);
        _graphe = g;

        //Creation de l'ensemble S = graines
        //On ne prend que l'historique pour le moment
        HashSet<Stream> ensembleS = new HashSet<Stream>();
        for(Historique h : utilisateur._historique) {
            ensembleS.add(h._stream);
        }
        //Initialisation des ensembles
        HashSet<Stream> ensembleC = new HashSet(ensembleS);
        HashSet<Stream> ensembleC2 = new HashSet<Stream>();
        HashSet<Stream> ensembleR = new HashSet<Stream>();
        HashSet<Stream> ensembleCfinal = new HashSet<Stream>();
        ArrayList<Stream> temp = new ArrayList<Stream>();
        //Pour une profondeur de n
        for(int i=1;i<=4;i++) {
            ensembleC2.clear();
            //Pour chaque TV dans l'ensemble C courant on calcul les TV related
            //Et on fait un Union de tous les ensembles crées
            for(Stream s : ensembleC) {
                ensembleR = new HashSet<Stream>(related_video(s, g));
                ensembleC2.addAll(ensembleR);
            }
            ensembleC.clear();
            ensembleC.addAll(ensembleC2);
            for(Stream s : ensembleC) {
```

```
                ensembleCfinal.add(s);
            }
            ensembleCfinal.addAll(ensembleC);
        }
        //Cfinal \ S
        ensembleCfinal.removeAll(ensembleS);
        //Supression des doublons !
        for(Stream s : ensembleCfinal) {
            for(Stream s2 : ensembleS) {
                if(s2._nom.equals(s._nom)) {
                    temp.add(s);
                }
            }
        }
        for(Stream s : temp) {
            ensembleCfinal.remove(s);
        }

        //Affichage reco
        for(Stream s : ensembleCfinal) {
            System.out.println("streamC : " + s._nom);
        }



    }

    /**
     * Calcule pour une TV donnée en parametre ces TVs apparentées
     * @param s stream
     * @param g le graphe
     * @return HashSet<Stream> les n meilleures TV apparentées
     */
    public HashSet<Stream> related_video (Stream s, Graphe g) {
        Noeud n = g.getNoeud(s._nom);
        if(n != null) {
            int Ci = n._nombresVue;
            double valeur;
            HashMap ensembleR2 = new HashMap();
            for(Arc a : n._voisins) {
                valeur = ((double)(a._valeur)/(a._voisin._nombresVue*Ci))*1000000;
