Traduction d'un jeu de données de dialogues en français et Détection d'émotion à partir de texte

Pierre-Yves Genest[*,**], Laurent-Walter Goix[*], Előd Egyed-Zsigmond[**]

[*]Alteca, 88 Boulevard des Belges
69006 Lyon
{pygenest, lwgoix}@alteca.fr
[**]Université de Lyon, LIRIS UMR 5205 CNRS
Bât. Blaise Pascal, 20 Avenue Einstein
69621 Villeurbanne Cedex
elod.egyed-zsigmond@insa-lyon.fr

Résumé. Les chatbots permettent à un programme informatique d'interagir plus naturellement avec un interlocuteur. Ils demeurent toutefois limités, n'étant pas sensibles à l'état d'esprit ou aux émotions de l'utilisateur, ce qui leur permettrait pourtant d'apporter des réponses plus pertinentes. La détection d'émotion sur des discussions textuelles a déjà été explorée pour l'anglais (SemEval 2019 Task 3), mais en français aucun dataset satisfaisant n'est disponible. Nous proposons de traduire le dataset de dialogues EmotionLines, dont les répliques anglaises sont issues de la série Friends, en exploitant sa diffusion en VF. Notre méthode de génération de dataset par traduction est adaptable à tout dataset tiré de séries ou films étrangers disponibles en VF. En utilisant ce dataset traduit, nous proposons un classifier basé sur le modèle de langage BERT, permettant de détecter l'émotion de l'utilisateur à partir de texte. Il tient compte du contexte de la discussion pour affiner ses prédictions.

1 Introduction

Les assistants conversationnels (chat-, voice-bots) sont actuellement en plein développement, particulièrement dans le domaine du service après vente. Ils automatisent la relation client tout en permettant à l'utilisateur d'interagir naturellement. Ils restent cependant limités en termes d'empathie : ils ne sont souvent pas sensibles à l'émotion du client. Or il ne faut pas réagir de la même manière si l'interlocuteur est heureux/neutre ou au contraire en colère.

Nous nous intéressons à la première étape pour résoudre ce problème : détecter quelle est l'émotion de l'interlocuteur. Nous représentons les émotions sous forme de 4 classes : *joy*, *sadness*, *anger*, *other* (*other* : autres émotions comme surprise, dégoût, ...). Nous ajoutons aussi une classe *neutral* (pas d'émotion). La détection est alors une classification avec 5 classes :

$$C = \{joy, neutral, sadness, anger, other\} \tag{1}$$

Notre objectif est de créer un modèle pour détecter l'émotion dans du texte, adapté au français et aux bots. Cela veut dire qu'il doit traiter des discussions entre interlocuteurs (en opposition

aux modèles qui analysent des tweets indépendants) ; et en français. Nous n'avons pas trouvé un jeu de données qui respecte cette contrainte et avons donc créé notre propre dataset.

Dans la suite, nous commencerons par un état de l'art des jeux de données disponibles et des méthodes de détection d'émotion textuelle ; nous nous intéresserons ensuite à la traduction du dataset anglais ; et nous proposerons enfin un modèle de détection d'émotion.

2 État de l'art

Plusieurs approches sont possibles pour modéliser les émotions :

— *Représentation discrète*. Liste finie d'émotions, innées et identiques chez tous les individus. Par exemple, Ekman (1992) définit 6 émotions primitives : joie, tristesse, colère, dégoût, surprise et peur.

— *Représentation dimensionnelle*. L'émotion est le mélange de l'activation de plusieurs aires cérébrales, d'axes indépendants ; Russell (1980) considère par exemple deux axes, de valence (déplaisir-plaisir) et d'intensité (calme-agité).

Les challenges récents de détection d'émotion textuelle utilisent la représentation d'Ekman, simplifiée à 4 classes *joy*, *neutral*, *sadness*, *anger* (Shmueli et Ku, 2019). Dans un contexte de bots, ignorer que d'autres émotions existent est réducteur, donc nous avons ajouté *other* (regroupant dégoût, peur, surprise), ce qui donne 5 classes (cf Eq. 1).

2.1 Jeux de données annotés par émotion

Dans notre contexte, nous avons besoin d'un jeu de données (1) français, (2) de discussions, (3) portant sur des thèmes généraux et (4) annoté par émotion. À l'issue de nos recherches, nous n'avons trouvé que le dataset DEFT 2015 [1] en français. Malheureusement, il ne contient pas de discussions et porte sur la transition écologique (domaine trop spécifique).

Par contre, en ignorant la contrainte du français, beaucoup de datasets sont utilisables. Par exemple, EmoContext de Chatterjee et al. (2019) respecte toutes les contraintes et contient en plus des dialogues avec des agents humains (idéal pour notre application). EmpatheticDialogues (Rashkin et al., 2019) contient des discussions de Facebook. Enfin, plusieurs datasets ont été créés autour de la série Friends comme EmotionDetection de Chen et Choi (2016) ; EmotionX et EmotionLines (Shmueli et Ku, 2019) ; et le MELD de Poria et al. (2019) [2].

