Filtrage collaboratif explicite par analyse de sentiments à l'aveugle

Charles-Emmanuel Dias, Vincent Guigue et Patrick Gallinari

21 janvier 2020

Résumé

En recommandation, les modèles de filtrage collaboratif sont souvent qualifiés de boites noires. Pour palier ce problème, nous proposons de mettre en parallèle analyse de sentiments et filtrage collaboratif dans un schéma que nous qualifions "d'analyse de sentiments à l'aveugle". Concrètement, nous proposons, par apprentissage adverse adverse d'effectuer du filtrage collaboratif par génération de textes plutôt que via la prédiction directe de notes. Nous détaillons comment ce schéma permet, en plus d'un gain de performances, d'expliciter naturellement la recommandation.

Mots-clef : Filtrage Collaboratif, Réseaux de Neurones, Réseaux Adverses.

1 Introduction

En recommandation, les modèles de l'état de l'art étant majoritairement appris à l'aide d'interactions utilisateur-produit, ils sont souvent abstrait [KBV09]. De ce fait, les algorithmes de filtrage collaboratif manquent de transparence et sont souvent qualifiés de boites-noires [BeK14]. En effet, les seules explications proviennent généralement d'une interprétation du voisinage et sont limitées à de simples comparaisons entre produits : *nous vous proposons XXX parce que vous avez aimé YYY*. Pourtant, il est crucial qu'une suggestion soit explicité. De plus, pour inspirer confiance, un système de recommandation se doit d'être transparent [TM07].

Pour améliorer conjointement performance et transparence, il est commun d'exploiter les avis textuels associées aux notes. Une première approche consiste à créer [DGG16, CC17] des profils uniquement textuels, naturellement interprétable. Une seconde approche consiste plutôt à enrichir des profils par l'apprentissage conjoint d'un modèle prédictif et d'un modèle thématique [ML13], de langue [AKCC15, NLVM17] ou d'analyse de sentiments [DdGGG18]. Dans tous les cas, l'enjeu est double : être capable de proposer un texte support – par extraction ou génération – pour chaque suggestion tout en ayant des profils encodant efficacement les gouts des utilisateurs.

Pour obtenir à la fois des profils interprétables et des suggestions explicite, nous explorons ici l'utilisation des réseaux adverses dans le cadre du filtrage collaboratif. Il s'agit d'exploiter les capacités de génération [KLA18] et de désambiguïsassions [CDH$^+$16, LZU$^+$17] de l'espace des modèles adverses pour créer un système de recommandation transparent. Jusqu'à présent, ces techniques ont été uniquement appliquées dans le cadre de la recommandation de séquences [BPL18, HHDC18].

Pour lier texte et note, nous exploitons le cadre de l'analyse de sentiment en parallèle de la recommandation. l'objectif est de tirer profit de la similitude entre ces tâches pour améliorer le filtrage collaboratif. Dans cet article, plutôt que d'associer les tâches, nous les opposons dans un schéma d'apprentissage adverse. En ce sens, nous partons d'une observation simple : l'analyse de sentiment, lorsqu'elle prend en compte les spécificités de chaque utilisateurs et produits [CST$^+$16], est similaire au filtrage collaboratif dans sa version régressive. La seule différence est l'absence de texte. Nous pouvons alors considérer le filtrage collaboratif comme de l'analyse de sentiment "à l'aveugle".

Ce papier s'organise de la façon suivante : Dans un premier temps, nous détaillons l'état de l'art en prédiction de note. Ensuite, nous formalisons le concept d' analyse de sentiment "à l'aveugle" avant de détailler le modèle adverse considéré. Enfin, nous proposons une évaluation quantitative et qualitative de notre modèle.

2 État de l'art

Dans cette section, nous détaillons brièvement les différents concepts abordés dans ce papier : le filtrage collaboratif et les réseaux adverses.

Le filtrage collaboratif se résume concrètement à l'apprentissage de préférences directement à partir d'interactions (clics, notes) d'utilisateurs. L'hypothèse sous-jacente est que si plusieurs personnes ont des intérêts communs pour certaines choses alors ces intérêts s'étendent probablement à d'autres produits. Tout l'enjeux des algorithmes de filtrage collaboratif est d'exploiter les corrélations d'interactions afin d'inférer des similarités existantes entre produits et utilisateurs. Suite au concours Netflix [BL07], les méthodes de filtrage collaboratif par factorisation se sont imposées [KBV09, SMH07]. Il s'agit d'obtenir des profils en décomposant la matrice d'interactions en deux sous matrices ; une codant pour les utilisateurs, une pour les produits. Aujourd'hui, la plupart des algorithmes développés utilisent des réseaux de neurones [HKBT15, HLZ$^+$17] qui, grâce à une grande flexibilité de modélisation permettent la prise en compte d'un nombre croissant de facteurs. Pourtant, malgré cette prise en compte d'un vaste nombre d'informations et l'amélioration constante de la précision les systèmes de recommandations, ils sont encore souvent qualifiées de "boites noires" [BeK14].