                ensembleR2.put(a._voisin._stream, valeur);
            }
            //Calcule du top score
            return top_score(ensembleR2, 5);
        }
        return new HashSet<Stream>();
    }
```

Visualisation du graphe de co-visitation

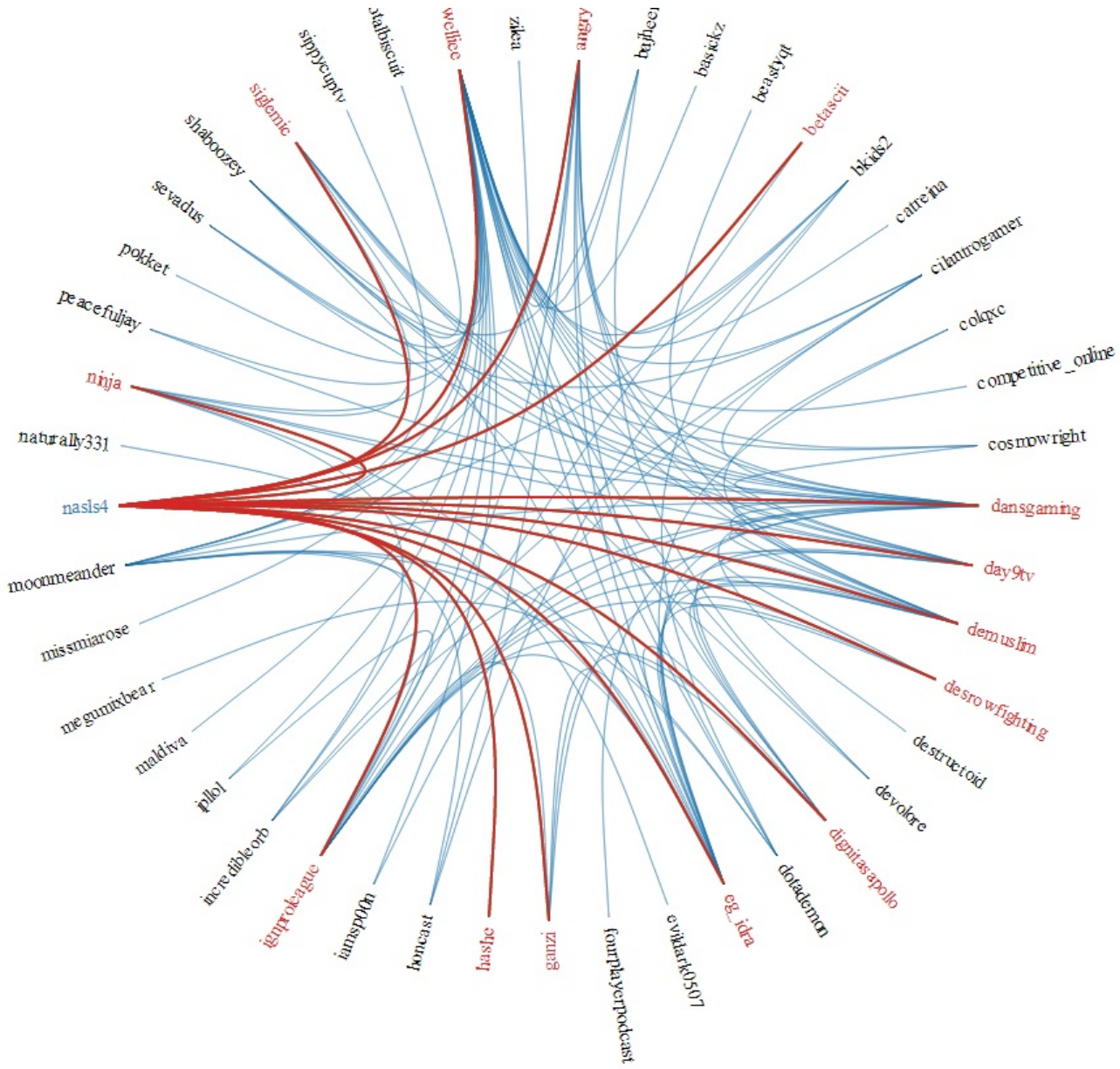

FIGURE 2 – Graphe de co-visitation des TVs lors d'une session

Fiches de lecture

Understanding Social TV : a survey [CG11]

La télévision traditionnelle est généralement associée à la passivité devant laquelle les téléspectateurs la regarde. L'article nous explique qu'aujourd'hui de nouvelles télévisions dites sociales (TVs Sociales) émergent. Celles-ci ont la particularité de permettre aux téléspectateurs d'interagir avec les autres, séparés par le temps ou l'espace, tout en regardant le même contenu. Nous pouvons le faire à l'aide de différents moyens physiques (smartphone, PC ou tablettes) et différents moyens techniques (chat, Twitter, Facebook, vidéoconférence, ...). Cela amène l'article à catégoriser les TVs Sociales via six aspects.

- **activité** : but de l'interaction sociale et principales tâches de l'application
- **environnement** : web, télévision traditionnelle ou smartphone
- **forme des interactions** : interaction entre les téléspectateurs (texte, audio ou vidéo)
- **représentation des utilisateurs** : liste de contacts plus ou moins évoluée
- **synchronisation des interactions** : synchrone ou asynchrone
- **portée sociale** : cercle familial, amical ou relation avec des inconnus

Ensuite l'article nous informe qu'une TV Sociale permet différents moyens de communiquer, modifiant son aspect, dont quatre d'entre eux sont les plus souvent utilisées.

- **la recommandation personnalisée** : c'est l'utilisation des informations des téléspectateurs pour aider à la sélection de ce que l'on va regarder. Les informations sont les notes, les commentaires et les recommandations qui peuvent être utilisées par l'utilisateur directement ou par un système. Dans le cade de notre recherche, nous nous intéressons à la recommandation personnalisée sur du contenu en direct. La recommandation directe entre les téléspectateurs (c'est à dire le partage d'un lien, d'une vidéo ou même d'une partie de celle-ci) est, selon l'article, reconnue comme étant beaucoup plus efficace que par un système automatique. Ce partage est une activité qui tend à élargir la portée sociale.