2.2 Modèle de détection d'émotion

Pour la suite, nous définissons : un dialogue U est composé d'une séquence de répliques (notées u_i) : $U = [u_0, u_1, ..., u_n]$. Notre tâche de détection d'émotion revient à classifier du texte parmi 5 classes. Chatterjee et al. (2019) divisent les approches de classification de texte en 3 domaines.

Analyse sémantique Il s'agit d'utiliser des informations sémantiques (issues de graphes sémantiques par exemple). Ces méthodes ne sont pas appropriées pour détecter l'émotion, car elle est souvent implicite, sous-entendue.

1. https://deft.limsi.fr/2015/corpus.fr.php?lang=fr.
2. EmotionX et MELD sont des améliorations d'EmotionLines : ils reposent sur les mêmes dialogues et répliques.

Analyse syntaxique Il s'agit d'utiliser les *mots/caractères* pour classifier le texte. Des features de bas niveau comme la ponctuation, les émoticônes, les abréviations ou des informations de POS-Tagging peuvent être utilisées (Goubin et al., 2019). On peut aussi analyser les mots ou groupes de mots récurrents (n-grams) et prendre en compte leur TF-IDF (Ciccone et al., 2018). Finalement, on peut utiliser un dictionnaire d'affect comme FEEL (Abdaoui et al., 2017). Un dictionnaire d'affect est une liste de mots annotés par émotion. Les prédictions sont généralement réalisées par des SVM, des arbres de décision, ou des approches ensemblistes.

Text-embedding Il s'agit d'étudier le *sens global* de la phrase et voir s'il véhicule des informations d'émotion. Il existe plusieurs types d'embeddings, mais à l'heure actuelle ce sont les modèles de langage, et surtout BERT (Devlin et al., 2019), qui fournissent les meilleures performances. BERT représente les mots sous forme de vecteurs de taille fixe, qui sont représentatifs du sens du mot et de son contexte d'utilisation. Pour classifier du texte, Devlin et al. (2019) proposent de faire du transfer-learning en utilisant un modèle BERT pré-entraîné. L'idée est de tokenizer et de calculer l'embedding du texte avec BERT (tokens en entrée $\boldsymbol{x} = [CLS] +$ tokens$(u) + [SEP]$, $+$ étant la concaténation). Ensuite, l'embedding de $[CLS]$ est utilisé dans un classifier (par exemple un réseau de neurones) pour faire la prédiction.

La version originale de BERT est adaptée à la langue anglaise. Pour le français, plusieurs modèles ont été pré-entraînés, dont CamemBERT (Martin et al., 2019).

Dans le cadre de discussions entre plusieurs personnes, il peut être pertinent de prendre en compte le contexte historique de l'échange (ce qu'il s'est passé avant). Zahiri et Choi (2018) proposent d'encoder chaque réplique sous forme de vecteur puis d'utiliser un réseau de neurones (type CNN ou RNN) pour exploiter la séquence/matrice des embeddings. Plus simplement, Yang et al. (2019) concatènent les répliques précédentes pour les fournir en entrée à BERT. Les tokens en entrée deviennent :

$$\boldsymbol{x} = [CLS] + \text{tokens}(u_t) + [SEP] + \text{tokens}(u_{t-1}) + [SEP] + ... + \text{tokens}(u_{t-n}) + [SEP] \quad (2)$$

3 Traduction du jeu de données

D'un côté, nous ne trouvons pas de dataset français qui respecte nos contraintes ; mais de l'autre côté plusieurs jeux de données anglais remplissent tous nos besoins, sauf la langue. De fait, nous proposons de traduire un de ces datasets anglais. Nous pourrions employer un traducteur humain ou utiliser un traducteur automatique (e.g. Google Translate ou DeepL), mais nous ne trouvons pas ces solutions satisfaisantes.

Nous avons noté que le dataset EmotionLines est basé sur la série TV Friends : les dialogues sont issus de la série en anglais et labellisés par des annotateurs humains. Or il existe une version française (*VF*) de Friends, adaptée par des traducteurs professionnels. Nous proposons de traduire le dataset EmotionLines en utilisant la VF de Friends.

3.1 Méthode de traduction VO/VF

L'idée de la traduction est simple : localiser le dialogue annoté dans la série (n° de saison, d'épisode ; positionnement dans l'épisode), pour trouver les répliques correspondantes en VF. Pour y parvenir, nous proposons un processus en 3 étapes, comme illustré sur la Fig. 1.

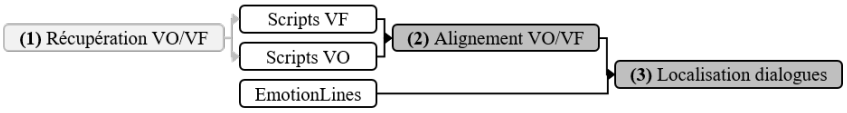

FIG. 1 – *Processus de traduction d'EmotionLines en utilisant la VO et la VF.*

Récupération VO/VF Plusieurs sources de données sont à notre disposition pour récupérer le texte en VO et VF, en particulier les sous-titres et les scripts [3]. Les sous-titres ont l'avantage d'être répandus et facilement accessibles. De plus, l'alignement entre la VO et la VF est facilité par les indications temporelles d'affichage des sous-titres. Par contre, il est très difficile de différencier et délimiter les répliques de personnages qui dialoguent ensemble, ce qui est pénalisant pour notre objectif de traduction de répliques. C'est pour cela que nous avons décidé de choisir les scripts comme source de données. Pour la VO, nous utilisons un dataset créé par Chen et Choi (2016). Pour la VF, nous avons scrappé le site fanfr [4].