Pour expliciter les suggestions des systèmes de recommandation, une technique consiste à exploiter les avis textuels en plus des notes. Concrètement, il s'agit, en plus des suggestions, de proposer un texte support, détaillant – de manière personnalisée – les avantages d'un produit. Une première approche consiste à simplement utiliser le texte comme un objectif auxiliaire. [AKCC15, NLVM17] proposent de directement prédire l'avis en plus de la note à l'aide d'un réseau de neurones récurrents. En pratique, en plus de permettre de régulariser les profils appris [AKCC15], cela permet d'expliciter les suggestions émises par le système. A l'inverse, [ZNY17, CC17, DGG16] proposent de construire directement chaque profils utilisateur/produit à partir de l'ensemble de ses avis textuels tout en apprenant une affinité textuel. Enfin, [DdGGG18] propose de lier analyse de sentiment et filtrage collaboratif afin d'extraire les biais textuels. Ces modèles permettent, par extraction, d'obtenir du texte support.

Les réseaux adverses ont permis d'obtenir de très bon résultats sur une multitudes de tâches génératives. Ces modèles s'inspirent de la théorie des jeux : un générateur et un discriminateur s'améliorent au cours du temps grâce à une compétition.

Ces deux modèles sont entraînés dans un jeu à somme nul : $\mathcal{G}$ apprend à tromper $\mathcal{D}$ qui lui essaye d'apprendre justement à ne pas être trompé. Dans leurs formulation originale, il s'agit de d'entraîner parallèlement :

- le discriminateur $\mathcal{D}$ à faire la différence entre les vrais et faux examples en maximisant à la fois $\mathbb{E}_{x\sim p_r(x)}[\log \mathcal{D}(x)]$ et $\mathbb{E}_{z\sim p_z(z)}[\log(1-\mathcal{D}(\mathcal{G}(z)))]$
- le générateur $\mathcal{G}$ à minimiser la capacité de $\mathcal{D}$ à repérer ses exemples $\mathbb{E}_{z\sim p_z(z)}[\log(1-\mathcal{D}(\mathcal{G}(z)))]$

Il convient de noter que les réseaux adverses, surtout dans leurs formulation d'origine, sont notoirement difficile à entraîner [[GPAM$^+$14]. Ces modèles sont particulièrement sensibles aux hyper-paramètres et à l'équilibre générateur/discriminateur. Cependant, leur principal intérêt réside dans leurs capacité à générer des données ayant l'air *a priori* vraie [KLA18]. Ces modèles sont principalement utilisés en image, domaines où leurs résultats sont les plus spectaculaires. Dans le cas du texte, l'espace étant discret, il est plus compliqué d'apprendre des GAN. Néanmoins, certaines extensions existes [FGD18]. De plus, contrairement aux modèles à base de réseaux de neurones récurrents qui n'offrent aucunes garanties, l'intérêt principal des réseaux adverses est que, dans certaines conditions, l'espace appris est non-ambiguë [LZU$^+$17, CDH$^+$16]. Cette propriété permet de retrouver certain facteurs latents, comme la présence où l'absence de lunettes par exemple, rendant l'interprétation directe.

Dans le cadre de ces travaux, c'est ces deux capacités de génération et de désambiguïsation qui nous intéresse. Celles-ci permettraient respectivement de rendre les système de recommandation plus explicite et transparent. En effet, il serait à la fois possible de générer des explications tout en pouvant interpréter pourquoi celles-ci ont été générées.

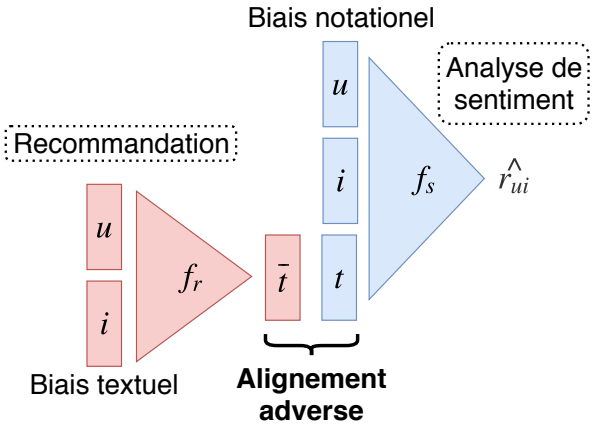

FIGURE 1 – Représentation schématique du concept d'analyse de sentiment à l'aveugle. Le modèle de recommandation (f_r, à gauche) a pour objectif de prédire l'avis t utilisé par le modèle d'analyse de sentiment (f_s, à droite) dans la prédiction de la note r_{ui} de l'utilisateur u sur le produit i. L'enjeu pour f_r est d'arriver à générer un texte $\bar{t}$ qui soit le plus proche possible du texte original t afin que f_s prédise la bonne note $\hat{r_{ui}}$.