- **la communication directe** : La communication directe est présente dans beaucoup de TVs Sociales. Elle est souvent synchrone par l'utilisation d'un chat, de vidéoconférences ou d'un système vocal entre des personnes regardant le même contenu.
- **"Community Building"** : C'est l'activité asynchrone de partager des impressions et des commentaires avec une large communauté de téléspectateurs (par exemple via un compte Twitter dédié à l'a chaîne). La mise à jour du statut : Elle permet de partager, sur les réseaux sociaux, ce que l'on regarde en ce moment. L'interaction est ici asynchrone car il n'y a pas de communication directe possible.

L'article termine sur une vision vers le futur où il est possible, via des TVs connectés, de réaliser une compilation des programmes en se basant sur les préférences de l'utilisateur. Dans le futur on peut s'attendre à ce que les environnements (TV, Web, Réseaux Sociaux) convergent et s'intègrent parfaitement pour nourrir la relation entre les téléspectateurs. Ils ne consomment alors plus le contenu statique où l'offre ne correspond pas, tous les programmes étant personnalisés.

Comprendre les caractéristiques d'une TV sociale, c'est à dire les moyens qu'elle met à disposition pour interagir et de communiquer, peut nous aider à améliorer un système de recommandation en nous demandant quelles sont les interactions qui nous fournissent des informations utiles à une recommandation en direct.

Watch me playing, I am professionnal : a First Study on Video Game Live Streaming [KSC+12b]

L'article nous présente une étude sur le phénomène du Live Streaming (e.g la diffusion en direct de contenu sur le Web) de jeux vidéo. Elle se focalise sur les jeux vidéo qui représentent un intérêt pour le sport électronique (E-sport). Le site web Twitch.tv, qui est le plus connu des sites spécialisés dans le domaine du streaming de jeux vidéo, a permis grâce à son API d'avoir accès aux données telles que la description des streams (TVs) ou encore le nombre de spectateurs à un moment précis. A l'aide de ces données, l'étude analyse l'audience (par rapport au temps et à la localisation du streamer) puis étudie le problème de la prédiction de popularité d'un stream. Elle s'intéresse ensuite à la corrélation entre les jeux qui sont les plus regardés à ceux qui sont les plus vendus et les plus utilisés dans les compétitions internationales. L'étude finit par discuter d'un moyen de classer un stream par sa popularité.

On remarque que le nombre de spectateurs augmente à la fin de la semaine (vendredi, samedi et

dimanche) du à la nature des vidéos (la plupart des compétitions se jouent le week-end) alors que le nombre de spectateurs et de steamers sont synchronisés les autres jours de la semaine. La plupart des streams sont localisés en Amérique, en Europe et en Asie de l'Est. L'analyse des données a permis de mettre en évidence le fait que le nombre de spectateurs augmente de façon significative avec les événements sportifs diffusés sur le site. Elle montre aussi que les streams les plus populaires (10%) contiennent 88% des spectateurs et que 10% des streamers contiennent 95% des spectateurs (la popularité d'un stream est calculé par le pic de téléspectateurs atteint alors que celui des streams l'est par la somme de la popularité de ses streams) . Cela peut être du à deux phénomènes : la rareté des bons streams ou la basse qualité du système de recommandation.

L'article explique comment prédire la popularité des streams grâce à un modèle de régression linéaire.

Les auteurs ont étudié les jeux qui étaient les plus regardés lors de leurs analyses. Ils remarquent que 375 jeux sont diffusés lors de la période d'analyse. Ils ont regardé ensuite le nombre de vues total enregistré pour chaque jeu, permettant de voir que les jeux joués de manière compétitive et présents sur les tournois majeurs de l'e-sport sont les plus regardés. Six jeux du top 20 font partis des jeux les plus vendus de l'année. On peut l'expliquer par l'envie de se faire un avis sur un jeu récemment sorti avant un éventuel achat. Effectivement l'audience de ces streams baisse avec le temps. A l'inverse les streams dont l'audience est régulière sont ceux joués en compétition et dont les parties sont intéressantes à regarder.