Alignement VO/VF L'objectif est de construire une retranscription intégrale de la série, avec pour chaque réplique en VO sa correspondante en VF. Nous pouvons traiter chaque épisode indépendamment des autres car le découpage des épisodes est le même en VO et en VF. Au sein d'un épisode, la VO et la VF sont globalement similaires ; en théorie, ce sont les mêmes personnages qui parlent dans le même ordre. En pratique, la VF n'est pas une traduction littérale, mais une adaptation : certains dialogues sont remaniés (ordre différent des répliques), des répliques sont retirées, ou leur sens est changé. Nous devons donc mettre en place une méthode d'alignement approchée, qui tolère ces ajouts, suppressions ou inversions.

Pour y arriver, nous implémentons l'algorithme de Needleman et Wunsch (1970) utilisé normalement pour aligner des séquences ADN. Il a une approche globale qui consiste à tester tous les alignements et sélectionner celui qui est optimal. Le point central de l'algorithme est la fonction $match$ qui détermine si deux éléments sont compatibles : dans notre cas si la réplique française u^{fr} est une traduction de la réplique anglaise u^{en}. Elle retourne un score entre $[-1, 1]$ ($1 = correspondance$). Pour cette fonction, nous utilisons le texte de la réplique et une métadonnée, le personnage qui la prononce :

— *Personnage*. Si le personnage n'est pas le même dans les deux répliques, alors elles ne correspondent pas. En pratique, beaucoup de personnages n'ont pas le même nom en VO et VF. Nous avons construit la liste des personnages récurrents avec leur traduction VO/VF. Pour u^{fr} et u^{en}, si les personnages correspondent dans la liste (ou sont égaux), alors on renvoie 1 ; sinon -1. Pour les personnages hors de la liste, on retourne 0.

— *Texte de la réplique*. Il s'agit de comparer le texte en VO et VF. Des métriques simples peuvent être employées (comme la comparaison de la ponctuation), mais ces approches sont limitées. Nous préférons utiliser un modèle de sentence-embedding, l'Universal Sentence Encoder (Cer et al., 2018). Il encode des phrases de langues différentes dans un vecteur de taille fixe. Pour comparer les répliques, on calcule la similarité cosinus

3. Un script est structuré comme une pièce de théâtre : l'épisode est découpé en scènes, avec pour chaque scène les dialogues entre personnages et des indications scénaristiques.

4. https://www.fanfr.com/scripts/.

$\text{sim}_{cos}(\boldsymbol{u}, \boldsymbol{v}) = \langle \boldsymbol{u}, \boldsymbol{v} \rangle$ (produit scalaire car les embeddings sont normés). Deux répliques avec un sens proche ont une similarité proche de 1 et l'inverse si elles ont un sens différent.

Pour agréger ces 2 scores, nous utilisons une moyenne arithmétique. Nous avons testé plusieurs autres formulations de $match$ (moyenne géométrique, pondération des scores), mais nous n'avons pas noté d'amélioration d'alignement. Une étude plus approfondie sur ce sujet pourrait être pertinente. Au final, nous réussissons à trouver une correspondance pour 91.1% des répliques de la VO avec la VF.

Localisation des dialogues La dernière étape consiste à localiser chaque dialogue d'EmotionLines dans la VO (n° saison, épisode, position dans l'épisode), et d'en déduire sa VF. La comparaison s'effectue cette fois-ci entre des répliques en anglais. Pour cela, nous réutilisons la fonction $match$ précédente (à noter qu'il pourrait être pertinent d'utiliser à la place une distance d'édition, comme la distance de Levenshtein). Nous réussissons à localiser 95.9% des répliques d'EmotionLines dans la VO. En tenant compte de l'alignement précédent, nous «traduisons» 85.9% des répliques d'EmotionLines grâce à la VF.

3.2 Vérification de la qualité de la traduction et raffinage

85.9% est un bon score, mais notre méthode n'offre pas de garantie sur la qualité de l'association : il y a certainement des erreurs d'alignement (donc des mauvaises traductions). L'idéal serait de vérifier les alignements manuellement : comparer la réplique d'EmotionLines avec la correspondance en VF que nous avons déterminée : cela permettrait de calculer la proportion de bonnes traductions. Étant donné la taille du jeu de données, on peut estimer cette proportion à l'aide d'un intervalle de confiance calculé sur un échantillon (sélectionné de manière stratifiée suivant la taille des répliques, la longueur des dialogues et les classes d'émotion). Par manque de temps nous ne l'avons pas fait, mais ce serait une perspective intéressante à étudier.