3 Analyse de sentiment à l'aveugle

Ici, notre objectif est d'ancrer le filtrage dans un espace latent explicite et non-ambigu. Concrètement, il s'agit d'obtenir à la fois des profils latents – d'utilisateurs et de produits – interprétables et des suggestions explicites. Dans ce papier, pour atteindre cet objectif, nous proposons de mettre en compétition analyse de sentiments et filtrage collaboratif dans le cadre de l'apprentissage adverse. En effet, en plus de leurs capacités génératives [KLA18], les réseaux adverses permettent d'obtenir – en fonction de leurs formulations – des espaces non-ambigüs [CDH+16, LZU+17].

Pour mettre le filtrage collaboratif dans un cadre d'apprentissage adverse nous le considérons comme de l'analyse de sentiments "à l'aveugle" (fig. 1). En pratique, la seule différence entre analyse de sentiment et filtrage collaboratif est l'absence de texte, les avis utilisateurs n'existant pas a priori. Nous proposons alors d'utiliser – comme modèle de recommandation – un modèle de prédiction de texte, l'apprentissage adverse servant à aligner les textes prédits avec les textes réels. Le texte prédit sera utilisé par un modèle d'analyse de sentiments qui devra, en plus de prédire de la polarité du texte, savoir si celui-ci est légitime. Le principal avantage de cette modélisation est que le modèle de génération de texte, comme dans [NLVM17], représente une manière intégrée d'interpréter les suggestions du modèle.

Concrètement, nous proposons de décomposer la tâche de filtrage collaboratif en deux sous-tâches : la prédiction d'un texte $t_{u,i}$, puis de la note associée $r_{u,i}$ (fig. 3). Nous postulons que cette décomposition évitera au modèle de se focaliser sur le problème dit de "complétion de matrice" et sur les biais de notation. En effet, à l'inverse, pour optimiser son objectif le modèle devra générer un texte fidèle à l'original il devrait plutôt encoder les biais d'écriture, ce qui nous intéresse pour expliciter les suggestions. Les biais de notations seront eux captés par le modèle d'analyse de sentiment.

$$(u,i) \xrightarrow{f_r} t_{u,i} \xrightarrow{f_s} r_{u,i}$$

FIGURE 2 – Décomposition du filtrage collaboratif en deux temps : prédiction de texte $t_{u,i}$ puis de note $r_{u,i}$

Le modèle proposé en figure 1 est simplement conceptuel. En pratique, dans la formulation classique des réseau adverses, avoir un générateur paramétrique ne semble pas possible. De plus, la génération de texte via les réseaux adverses – discret par nature – ne fonctionne pas aussi bien que celle des images et reste à l'état de concept. Cependant, dans la section suivante nous proposons une instanciation particulière de ce modèle qui permet de s'abstraire de ces problèmes.

4 Modèle adverse proposé

Formellement soit des tuples utilisateur, produit, note, texte (u, i, r, t) comme données. Nous avons deux tâches : l'analyse de sentiment et le filtrage collaboratif. L'objectif de la première tâche est de prédire r, à l'aide du texte t mais aussi avec les profils u, i. Pour la seconde, l'objectif final est le même ; prédire la note mais seulement avec les profils (u, i). Dans cette contribution, nous proposons de mettre les deux tâches en compétitions : Au lieu de simplement prédire la note directement depuis les profils, nous inférons d'abord un clone $\bar{t}$ du texte original t qui sera utilisé à sa place dans le prédicteur de sentiment. La prédiction d'une note r_{ui} se fera donc de la manière suivante :

