

# Détection et recodage d'éléments paralinguistiques avec Python

*Philippa Payne*

*Sous la direction d'Aris Xanthos*

*Projet en Informatique – SA2023*

## 1. Introduction

La tache que j'ai choisi d'entreprendre pour les besoins du cours *Projet en informatique ou méthodes mathématiques* vise principalement à analyser et encoder des messages WhatsApp d'une manière qui préserve les aspects paralinguistiques du langage, afin de permettre l'analyse attendue de cette communication.

J'ai ainsi développé un module Python qui permet la manipulation de ces chats et peut être utilisé par des chercheurs pour analyser efficacement ces aspects tout en préservant autant qu'informations utiles que possible, sans compromettre l'anonymat souhaité de ceux et celles qui ont fourni leurs chats pour la recherche.

Dans ce rapport, je souligne les aspects de langage que mon programme préserve, je détaille les fonctionnalités de mon package, et je donne enfin une analyse des résultats de ce que j'ai entrepris dans le contexte de ce projet.

## 2. Paralangage

On peut définir le paralangage comme les « parties de discours qui sont exprimées dans la communication en ligne ».<sup>1</sup> Plus explicitement, le paralangage textuel peut être conceptualisé comme « les manifestations écrites des éléments non verbaux audibles, tactiles et visuels qui complètent ou remplacent le langage écrit ».<sup>2</sup> Dans l'ère numérique, la communication au moyen d'un ordinateur a pris un rôle très significatif dans les interactions entre les êtres humains et, par extension, dans la façon dont nous naviguons les relations sociales.

La communication au moyen d'un ordinateur, ou CMC (Computer-Mediated Communication), a introduit de nouvelles façons de communiquer, ainsi que de

---

<sup>1</sup> Luangrath, et al., 'Paralanguage Classifier (PARA)', *Journal of Marketing Research* 60.2 (2022), p. 388.

<sup>2</sup> Luangrath, et al., p. 399.

nouvelles contraintes.<sup>3</sup> La communication en ligne, à travers les réseaux sociaux et les applications de messagerie, s'est adaptée à ces nouvelles possibilités et contraintes. Identifier ces adaptations du langage est donc crucial pour permettre aux chercheurs en sciences sociales et linguistiques de mieux comprendre la communication entre les êtres humains à cette époque. Ce projet cherche ainsi à automatiser les premières étapes d'une analyse textuelle du paralangage.

Dans ce projet, je me suis concentrée sur les éléments non verbales de conversation qui tentent souvent de pallier les contraintes de la communication par ordinateur. Cela inclut la représentation d'aspects tels que l'intonation, le tempo, le rythme, l'accentuation, le ton, le volume, le langage corporel et l'expression du visage, ainsi que les interjections et la ponctuation.<sup>4</sup> Le paralangage textuel (TPL) peut être divisé en trois catégories : auditive, tactile, et visuelle.<sup>5</sup> Le paralangage tactile et visuel inclut souvent l'utilisation d'emojis (😊) et d'émoticons (:D), ainsi que de mots mimétiques (\*smiles\*), dans le but de représenter le langage corporel et les expressions du visage. En revanche, le paralangage auditif comprend des éléments textuels qui expriment le son de la parole, en utilisant principalement des répétitions de lettres, la manipulation des règles de majuscules et la ponctuation.

J'ai pris en considération les caractéristiques qui fournissent le plus d'informations utiles et fiables sans compromettre les informations personnelles et identifiables des messages. J'ai choisi d'encoder tous les aspects qui font partie du vocabulaire standard avec la lettre grecque λ, en utilisant la lettre majuscule Λ pour préserver les lettres majuscules des messages (ainsi, *Hello there* devient ΛΛΛΛΛ ΛΛΛΛΛ). Les chiffres sont encodés avec μ (par exemple, 2023 devient μμμμ), pour permettre l'identification des chiffres dans les analyses textuelles. Les liens sont encodés avec ω.

La question la plus importante était de décider quels aspects resteraient sans encodage. Il était clair que la ponctuation (et, par extension, les émoticons) et les emojis devaient rester non traités ; ces éléments du langage sont facilement identifiables en utilisant les expressions régulières. Pour conserver la plupart des

<sup>3</sup> Luangrath, et al., 'Textual Paralanguage and Its Implications for Marketing Communications', *Journal of Consumer Psychology* (2017), p. 3.