Une manière automatique d'estimer la qualité de la traduction est d'utiliser l'Universal Sentence Encoder. En calculant la similarité cosinus entre les embeddings des répliques, nous pouvons avoir une idée de la correspondance entre la réplique anglaise d'EmotionLines et notre «traduction» en VF. Nous devrions observer une loi normale ayant une moyenne proche de 1.

Comparaison VF et traduction directe avec Google Translate La similarité moyenne entre les répliques d'EmotionLines anglais et celles «traduites» avec la VF est de 0.68. Pour référence, en traduisant EmotionLines avec Google Translate, la similarité moyenne entre l'anglais et la traduction française littérale est de 0.82. L'écart est de 0.14 en faveur de Google Translate, mais il faut garder à l'esprit que la VF n'est pas une traduction mot à mot : elle retranscrit l'esprit général de la réplique.

Impact de la longueur de la réplique sur la traduction Nous n'observons pas d'impact de la taille de la réplique sur la similarité calculée avec l'Universal Sentence Encoder.

Qualité de l'alignement et traduction Comme 85.9% des répliques ont été alignées, cela signifie que nous avons 14.1% de répliques sans correspondance : ce sont des *trous*. Ces trous ne sont pas uniformément répartis : certains dialogues sont intégralement alignés, alors que

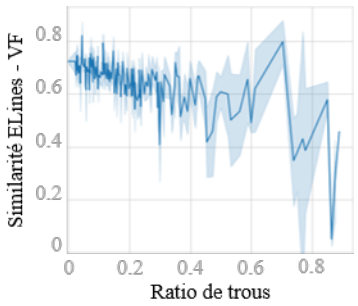

FIG. 2 – *Évolution de la similarité moyenne des répliques d'un dialogue en fonction du ratio de trous.*

Classe	**Proportion**	*EmotionLines*
neutral	44.5%	*45%*
conflict	19.3%	*19.1%*
other	15.6%	*15.5%*
joy	12.0%	*11.8%*
anger	5.2%	*5.2%*
sadness	3.4%	*3.4%*

TAB. 1 – *Répartition des classes dans le dataset final. Pour référence, les proportions initiales d'EmotionLines anglais sont affichées.*

d'autres ont beaucoup de trous. Sur la Fig. 2, nous voyons un lien clair entre un ratio de trous élevé et une similarité plus faible. En bref, le mauvais alignement d'un dialogue implique de mauvaises traductions. Pour rester dans un intervalle de traductions fiables, nous décidons de supprimer d'EmotionLines tous les dialogues où il y a plus de 20% de trous. Cela revient à garder 78% des dialogues et 67% des répliques d'EmotionLines.

3.3 Jeu de données final

Le dataset EmotionLines initial est composé de 14 997 répliques. Après traduction et raffinage, notre méthode traduit 9 965 répliques. Nous constatons que la distribution des classes de notre dataset est similaire à celle d'EmotionLines (voir Tab. 1). Également, le nombre moyen de répliques par dialogue de notre dataset est proche d'EmotionLines (resp. 13.9 et 14.5). Nous pouvons conclure que notre méthode de traduction ne modifie pas les caractéristiques essentielles d'EmotionLines.

Nous augmentons également nos données pour améliorer les performances du classifier. Lors du challenge EmotionX, Shmueli et Ku (2019) ont proposé 3 doubles-traductions pour augmenter la quantité de données. Une double-traduction consiste à utiliser un traducteur automatique pour traduire un texte vers une langue étrangère, et le retraduire en sens inverse (e.g. $fr \rightarrow en \rightarrow fr$) ; cela permet de faire varier les formulations et la syntaxe. De notre côté, nous utilisons 3 doubles-traductions avec Google Translate (anglais, allemand et espagnol en langues intermédiaires). Nous ajoutons aussi 3 doubles-traductions, qui partent de la traduction littérale d'EmotionLines par Google Translate, sans passer par notre méthode d'alignement VO/VF.

Le dataset final est composé de 99 848 répliques ($4 \times 9\,965 + 4 \times 14\,997$), et la répartition des classes figure sur le Tab. 1. Nous constatons le même déséquilibre de classes que pour EmotionLines. Cette distribution nous paraît toutefois représentative de la vie courante (majorité de répliques neutres).

Pour finir, EmotionLines a été annoté par 5 annotateurs humains : chaque annotateur vote, et c'est la classe avec le plus de votes qui est retenue. Dans presque 20% des cas, il n'y a pas de majorité : 2 classes ou plus sont ex-aequo. Dans ce cas là, la réplique est annotée *conflict*.

4 Détection d'émotion

4.1 Présentation du modèle

Pour la suite, nous définissons : $\boldsymbol{y}$ *et* $\hat{\boldsymbol{y}}$*, vecteurs qui suivent le format [joy, neutral, sadness, anger, other].* $\boldsymbol{y}$ *émotion(s) réelle(s) de la réplique :* $[1, 0, 0, 0, 0] = joy$*; et* $\hat{\boldsymbol{y}}$ *prédiction :* $[0.68, 0, 0.20, 0.12, 0]$. Nous utilisons un classifier qui s'appuie sur le modèle de langage CamemBERT (Martin et al., 2019), déjà pré-entraîné pour la langue française.