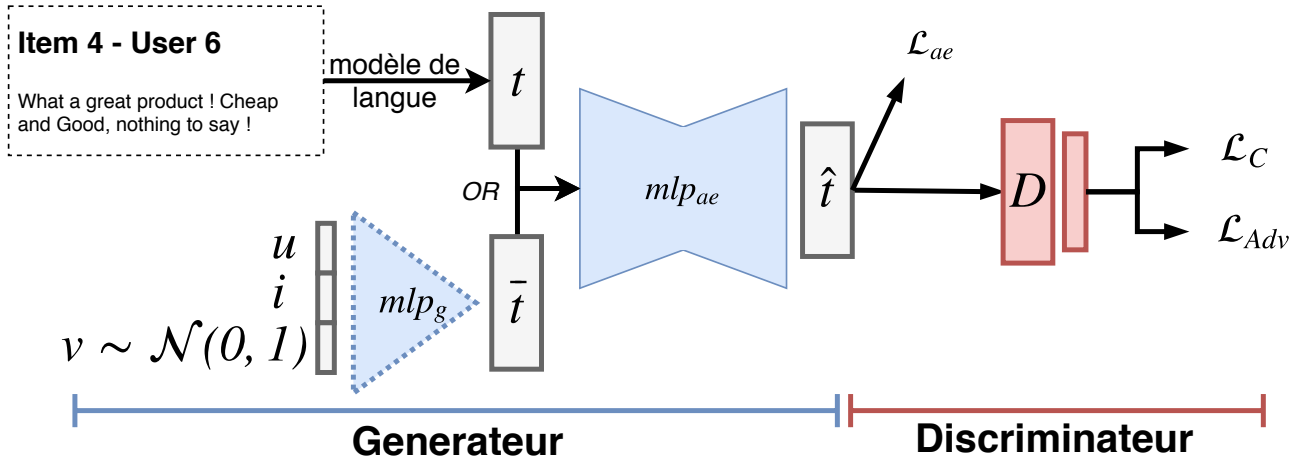

FIGURE 3 – (à gauche) Generateur : (a) Génère une représentation clone $\bar{t}$ à partir d'une paire de représentations (utilisateur,produit). (b) Auto-encode t ou $\bar{t}$ en $\hat{t}$. (à droite) Discriminateur : Predit le sentiment et la source de $\hat{t}$

$$r_{ui} = \underbrace{f_s(\overbrace{f_r(u,i)}^{\text{Recommandation}})}_{\text{Analyse de Sentiment}} \tag{1}$$

Par nature, le texte pose deux problème quant à l'utilisation des réseaux adverses tel que défini par [GPAM+14]. Déjà, il n'est pas utilisable en l'état au sein d'un réseau de neurone et nécessite une vectorisation. De plus, la génération de données discrète empêche le bon apprentissage des générateur. Ici, afin de nous abstraire de ces contrainte et par simplicité, nous proposons dans un premier temps d'encoder le texte à l'aide de *word2vec* [MCCD13]. Concrètement, nous apprenons dans un premier temps des représentations pour chaque mots de l'ensemble d'apprentissage. Ensuite, chaque avis est représenté comme le barycentre de ses représentation de mots :

$$t = \frac{1}{n}\sum_{i=1}^{n} w_i \tag{2}$$

avec $w_i \in \mathbb{R}^d$ les représentations de mot issue de *word2vec*. Par la suite, lorsque nous faisons référence à t, il s'agit donc d'une représentation $t \in \mathbb{R}^d$ et non d'un texte brut.

Notre modèle adverse (fig. 3) est composé de deux sous-reseaux – un générateur et un discriminateur – optimisés à tour de rôle dans un schéma d'apprentissage adverse. Nous détaillons d'abord séparément les fonctions de chaque réseau avant d'expliciter les coûts optimisés.

Générateur : Le premier sous-réseau – f_r – fait office de générateur. Il est lui même composé de deux modules : un auto-encodeur mlp_{ae} et un perceptrons multicouches mlp_g qui génère un texte clone $\bar{t}$ à partir d'un triplet (u, i, v) ; avec v est un vecteur de "bruit". Génerer de but en blanc du texte est complexe. Pour aider mlp_g, l'auto-encodeur apprend la distribution des données en auto-encodant le texte réel t en une représentation $\hat{t}_r$. De ce fait, mlp_g est simplement entraîné à générer un texte "reconstructible" $\hat{t}_f$, indissociable de $\hat{t}_r$; ce texte devant logiquement être t.

$$u, i, v \in \mathcal{R}^d, \qquad \bar{t} = mlp_g(u, i, v), \qquad \hat{t} = mlp_{ae}(t \vee \bar{t}) \tag{3}$$

Formellement, le générateur minimise trois coûts séparés :

($\mathcal{L}_{\mathbf{ae}}$) Tout d'abord, le générateur doit être capable d'auto-encoder un texte originale t. Pour cela, il doit minimiser l'erreur de reconstruction. Ici, nous utilisons l'erreur quadratique comme erreur de reconstruction : $MSE(t, \hat{t}_r)$ [1]. En apprenant à auto-encoder les données réelles, le générateur apprend la distribution des données [GLTFS87]. En parallèle, les clones générés doivent également être "reconstructibles". Nous minimisons

1. $MSE(x, \hat{x}) = (x - \hat{x})^2$

donc également l'erreur de leur reconstruction $MSE(\bar{t}, \hat{t}_f)$. De cette façon, nous garantissons que tous les $\hat{t}$ proviennent de la distribution des données apprises.