L'article montre comment classer simplement les streamers en fonction de leur popularité. Il faut alors appliquer un pré-traitement en soustrayant la "popularité incorrect" du au fait que le nombre de vues n'est pas égale à la popularité. En moyenne l'étude montre que les streamers du top 100 ont 5% de leur audience qui font partie de leur "popularité incorrect" lorsqu'ils commencent à diffuser. La **méthode Condorcet** consiste à prendre les streamers par paire et à regarder lequel les spectateurs préfèrent regarder. Ensuite on corrige la "popularité incorrect" en multipliant par un coefficient selon le temps depuis que la session de stream a démarré. Un classement est alors effectué en comparant les paires.

Pour en revenir au système de recommandation, les jeux les plus regardés, la prédiction de popularité et le système de classement peut nous être réellement intéressant dans notre recherche d'une solution adéquate aux questions : quels jeux le système doit-il recommander en priorité ? Quels streams peuvent être plus intéressants au fil des heures ? Comment savoir si un stream sera plus populaire qu'un autre ?

Witnessing the Digitization of Sport through Social TV [KSC+12a]

Twitch.tv est un service de TV social dédié aux jeux vidéo. Chaque utilisateur (préalablement enregistré) peut créer une TV pour diffuser en direct un contenu lié au jeu vidéo. Comme nous l'avons remarqué sur l'article présentant les TVs Sociales, Twitch.tv est caractérisé par plusieurs aspects :

- Le service est accessible depuis n'importe quelle machine qui a accès au Web (PC, smartphone ...).
- Le service propose une grande portée social permettant l'interaction avec des inconnus.
- Les interactions sont synchronisées avec la présence d'une liste d'utilisateurs permettant l'échange de texte entre ceux-ci. On note la présence d'une mise à jour du statut de Facebook.
- Plusieurs types d'interaction entre les utilisateurs : one-to-one (entre deux utilisateurs), many-to-object (entre les utilisateurs et le contenu) et many-to-many (avec la possibilité de partager automatiquement l'activité de l'utilisateur sur Facebook).

L'article propose ensuite de caractériser la communauté (les spectateurs comme les diffuseurs) par les aspects conventionnels (qui, où, quoi, combien de fois, ...). Les analyses de l'audience dans la semaine, des caractéristiques des streams et des sessions de jeu et de la corrélation avec les événements sont les mêmes que celles mentionnées dans l'article *Watch me playing I am professionnal*. Les auteurs portent alors leurs regards sur le chat de chaque TVs, récupérant les signaux de connexion, déconnexion et d'activité dans le chat. Les graphiques montrant les moyennes obtenues dans la semaine indiquent que les connexions/déconnexions sont stables dans la semaine alors que le signal du chat augmente le week-end. Les auteurs en concluent que l'audience "concernée" regarde les TVs tous les jours alors que la moins "concernée" préfère

les regarder le week-end lorsque les événements sont diffusés.

L'article pose alors la question de savoir si la langue est une barrière pour l'utilisateur. Les auteurs portent leur attention sur le langage utilisé sur chaque TV et par chaque spectateur. Il apparaît que la vaste majorité des messages sont en anglais (89% des TVs et 94% des spectateurs). Grâce aux TVs sociales les frontières sont déjà brisées alors que l'eSport n'en est qu'à ses débuts. Dans le cas d'une interaction audio (commentaires du/des diffuseurs) le seul moyen de parler au monde entier est l'anglais. Les auteurs en concluent que l'eSport n'est pas assez développé pour former des communautés par pays, sauf en Amérique. Le français arrive en deuxième position dans la langue utilisé par les TVs. Cela peut s'expliquer par le fait que Millenium (une organisation française populaire dans l'eSport) contenait parmi ses rangs un des meilleurs joueurs du monde d'un des jeu les populaires (Stephano, Starcraft 2).

Les auteurs analysent ensuite les favoris des utilisateurs (Twitch.tv étant associé à Justin.tv, la catégorie des TVs n'est plus seulement le jeu vidéo). Il apparaît clairement que la plupart des TVs en favoris sont dans la catégorie Jeux Vidéos. L'article précise la différence entre l'apparition en favori et l'audience. L'utilisateur ajoute facilement une TV en favori sans pour autant la suivre régulièrement (cas de la catégorie de Lifecasting). A l'inverse l'utilisateur peut suivre régulièrement une TV sans l'avoir en favori (cas de la catégorie des Movies, TV show et Musics).