Prétraitement Nous normalisons le texte des répliques : passage en minuscules, simplification de la ponctuation (!!!!! →!!) [5]. Comme nous utilisons plusieurs traductions (augmentation), certaines traductions peuvent renvoyer le même texte ; dans ce cas nous retirons les doublons.

Contexte historique Les dialogues sont souvent composés de répliques courtes qui sont ambiguës en termes d'émotion (ex : *« Vraiment ? »*). Avoir accès à ce qu'il s'est passé avant permettrait d'affiner la prédiction. Pour tenir compte de l'historique, nous décidons de concaténer le texte des répliques précédentes du personnage. L'entrée du modèle devient celle de l'Eq. 2. À noter que pour l'entraînement, la taille de l'entrée (nombre de tokens) est fixe ; et également que le temps d'entraînement et la taille en RAM du modèle dépendent de ce nombre de tokens. Plus nous considérons d'historique, plus le modèle est volumineux et prend du temps à s'entraîner. Il faut donc trouver un compromis entre la taille du modèle, le temps d'entraînement et la qualité de la prédiction.

Architecture du classifier Il est composé de 2 parties. (1) Nous utilisons CamemBERT pour calculer l'embedding des mots de la réplique (+ historique). (2) L'embedding du token $[CLS]$ est fourni en entrée d'un petit classifier composé de deux couches denses/fully-connected, avec une fonction softmax en sortie pour avoir des probabilités sur les 5 classes.

Pour l'entraînement, nous utilisons les conseils de Devlin et al. (2019) : cross-entropy comme fonction d'erreur, optimiseur Adam, learning rate de 10^{-5}, taille de batch de 32, entraînement sur 4 époques, dropout à 50%. En outre, nous ajoutons de l'early-stopping pour interrompre l'apprentissage lorsque le score F1_{macro} cesse d'augmenter.

Gestion de la classe *conflict* La 2ème classe la plus représentée du dataset est *conflict* avec 19.3%. Ces répliques ne sont pas exploitées pour l'entraînement car plusieurs classes sont ex-aequo ; mais elles contiennent pourtant de l'information. Si on prend une réplique labellisée *conflict*, où 2 annotateurs ont voté pour *joy*, 2 pour *neutral* et 1 pour *other* alors :

— Si le modèle prédit *joy* ou *neutral*, on peut considérer que la prédiction est valide.
— À l'inverse si le modèle prédit une autre classe, on peut estimer la prédiction fausse.

5. D'une part, !!!!! et !!!! donnent une information très proche, et les différencier peut entraîner du surapprentissage. D'autre part, pour un voice-bot, la ponctuation est très pauvre à cause du speech-to-text.

Pour prendre en compte ces cas, notre idée est de modifier la fonction d'erreur cross-entropy. Pour rappel, elle est définie comme : $\text{CELoss}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{c \in C} \boldsymbol{y}_c \log(\hat{\boldsymbol{y}}_{\boldsymbol{c}}) = -\boldsymbol{y} \cdot \log(\hat{\boldsymbol{y}})$. Il est intéressant de remarquer que le vecteur $\boldsymbol{y}$ agit comme une clé : $-log(\hat{\boldsymbol{y}}_{\boldsymbol{c}})$ n'est pris en compte que si $\boldsymbol{y}_c \neq 0$ (l'émotion annotée). Notre idée est de créer un vecteur $\boldsymbol{y}^{\boldsymbol{conflict}}$ (qui remplace $\boldsymbol{y}$), respectant les constatations précédentes.

Soit $C_m = \{c \in C, \boldsymbol{y}_c = \max_{c \in C}(\boldsymbol{y})\}$, l'ensemble des classes ex-aequo pour une réplique. Il est constitué d'une classe (réplique annotée *joy*, *neutral*, ...) ou plusieurs (*conflict*).

— Si le modèle prédit une classe majoritaire ($c_m \in C_m$), alors $\boldsymbol{y}_{c_m}^{\boldsymbol{conflict}} = 1$ et $\forall c \neq c_m$, $\boldsymbol{y}_c^{\boldsymbol{conflict}} = 0$.

— Sinon (le modèle se trompe), $\forall c_m \in C_m, \boldsymbol{y}_{c_m}^{\boldsymbol{conflict}} > 0$ et $\forall c \notin C_m, \boldsymbol{y}_c^{\boldsymbol{conflict}} = 0$.