($\mathcal{L}^{\mathbf{g}}_{\mathbf{Adv}}$) Cependant, rien ne garantie que les reconstructions de texte généré $\hat{t}_f$, et celles de texte réel $\hat{t}_r$ soit similaire. Pour contraindre le générateur à s'aligner sur les données réelles, les reconstructions doivent être indissociables. Cette propriété est apprise grâce à une optimisation min-max où le générateur et le discriminateur on des objectifs opposés $\mathcal{L}^g_{adv}$ et $\mathcal{L}^d_{adv}$. Concrètement, le générateur, via l'auto-encodeur, va essayer de faire passer la reconstruction du texte réelle $\hat{t}_r$ pour la reconstruction du clone $\hat{t}_f$ et vice-versa. Formellement, le coup minimisée est l'entropie croisée binaire : $BCE(mlp_d(\hat{t}_r), 0)$ et $BCE(mlp_d(\hat{t}_f), 1)$ [2]

($\mathcal{L}_{\mathbf{C}}$) Le sentiment r lié aux reconstuctions $\hat{t}_r$ doit être correctement prédit. Cela ce fait dans le cadre d'une minimisation simultanée de la log-vraisemblance par le générateur.

Finalement, la fonction de coût du générateur est la suivante :

$$\begin{aligned}\mathcal{L}_g &= \mathcal{L}_{ae} + \mathcal{L}^g_{adv} + \mathcal{L}_C \\ &= MSE(t, \hat{t}_r) + MSE(\bar{t}, \hat{t}_f) \\ &+ BCE(mlp_d(\hat{t}_f), 1) + BCE(mlp_d(\hat{t}_r), 0) \\ &+ CCE(mlp_d(\hat{t}_r), r)\end{aligned} \tag{4}$$

Où CCE et BCE veulent respectivement dire Categorical et Binary Cross Entropy.

Discriminateur : Le second sous-réseau – f_s – est composé d'un perceptron multicouches mlp_d à deux têtes. Il fait à la fois office de dicriminateur et de classifieur : Il doit arriver à prédire correctement les polarités associées aux reconstructions $\hat{t}$ tout en arrivant à faire la différence entre celles provenant d'un texte légitime t ou d'un clone $\bar{t}$.

Formellement, le discriminateur minimises donc deux fonctions de coût distinctes :

($\mathcal{L}_{\mathbf{C}}$) Dans un premier temps, le discriminateur, qui fait de l'analyse de sentiment, doit être capable de prédire correctement de la polarité d'une reconstruction $\hat{t}$

($\mathcal{L}^{\mathbf{d}}_{\mathbf{adv}}$) Enfin, a l'inverse du générateur, le discriminateur doit réussir à faire la différence entre une reconstruction issue d'un texte généré $\bar{t}$ et une issue d'un texte original t. Cette compétition entre générateur et discriminateur doit forcer le générateur à mêler $\bar{t}$ et t au sein d'une même reconstruction $\hat{t}$ et, donc, aligner t et $\bar{t}$.

Finalement, avec $\mathcal{L}_C$ l'entropie croisée et $\mathcal{L}^d_{adv}$ la fonction de coût adverse, le coût minimisé est le suivant :

$$\begin{aligned}\mathcal{L}_d &= \mathcal{L}_C + \mathcal{L}^d_{adv}. \\ &= CCE(c, r) + BCE(\hat{t}_f, 0) + BCE(\hat{t}_r, 1)\end{aligned} \tag{5}$$

5 Données, prétraitements & hyperparamètres

Pour rappel, dans cette contribution nous cherchons à exploiter les techniques d'apprentissage adverses avec un double objectif : **(a)** Apprendre les profiles utiles permettant des **(b)** recommandations compréhensibles. Pour évaluer la faculté de notre modèle à atteindre ces deux objectifs, nous proposons plusieurs expériences. Après avoir détaillé les données que nous utilisons pour celles si, nous proposons une évaluation quantitative et qualitative des profils obtenus.