<sup>4</sup> Carey, 'Paralanguage in Computer Mediated Communication', Association for Computational Linguistics (1980), p. 68.

<sup>5</sup> Luangrath, et al., 'Paralanguage Classifier (PARA)', p. 399.

caractéristiques paralinguistiques, il était évident qu'il fallait laisser les interjections non encodées. On peut définir les interjections comme des mots qui « peuvent constituer des énoncés par eux-mêmes ».<sup>6</sup> En anglais, les mots « hey », « yeah » ou « oops » peuvent être considérés comme des exemples d'interjections.

Les éléments non verbaux de la communication, tels que l'intonation et le tempo, sont souvent exprimés avec des répétitions, soit une lettre qui est répétée une fois ou plus (par exemple, *whaat, riiight*), soit plusieurs lettres qui sont répétées (par exemple, *yeessss*). Afin de préserver l'information linguistique associée, j'ai choisi d'encoder le noyau du mot (par exemple, *yes*), tout en laissant les lettres répétées non encodées (par exemple, *yeessss* devient *λeλsss*), ce qui permet l'identification et l'analyse textuelle de ces répétitions.

Il est nécessaire d'abandonner certains aspects du paralangage dans le cadre de ce projet, tels que les mots mimétiques marqués avec des astérisques. La ponctuation est conservée, mais il arrive que les utilisateurs incluent plusieurs mots à l'intérieur de ces astérisques pour décrire une action plus complexe. Par conséquent, il existe un risque que ces descriptions contiennent des informations plus personnelles. Au moins, les astérisques sont maintenus pour indiquer que cette formulation a été utilisée, même si on ne peut pas identifier leur nature.

Un autre élément que j'ai tenté de capturer concerne les abréviations textuelles (comme *idk* ou *wdym*). Cependant, dans le but de protéger les informations personnelles, j'ai encodé tous les tokens qui ne peuvent pas être identifiés comme des interjections ou comme des mots faisant partie du vocabulaire standard. Même s'il existe certains cas où ces abréviations font partie des interjections, il y a également des cas où elles font partie du vocabulaire standard, ou ne sont pas identifiables en tant que caractéristiques du langage. Il y a ainsi une certaine perte d'information lorsque ces abréviations sont encodées.

### **3. Fonctionnalité**

J'ai créé un programme en Python qui sert à analyser et encoder des fichiers de texte fournis par l'utilisateur. Il existe deux façons de l'utiliser : soit on peut exécuter le script lui-même, soit on peut importer les fonctions du programme en tant que module et les

---

<sup>6</sup> Ameka, 'Interjections', *Journal of Pragmatics* 18 (1992), p. 101.

utiliser indépendamment. L'utilisateur peut fournir un fichier-texte qui est directement exporté de WhatsApp ; ce fichier est formaté d'une certaine manière avec la date, l'heure, l'auteur et finalement le message associé sur chaque ligne. J'ai fourni une fonction de formatage qui permet à ces messages formatés d'être transformés en une liste de dictionnaires, préservant ainsi toutes les informations associées aux messages. Les noms des auteurs sont anonymisés sous la forme de « speaker0 », « speaker1 », etc. Si le fichier-texte fourni par l'utilisateur ne respecte pas les règles de formatage de WhatsApp, il est possible de le traiter et de l'encoder comme un texte entier, en sautant cette étape de formatage.

Après le traitement des messages, plusieurs étapes d'analyse suivent afin d'identifier les éléments suivants :

<b>Aspect de langage</b>	<b>Input</b>	<b>Encodage</b>	<b>Output</b>
Les mots faisant partie du vocabulaire standard	<i>what</i> , <i>friend</i> , <i>social</i>	Encodés	λλλ, λλλλλ, λλλλλ
Les interjections	<i>hey</i> , <i>yeah</i> , <i>oops</i>	Non encodés	<i>hey</i> , <i>yeah</i> , <i>oops</i>
Les mots contenant une répétition	Faisant partie du vocabulaire standard	<i>whaat</i>	Partiellement encodés
	Représentant les interjections	<i>heey</i>	Non encodés
	Pas identifiable	<i>abcd</i>	Partiellement encodés
Les mots contenant plusieurs répétitions	Faisant partie du vocabulaire standard	<i>whaatt</i>	Partiellement encodés
	Représentant les interjections	<i>heeyy</i>	Non encodés
	Pas identifiable	<i>abcccd</i>	Partiellement encodés