L'article explique avoir réalisé une simple analyse sur la nature des message dans le chat. La plupart des messages sont des questions qui montrent que le chat est utilisé pour avoir des informations sur le contexte et le contenu de la vidéo. Arrivent alors les émoticônes négatives qui prouvent que l'on exprime plus son mécontentement. Les messages contenant "*your, yours, @users*" expriment des échanges directs entre les utilisateurs alors que ceux contenant des URLs montrent que beaucoup d'informations sont échangées.

Notre système de recommandation pourra s'appuyer sur les données de cet article. Par exemple la langue des TVs recommandées pourra être en anglais et/ou dans la langue originale de l'utilisateur. Il faudra aussi faire attention aux favoris qui ne prouvent pas forcement l'attention que porte l'utilisateur à la catégorie de la TV présente.

The Youtube Video Recommendation System [DLL+10]

Youtube propose des millions de vidéos aux utilisateurs du monde entier. Actuellement, pour les utilisateurs les recommandations sont très importantes. Ils se sentent accompagnés dans leurs recherches.

Un système de recommandation doit aider les utilisateurs à trouver le contenu qui les intéressent. Il doit être mise à jour régulièrement et proposer du contenu varié tout en reflétant l'activité de l'utilisateur sur le site. L'utilisateur doit comprendre pourquoi on lui recommande une vidéo.

Pour recommander des vidéos à l'utilisateur, on peut se servir de ce qu'il a mis en favori, "liker" ou souscrit à la chaine de l'uploader (celui qui met en ligne la vidéo). Lorsqu'un utilisateur regarde une vidéo entièrement, on ne peut pas en conclure que l'utilisateur a aimé cette vidéo.

Pour appliquer leur méthode de recommandation, ils ont besoin de plusieurs données :
- **metadata** : titre de la vidéo, description
- **l'activité de l'utilisateur** : ses vidéos favorites, les chaines qu'il suit, ses interactions avec les vidéos.

Si l'on choisit de n'utiliser que les metadata, nous ne pouvons pas être sûr que ces informations correspondent exactement au contenu de la vidéo. De plus, ces informations peuvent être incomplètes, obsolètes ou incorrects. Il faut donc faire attention aux données que l'on récupère sur les vidéos.

Pour générer l'ensemble de vidéo qui sont recommandable à l'utilisateur, ils commencent par créer un graphe, un graphe correspond a une session de 24 heures. Chaque nœud correspond à une vidéo et chaque arc correspond au nombre de co-visitation des nœuds. Après chaque pair de vidéo possède un score calculé par rapport au nombre de co-visitation divisé par le nombre de vu de chaque vidéo pendant la session. Ils choisissent ensuite le top N des vidéos, ces vidéos sont choisies par rapport au score calculé précédemment.

Lorsqu'ils ont construit le graphe de co-visitation, ils font l'union de l'historique de l'utilisateur et du graphe de co-visitation.

Dans leur système de recommandation, Youtube a choisi de classer les résultats dans 3 niveaux différents : la qualité de la vidéo, les spécificités de l'utilisateur et la diversité. La qualité de la vidéo correspond au nombre de vue total ainsi que de l'appréciation des utilisateurs, le nombre de partage et le nombre de fois qu'elle a été mise en favori.

Lorsque l'on a notre liste de vidéo à recommander, on peut faire une sous liste d'éléments contenant des éléments de différentes catégories. Sur Youtube, les vidéos recommandées sont visibles sur la page d'accueil du site. Il y a aussi d'autres algorithmes présent sur celle-ci comme par exemple :

- **Most viewed** : les vidéos qui ont le plus de vue en un jour
- **Top favorited** : les vidéos qui ont été ajoutées aux favoris des utilisateurs
- **Top rated** : les vidéos qui ont reçu le plus de vote en un jour

Ils ont étudié l'impact des différents algorithmes sur une période de 21 jours. Ils ont vu qu'il y a une augmentation de 207% entre les vidéo "Most viewed" et les "recommanded".