Hyperparametres Lors de nos expériences, l'architecture de notre réseaux est resté constante. Les profils utilisateur, produit ainsi que le vecteur de bruit sont tous de taille identique : $u, v, i \in \mathbb{R}^5$. Ils sont volontairement petit pour eviter le sur-apprentissage. Enfin, la taille des représentation des mots est aussi fixé : emb_size = 100. Les deux fonctions de coûts L_g et L_d sont toutes deux optimisées séquentiellement en utilisant le schéma d'optimisation Adam.

Données et prétraitements Pour nos experiences, nous utilisons des avis consommateurs provenant d'Amazon [MPL15]. Nous choisissons cinq datasets aléatoirement, de différentes tailles, sur différents types d'objets. Les propriétés des bases de données sont répertoriés Table 1. Ici, n'évaluant pas le démarrage à froid, seul les utilisateurs et les produits avec un minimum de 5 avis sont conservés. Les représentations des 10 000 mots les plus fréquents apparaissant au minimum cinq fois sont pré-apprises en utilisant *word2vec*. Afin d'effectuer de la validation croisée, chaque base de donnée est divisé en cinq parts égales. Quatre partie servent pour l'entraînement (80%) et une pour la validation (10%) et l'évaluation (10%). Ce schéma de 80/10/10 est commun dans l'évaluation de système utilisant le texte [ML13, AKCC15].

2. $BCE(x, y) = -[y \cdot \log x + (1 - y) \cdot \log(1 - x)]$

Dataset (#reviews)	Mean (μ)	w/offset	MF	HFT	BSA
Instant Video (37.126)	1.25	1.137	1.024	0.933	**0.924**
Digital Music (64,706)	1.19	0.965	0.903	0.844	**0.832**
Video Games (231,780)	1.45	1.281	1.267	1.097	**1.082**
CSJ (278,677)	1.215	1.323	1.365	1.107	**1.101**
Movie (1,697,533)	1.436	1.148	1.118	1.020	**1.009**

TABLE 1 – Erreur quadratique moyenne en prédiction de note. Les modèles de références sont : La moyenne globale de la base de donnée μ, le bias de notation utilisateur-item global et un algorithme de factorisation matricielle commun [KBV09]. Les valeurs présentées sont des moyennes obtenues par validation croisée sur 5 parties

6 Evaluation en prédiction de note

Nous évaluons dans un premier temps la tâche classique de filtrage collaboratif : la prédiction de note. Cela permet d'évaluer quantitativement les profils appris. Pour prédire la note d'un couple (utilisateur,item) r_{ui} un texte clone $\bar{t}$ est d'abord généré par mlp_g. Ensuite, ce texte est auto-encodé puis classifié. Comme algorithmes de références nous choisissons deux modèles commun : un algorithme de factorisation matricielle [KBV09] et un modèle enrichissant les profils obtenue par factorisation avec le texte - HFT [ML13]. Comme souvent, nous ajoutons les moyennes objet et utilisateur qui sont déjà de bonnes références. L'erreur quadratique moyenne est reportée Table 1.

De façon surprenante, notre modèle, bien que non directement entrainé à optimiser la prédiction de note, obtient de meilleurs performance que les modèles de références qui le sont. Cela confirme, a priori, l'intérêt de modéliser le texte au lieu des notes. Aussi, de tels résultats montre que les profils appris contiennent des informations

7 Explicitation des suggestions

Pour expliciter les suggestions, il est souvent proposé de générer des avis en plus des notes [NLVM17]. Ici, Le modèle présenté n'est pas explicitement entrainé pour cela et il n'est pas capable de générer des textes mots à mots. Cependant, il génère des représentations de mots qui devraient être similaire à la représentation réelle (celle-ci étant une moyenne de mots). Il est donc possible, en comparant la représentation générée $\bar{t}$ avec toutes les représentations de phrases existantes, de créer un avis par extraction. Formellement soit un couple (u, i) cible de la recommandation, $\overline{t_{ui}}$ la représentation généré et $s_{*i} = \{t_{*i}\}$ l'ensemble des représentations des phrases pré-existantes sur l'item i. En utilisant la similarité cosinus (eq. 6) entre $\overline{t_{ui}}$ et toutes les phrases t_{*i}, il est possible d'extraire celles les plus proches de la représentation généré.

$$cos(u, v) = \frac{u.v}{||u||.||v||} \in [-1, 1] \tag{6}$$

Un exemple de génération d'avis par extraction est donné Fig. 2. Bien qu'il soit impossible de tirer des conclusions d'un seul exemple, celui-ci montre clairement qu'on est capable d'extraire des phrases pertinentes.