Nous normalisons $\boldsymbol{y}^{\boldsymbol{conflict}}$ pour que les répliques où plusieurs classes sont ex-aequo aient moins d'importance que les autres. Nous utilisons la norme l_1 ($[\![\cdot]\!]_1$, notation pour un vecteur normé avec l_1). Nous mettons le vecteur normalisé au carré. L'Eq. 3 définit $\boldsymbol{y}^{\boldsymbol{conflict}}$, l'Eq. 4 la cross-entropy modifiée.

$$\boldsymbol{y}^{\boldsymbol{conflict}} = [[\![\text{ismax}(2 \cdot \text{ismax}(\boldsymbol{y}) + \text{ismax}(\hat{\boldsymbol{y}}))]\!]_1]^2 \tag{3}$$

$$\text{avec } \forall c \in C, \text{ismax}(\boldsymbol{v})_c = \begin{cases} 1, & \boldsymbol{v}_c = \max\limits_{c \in C}(\boldsymbol{v}) \\ 0, & \text{sinon} \end{cases}$$

$$\text{CELoss}^{conflict}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{c \in C} \boldsymbol{y}_c^{\boldsymbol{conflict}} \log(\hat{\boldsymbol{y}}_{\boldsymbol{c}}) = -\boldsymbol{y}^{\boldsymbol{conflict}} \cdot \log(\hat{\boldsymbol{y}}) \tag{4}$$

Pour illustrer le format du vecteur $\boldsymbol{y}^{\boldsymbol{conflict}}$, plusieurs exemples sont présentés dans le Tab. 2. Le vecteur $\boldsymbol{y}^{\boldsymbol{conflict}}$ est identique à $\boldsymbol{y}$ s'il n'y a qu'une classe majoritaire, donnant la même formule qu'une cross-entropy classique.

Classes majoritaires ($\boldsymbol{y}$)	**Classe prédite ($\hat{\boldsymbol{y}}$)**	$\boldsymbol{y}^{\boldsymbol{conflict}}$
$[1,1,0,0,0]$ (*joy*, *neutral*)	$[1,0,0,0,0]$ (*joy*)	$[1,0,0,0,0]$
	$[0,0,0,0,1]$ (*other*)	$[\frac{1}{4},\frac{1}{4},0,0,0]$
$[1,0,0,0,0]$ (*joy*)	$[1,0,0,0,0]$ (*joy*)	$[1,0,0,0,0]$
	$[0,0,0,0,1]$ (*other*)	$[1,0,0,0,0]$

TAB. 2 – *Exemples de vecteurs $\boldsymbol{y}^{\boldsymbol{conflict}}$, en fonction de différents $\boldsymbol{y}$ et $\hat{\boldsymbol{y}}$.*
Les vecteurs suivent le format $[joy, neutral, sadness, anger, other]$.

4.2 Méthode d'évaluation du modèle

Évaluation du modèle Les auteurs d'EmotionLines proposent un découpage en jeu d'entraînement, de développement et de test. Nous avons conservé le jeu de test, et rassemblé les jeux d'entraînement et de développement ensemble. Pour diminuer la variabilité des métriques, nous mettons en place une validation croisée (avec 4 blocs). Pour gérer le déséquilibre des classes, nous pondérons l'erreur par classe.

Nous sélectionnons le score F1_{macro} comme métrique principale : il permet de voir si les performances sont équilibrées entre classes, et équilibrées en précision et sensibilité. Pour avoir une vision plus détaillée des performances par classe, il est aussi intéressant de calculer la matrice de confusion du modèle. Dans la suite, les métriques sont calculées sur le jeu de test.

Points de repère de performance Pour la baseline, en prédisant au hasard une des 5 classes (en respectant les proportions : 55% de *neutral*, ...), le score F1_{macro} vaut 20%.

Par ailleurs, le dataset EmotionLines nous permet d'avoir accès aux votes des annotateurs : nous savons combien d'annotateurs ont voté pour chaque classe. En utilisant ces données, nous calculons la performance moyenne des annotateurs. Nous trouvons que $\text{F1}_{macro}(humain)$ vaut 0.72. Concernant la matrice de confusion (Fig. 3 p. 10), nous voyons que les annotateurs ont tendance à confondre les émotions avec *neutral* (ligne horizontale), et la précision de certaines émotions est assez basse : 0.55 pour *sadness* et 0.62 pour *other*.

Enfin, le challenge EmotionX 2019 utilise le dataset EmotionLines original (en anglais). Il faut noter que les prédictions se font sur 4 classes sans *other*, et que le dataset d'évaluation est privé. Mais cela peut nous permettre de comparer nos résultats avec des modèles existants.

4.3 Performances du modèle

Taille de l'historique Nous avons testé différentes tailles d'historique (entre 1 et 10 répliques précédentes), et nous obtenons les scores du Tab. 3. Nous constatons que le simple fait d'utiliser un historique augmente le score F1_{macro} de 1 à 2.5%. Ensuite, nous voyons que le score F1_{macro} maximum n'est pas atteint pour l'historique de taille 10 ; il est au contraire atteint pour un petit historique de 2 répliques. On peut y voir une explication logique : pour comprendre l'émotion actuelle, il n'est pas nécessaire de savoir ce qu'il s'est passé il y a 10 minutes, les échanges récents suffisent.

Nous avons sélectionné pour la suite un historique de 2 répliques, d'autant plus que le nombre de répliques d'historique impacte la taille du modèle et son temps d'entraînement.