Les nombres	1234	Encodés	$\mu\mu\mu\mu$
Les liens	<a href="https://github.com">https://github.com</a>	Partiellement encodés	$\omega\omega\omega\omega$ $\omega://\omega\omega$ $\omega\omega\omega\omega.$ $\omega\omega\omega$
La ponctuation	?!*^	Non encodés	?!*^
Les emojis	😍	Unicode	\U0001f60e
Les séquences non identifiables	abcd	Encodés	λλλλ

Pour identifier les interjections, j'ai utilisé deux bibliothèques de traitement du langage naturel (NLP) – SpaCy et le Natural Language Toolkit (NLTK). Ces deux bibliothèques permettent l'analyse textuelle en utilisant l'étiquetage grammatical. On peut ainsi identifier les interjections à travers ces étiquetages. SpaCy utilise l'étiquetage « INTJ » pour désigner les interjections, tandis que NLTK utilise les étiquetages « UH » (spécifique à la langue utilisée) et « X » (un étiquetage universel qui englobe toutes les langues).<sup>7</sup> L'étiquetage grammatical dépend du contexte dans lequel un token est utilisé dans une phrase ; ainsi, le token « hey » peut être considéré comme une interjection dans certaines phrases et pas dans d'autres.

La variabilité résultante de l'identification des interjections dans les messages m'a poussé à utiliser la combinaison de tous ces choix d'étiquetage pour identifier le maximum des interjections. De plus, au début du programme, tous les messages sont parcourus pour obtenir une liste provisoire des interjections, qui peut ainsi être utilisée pour identifier les interjections les plus courantes dans le texte lorsque l'on a besoin d'identifier des interjections dans des mots avec des répétitions (par exemple, si l'on a déjà identifié « hey » comme une interjection, une comparaison des noyaux possibles de « heeyy » avec la liste initiale des interjections fournit l'interjection souhaitée). Bien que cette méthode ne soit pas totalement fiable, et qu'il puisse y avoir une certaine perte d'interjections, j'ai également donné à l'utilisateur la possibilité d'ajouter un fichier externe avec une liste d'interjections (ou même de mots, plus

---

<sup>7</sup> ‘Categorising and Tagging Words’, NLTK (2023).

généralement, qu'il ne souhaite pas encoder). Si des correspondances sont trouvées dans le texte, ces mots sont ainsi laissés non encodés.

La partie qui s'est avérée la plus difficile était l'identification des mots à partir des tokens contenant des séquences de caractères répétés qui n'étaient pas immédiatement identifiables comme faisant partie du vocabulaire standard. Ces tokens sont souvent des mots, si l'on supprime les répétitions ; le token « yeess » est clairement une version allongée du mot « yes ». Pour surmonter ce défi, j'ai utilisé une combinaison d'expressions régulières et la propriété « span », qui donne le début et la fin de toutes les séquences de caractères identifiées à travers la correspondance regex.

En utilisant cette propriété « span », j'ai créé un algorithme de rétrogradation qui a trouvé toutes les combinaisons possibles des « spans » de ces caractères répétés. Cet algorithme de rétrogradation était inspiré par le problème des n-reines (combien de façons il y a d'organiser n-reines sur un échiquier de taille n, de sorte que ces reines ne puissent pas s'attaquer).<sup>8</sup> L'algorithme s'appelle lui-même récursivement chaque fois qu'il a trouvé une combinaison complète pour trouver toutes les autres combinaisons avec les mêmes séquences au début, jusqu'à ce que toutes les combinaisons soient trouvées. Une fois que toutes les combinaisons des « spans » sont trouvées, elles sont converties en une liste de mots possibles (yeess, yess, yees, yes).