8 Visualisation des caractéristiques apprises

Certains réseaux adverses permettent d'obtenir des espaces non-ambigus [CDH+16], où chaque dimension code pour une caractéristique particulière. Par exemple, [LZU+17] montre qu'il est possible d'ajouter des lunettes à quelqu'un en modifiant une composante latente du réseaux adverse. Attester de la non-ambiguïté d'un espace latent repose principalement sur l'observation des exemples générés. Dans notre cas, cette visualisation est moins évidente. En effet, nous générons un vecteur latent codant pour l'intégralité d'un avis et récupérons des mots/phrases existantes par similarité cosinus.

Pour essayer de visualiser l'effet d'une composante sur la génération, nous procédons de la sorte. Nous choisissons un couple (utilisateur,produit) au hasard et fixons toutes les composantes du vecteur "bruit" v à 0. A partir de cette entrée (u, i, v), nous générons un vecteur latent généré $\bar{t}_0$. Ce vecteur étant théoriquement dans le même espace que celui des mots, nous récupérons les n mots les plus proches de $\bar{t}_0$. Nous considérons cet ensemble $\{w_0\}$ comme notre base. A partir de

Vérité Terrain	Phrases les plus proches
I already played Bioshock.	Great sequel !
The game is more interesting.	This game has a great storyline !
The levels are bigger and more beautiful.	Great environment for an FPS.
There are new characters, new weapons.	For the most part this game is beautiful.
Enemies are clever and hard to kill .	Highly recommended for anyone that likes FPS games.
In conclusion this game is fantastic and beautiful.	I love the use of color and the art in general.

TABLE 2 – Exemple de prédiction par extraction à l'aide de la similarité cosinus entre le texte généré $\bar{t}$ et des phrases existantes.

là, nous répétons la procédure en modifiant une seule composante du vecteur de bruit, de 0 à 0.5 puis à 1 et de 0 à -0.5 puis à -1. A chaque fois, nous récupérons l'ensemble des mots les plus proches du vecteur généré : $\{w_{-0.5}\}\{w_{-1}\}\{w_{0.5}\}\{w_1\}$. Finalement, nous visualisons les différences entre notre ensemble base $\{w_0\}$ et tous les autres. En effet, les mots ajoutés sont liées au vecteur de bruit. La figure 4 représente tous ces mots pour la quatrième composante. Bien que dur à interpréter, changer la composante semble changer la polarité globale des mots.

9 Discussion

Ici, nous nous sommes intéressés à rendre le filtrage collaboratif plus explicite. En effet, pour être accepté, une suggestion se doit d'être accompagné d'une explication. Seulement, si l'explication est purement générée – et donc tout droit sortie d'une boite noire – elle nécessiterais elle-même une explication. En réalité, tout l'enjeu est d'obtenir un système suffisamment transparent et scrutable pour que l'utilisateur lui fasse confiance. Pour améliorer la transparence des systèmes de recommandations tout en restant dans le cadre classique du filtrage collaboratif, nous avons principalement proposé d'utiliser les réseaux adverses dans un cadre d'analyse de sentiment à l'aveugle.

Théoriquement, l'avantage d'un modèle adverse est qu'il peut combiner génération et espace latent non-ambigu. Ici, pour des raisons pratiques, nous utilisons des représentation pré-apprise et n'avons pas utilisé un décodeur directement textuel. De plus, l'architecture proposé, du fait de la personnalisation du générateur, ne permet probablement pas un apprentissage optimal de l'espace latent.Cependant, le modèle présenté ici, malgré ses défauts, fait figure de preuve de concept. Bien qu'incapable de générer directement du texte, il montre que l'exploitation de ces textes dans un cadre adverse permet notamment, en plus d'améliorer les performances, d'extraire du texte support – des mots, des phrases ou encore des avis entiers – pour aider l'utilisateur dans sa décisions.

Remerciements : ce travail a été financé en partie par le projet FUI BInD.

Références

[AKCC15] Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. Learning distributed representations from reviews for collaborative filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, 2015.

[BeK14] Shay Ben-elazar and Noam Koenigstein. A Hybrid Explanations Framework for Collaborative Filtering Recommender Systems. In *RecSys 2014*, 2014.

[BL07] James Bennett and Stan Lanning. The Netflix Prize. *KDD Cup and Workshop*, pages 3–6, 2007.

[BPL18] Homanga Bharadhwaj, Homin Park, and Brian Y Lim. Recgan : recurrent generative adversarial networks for recommendation systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 372–376. ACM, 2018.

[CC17] Rose Catherine and William Cohen. Transnets : Learning to transform for recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 288–296, New York, NY, USA, 2017. ACM.