Historique	**F1_{macro}**
aucun	0.512
1	0.525
2	**0.537**
3	0.524
5	0.53
10	0.533

TAB. 3 – *Comparaison du score F1 en fonction du nombre de répliques d'historique.*

Entropy	0 ann	1 ann	2 ann	3 ann	4 ann	5 ann
conflict	**100**	**84.8**	**69**	**53.4**	**35.6**	**16.7**
ref	**100**	82.2	66.5	51.8	34.5	16.3

TAB. 4 – *Distribution cumulative du nombre d'annotateurs d'accord avec la prédiction. Comparaison entre la cross-entropy classique (ref) et CELoss^{conflict} (conflict).*
Lecture : « dans ...% des cas, au moins X annotateurs sont d'accord avec la prédiction ».

Répliques annotées *conflict* À l'heure actuelle, nous ignorons les 20% de répliques labellisées *conflict*. La fonction $\text{CELoss}^{conflict}$ que nous avons définie permet de les prendre en compte pendant l'entraînement. En pratique, nous n'observons pas d'écart significatif de performance entre un modèle entraîné avec une cross-entropy classique et un autre avec notre version modifiée (que ce soit en score F1_{macro} ou pour les matrices de confusion).

La différence notable entre ces deux modèles se situe sur l'accord entre le modèle et les annotateurs. EmotionLines nous donne accès aux votes des annotateurs. Ainsi, lors d'une pré-

diction nous pouvons regarder combien d'annotateurs sont d'accord avec la classe prédite. Si la prédiction correspond au label, alors au moins 3 annotateurs sont d'accord (3 annotateurs donnent la majorité). Mais le cas le plus intéressant se situe plutôt quand le modèle se trompe. En effet, nous distinguons 2 cas :

— Aucun annotateur n'est d'accord avec la prédiction. La prédiction du modèle est complètement absurde : il faut absolument éviter ces cas.

— 1 ou 2 annotateurs sont d'accord avec le modèle. Même si cela reste une erreur, des annotateurs se sont trompés, le texte est probablement ambigu (erreur moins grave).

Dans le Tab. 4, nous avons tracé la distribution cumulative du nombre d'annotateurs d'accord avec la prédiction. En pratique, il y a un écart de $1-3\%$ en faveur de CELoss^{conflict}, ce qui signifie que le modèle fait des prédictions plus proches de celles des annotateurs humains. En particulier, le modèle de référence n'est d'accord avec aucun annotateur dans 17.8% des cas (différence entre les colonnes *0 ann* et *1 ann* : $100 - 82.2 = 17.8$). Cette valeur descend à 15.2% avec CELoss^{conflict}. À noter que les écarts tendent à se raccourcir plus les annotateurs sont équivoques (l'émotion de ces répliques est moins ambiguë).

La fonction CELoss^{conflict} vient avec des contraintes : elle est plus lente à calculer que la cross-entropy classique, et elle n'est pas différentiable (à cause de $\boldsymbol{y^{conflict}}$), ce qui perturbe un peu l'entraînement (nous avons dû réduire le learning rate à 10^{-6}). Ces contraintes augmentent le temps d'entraînement de $30-40\%$. Pour nous, cette fonction est pertinente car elle améliore les résultats du modèle, mais ces contraintes ne sont pas négligeables.

Modèle final En utilisant le classifier avec un historique de 2 répliques et CELoss^{conflict}, nous avons un score F1$_{macro}$ de 0.54. Nous sommes au-dessus de la baseline, mais il reste encore une marge de progression pour atteindre un niveau de performance similaire à un humain.

Il est toutefois intéressant de remarquer que le modèle a un comportement proche d'un humain. En comparant les Fig. 3 et 4, on peut noter des similarités : la classe la mieux prédite est *neutral*, suivie de *joy*, *other*, *anger* et *sadness*. De la même manière, on voit que le modèle a tendance à sur-prédire les classes d'émotion, comme un humain (ligne horizontale au niveau de *neutral*). Plus secondairement, le modèle et l'humain ont la même faiblesse en confondant *anger* avec *other*.

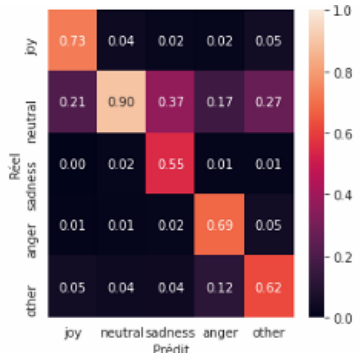

FIG. 3 – *Matrice de confusion moyenne des annotateurs humains.*

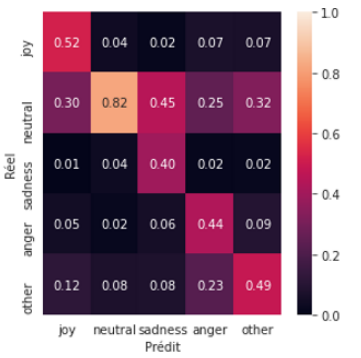

FIG. 4 – *Matrice de confusion du modèle de détection d'émotion.*

Pour finir, nous avons entraîné notre classifier uniquement sur 4 classes sans *other*, pour nous comparer avec les participants du challenge EmotionX (Shmueli et Ku, 2019). Le dataset d'évaluation d'EmotionX est privé, nous n'y avons pas eu accès ; nous avons utilisé le jeu de test précédent. Nous obtenons un score $\mathrm{F1}_{macro}$ de 67%, ce qui nous placerait entre la 4ème et 5ème place du challenge [6], en utilisant un dataset français.