[CDH+16] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan : Interpretable representation learning by information maximizing generative adversarial nets. In *Ad-*

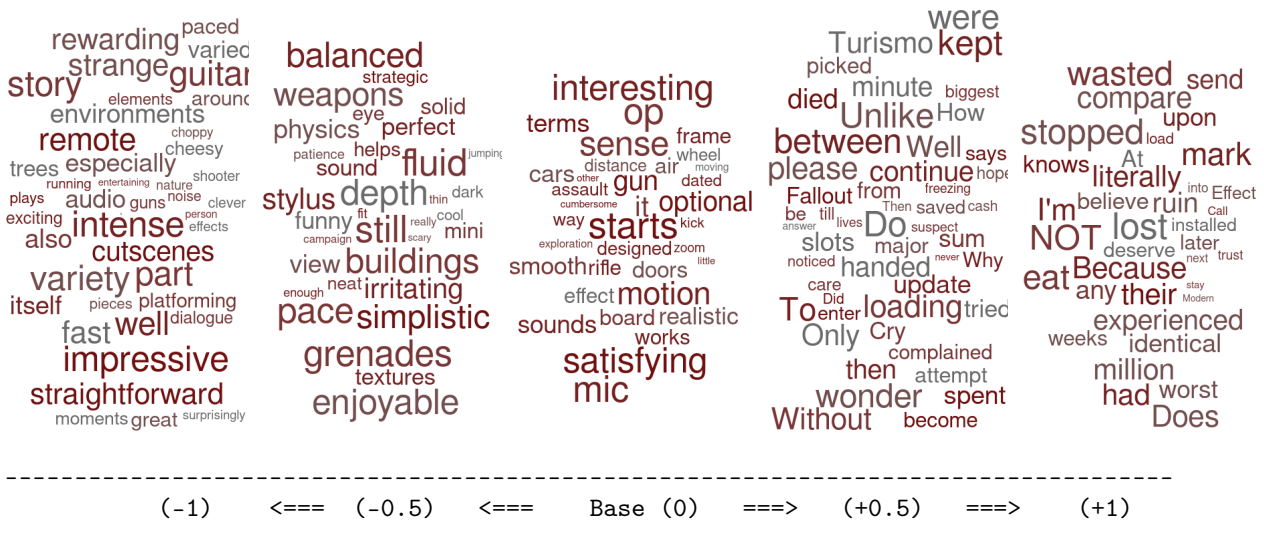

FIGURE 4 – Nuage de mots changeant en fonction d'une seule dimension du vecteur de bruit v

vances in neural information processing systems, pages 2172–2180, 2016.

[CST+16] Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1650–1659, 2016.

[DdGGG18] Charles-Emmanuel Dias, Clara Gainon de Forsan de Gabriac, Vincent Guigue, and Patrick Gallinari. Rnn & modèle d'attention pour l'apprentissage de profils textuels personnalisés. *Proceedings of CORIA*, 2018.

[DGG16] Charles-Emmanuel Dias, Vincent Guigue, and Patrick Gallinari. Recommandation et analyse de sentiments dans un espace latent textuel. In *CORIA-CIFED*, pages 73–88, 2016.

[FGD18] William Fedus, Ian Goodfellow, and Andrew M Dai. Maskgan : Better text generation via filling in the _. *arXiv preprint arXiv :1801.07736*, 2018.

[GLTFS87] Patrick Gallinari, Yann LeCun, Sylvie Thiria, and Francoise Fogelman-Soulie. Memoires associatives distribuees. *Proceedings of COGNITIVA*, 87 :93, 1987.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[HHDC18] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. Adversarial personalized ranking for recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 355–364. ACM, 2018.

[HKBT15] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv :1511.06939*, 2015.

[HLZ+17] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.

[KBV09] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42 :42–49, 2009.

[KLA18] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv :1812.04948*, 2018.

[LZU+17] Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, et al. Fader networks : Manipulating images by sliding attributes. In *Advances in Neural Information Processing Systems*, pages 5967–5976, 2017.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. *Nips*, pages 1–9, 2013.

[ML13] J McAuley and J Leskovec. Hidden factors and hidden topics : understanding rating dimensions with review text. *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13*, pages 165–172, 2013.

[MPL15] Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2015.

[NLVM17] Jianmo Ni, Zachary C Lipton, Sharad Vikram, and Julian McAuley. Estimating reactions and recommending products with generative models of reviews. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, volume 1, pages 783–791, 2017.

[SMH07] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[TM07] Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. *Proceedings - International Conference on Data Engineering*, pages 801–810, 2007.

[ZNY17] Lei Zheng, Vahid Noroozi, and Philip S Yu. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 425–434. ACM, 2017.