5 Conclusion

Dans cet article, nous avons présenté une méthode pour traduire le dataset EmotionLines, s'appuyant sur les similarités entre VO et VF de la série Friends. Elle se base sur l'algorithme de Needleman-Wunsch pour aligner la VO avec la VF. En exploitant ce dataset français, nous avons proposé un modèle de détection d'émotion, exploitant l'historique de la discussion pour affiner ses prédictions. En outre, nous proposons une fonction d'erreur $\mathrm{CELoss}^{conflict}$ qui utilise les répliques où les annotateurs ne sont pas unanimes ; et nous constatons que les prédictions du modèle sont plus en accord avec l'annotation.

Dans de futurs travaux, il serait intéressant de vérifier plus formellement la validité de l'alignement/traduction. Il faudra tenir compte du fait que la VF n'est pas une traduction littérale, mais qu'elle prend quelquefois des libertés sur la VO. Une étude plus approfondie sur la formulation de la fonction $match$ pourrait aussi être pertinente.

Références

Abdaoui, A., J. Azé, S. Bringay, et P. Poncelet (2017). FEEL : A French Expanded Emotion Lexicon. *Language Resources and Evaluation 51*(3), 833–855.

Cer, D., Y. Yang, S. yi Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, et R. Kurzweil (2018). Universal Sentence Encoder.

Chatterjee, A., K. N. Narahari, M. Joshi, et P. Agrawal (2019). SemEval-2019 Task 3 : EmoContext Contextual Emotion Detection in Text. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, Minneapolis, Minnesota, USA, pp. 39–48. Association for Computational Linguistics.

Chen, Y.-H. et J. D. Choi (2016). Character Identification on Multiparty Conversation : Identifying Mentions of Characters in TV Shows. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Los Angeles, pp. 90–100. Association for Computational Linguistics.

Ciccone, G., A. Sultan, L. Laporte, E. Egyed-Zsigmond, A. Alhamzeh, et M. Granitzer (2018). Stacked Gender Prediction from Tweet Texts and Images Notebook for PAN at CLEF 2018. In *CLEF 2018 - Conference and Labs of the Evaluation*, Avignon, France.

Devlin, J., M. Chang, K. Lee, et K. Toutanova (2019). BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT 2019*, pp. 4171–4186.

6. Leaderboard disponible à l'url : https://sites.google.com/view/emotionx2019/leaderboards.

Ekman, P. (1992). An Argument for Basic Emotions. *Cognition and Emotion 6(3-4)*, 169–200.

Goubin, R., D. Lefeuvre, A. Alhamzeh, J. Mitrović, E. Egyed-Zsigmond, et L. Fossi (2019). Bots and Gender Profiling using a Multi-layer Architecture Notebook for PAN at CLEF 2019. In *CLEF 2019*.

Martin, L., B. Müller, P. J. O. Suárez, Y. Dupont, L. Romary, É. V. de la Clergerie, D. Seddah, et B. Sagot (2019). CamemBERT : a Tasty French Language Model. In *ACL 2020 - 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7203–7219.

Needleman, S. B. et C. D. Wunsch (1970). A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. *Journal of Molecular Biology 48*(3), 443–453.

Poria, S., D. Hazarika, N. Majumder, G. Naik, E. Cambria, et R. Mihalcea (2019). MELD : A Multimodal Multi-Party Dataset for Emotion Recognition in Conversations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, pp. 527–536. Association for Computational Linguistics.

Rashkin, H., E. M. Smith, M. Li, et Y.-L. Boureau (2019). Towards Empathetic Open-domain Conversation Models : A New Benchmark and Dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, pp. 5370–5381. Association for Computational Linguistics.

Russell, J. A. (1980). A Circumplex Model of Affect. *Journal of Personality and Social Psychology 39*(6), 1161–1178.

Shmueli, B. et L.-W. Ku (2019). SocialNLP EmotionX 2019 Challenge Overview : Predicting Emotions in Spoken Dialogues and Chats.

Yang, K., D. Lee, T. Whang, S. Lee, et H. Lim (2019). EmotionX-KU : BERT-Max based Contextual Emotion Classifier.

Zahiri, S. M. et J. D. Choi (2018). Emotion Detection on TV Show Transcripts with Sequence-based Convolutional Neural Networks. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 44–51.

Summary

Chatbots allow a computer program to interact more naturally with a human being. However, they remain limited in empathy, as they are not aware of the user's state of mind and emotions. Yet, it would help them to provide answers that are more appropriate. EmoContext (SemEval Task 3) has already explored textual emotion detection in English discussions, but no satisfactory dataset is available in French to this date. We propose to translate the EmotionLines dialog dataset, whose English utterances are taken from the Friends sitcom, using its French version. Our translation approach is generalizable to any dataset created from series or movies available in French. Using this translation, we propose a classifier based on the BERT language model, which aims to detect the user's emotion from text. This classifier takes into account the context of the discussion to refine its predictions.