Enfin, on recherche les interjections pour une correspondance dans cette liste. S'il y a une correspondance, elle est laissée non encodée. S'il n'y a pas de correspondance, on identifie si la liste contient des mots faisant partie du vocabulaire, à partir desquels on choisit le mot le plus long. On encode le noyau du mot tout en laissant les répétitions non encodées. De manière similaire, s'il n'y a pas de mots identifiables, on encode tous les caractères qui ne sont pas répétés et on laisse la séquence de caractères non encodée.

Une fois que l'analyse et l'encodage ont eu lieu, l'utilisateur a la possibilité de sauvegarder les messages encodés résultants dans un fichier texte.

#### **4. Réflexions**

---

<sup>8</sup> 'N Queen Problem', Geeks for Geeks (2023).

J'ai réussi dans le contexte de ce projet en créant un module Python qui permet l'analyse et l'encodage fiable des messages WhatsApp. Il y a une flexibilité dans l'encodage qui permet de traiter différents types d'entrées (de fichiers texte) et la mise à disposition de ce module permet à l'utilisateur d'adapter la fonctionnalité à ses besoins et de produire différents résultats.

Etant donné les contraintes de temps, il y avait plusieurs améliorations que j'envisageais pour l'avenir de ce projet. Certaines formulations de mots qui contiennent des répétitions ne sont pas incluses dans les modèles de langage de NLTK que j'ai choisi d'utiliser. Il existe d'autres modules Python qui peuvent permettre une représentation plus complète de la langue. Par exemple, l'infinitif du verbe « omit » est représenté, mais le participe passé « omitted » ne l'est pas ; le mot est ainsi considéré comme ne faisant pas partie du vocabulaire standard et est encodé en laissant les lettres répétées, plutôt qu'en encodant le mot entier. Il existe d'autres modules Python qui permet la manipulation des formes de verbe, comme Inflect, ce qui pourrait être envisagé pour l'avenir du projet.

J'ai choisi de me concentrer principalement sur les chats de WhatsApp, mais il est évident que le module peut être facilement adapté à d'autres applications de messagerie. Même si j'ai préservé les messages sans auteur produits par WhatsApp, ainsi que les messages qui déclarent les médias qui ne sont pas inclus (*<Media omitted>*), il est possible de créer un fichier externe qui contient les mots réservés par les applications (*<This message was edited>*). Intégrer ces phrases réservées nécessiterait une modification de mon code, qui fonctionne principalement avec des tokens plutôt qu'avec des séquences de tokens. De manière similaire, l'identification des séquences de tokens qui contiennent un sens paralinguistique, comme avec des actions mimétiques en astérisques, pourrait être une orientation utile pour une modification à l'avenir.

J'ai principalement travaillé avec des messages et des modèles de langage en anglais. Il reste à voir plus concrètement si ce processus fonctionne aussi bien en français, par exemple, qu'en anglais. Le code peut être adapté pour donner à l'utilisateur l'option de choisir la langue qu'il souhaite utiliser pour effectuer l'encodage.

Un autre aspect intéressant de ce projet concernerait la répétition de multiples noyaux en séquence. J'ai principalement examiné les séquences composées d'une seule

lettre, mais il existe des cas où plusieurs lettres sont répétées alternativement (par exemple, *ohohoh* contient trois fois la séquence *oh* ; dans ce cas, bien que le noyau du mot représente une interjection, mon code encoderait cet exemple comme une séquence de caractères non identifiable). La recherche de Gray et al., par exemple, s'est concentrée sur le concept d'un noyau (« kernel » en anglais) pour désigner la racine d'un mot qui peut être utilisé pour identifier des tokens significatifs dans une séquence de caractères répétés.<sup>9</sup> Cette recherche peut servir de point de départ utile.

## 5. Conclusion

En conclusion, ce projet avait pour objectif d'analyser et d'encoder les messages WhatsApp de manière à préserver les aspects paralinguistiques du langage, tout en facilitant leur analyse et en préservant l'anonymat des contributeurs. J'ai développé un programme et un module Python pour répondre à ce besoin.

Comme c'est souvent le cas avec les projets de programmation, il existe nombreuses possibilités de développement pour l'avenir. Cependant, j'ai réussi à créer un produit utilisable, tout en développant mes compétences en Python et ma compréhension de la communication au moyen d'un ordinateur, du paralangage, et de l'étiquetage grammatical. J'ai ainsi atteint mes objectifs en accomplissant un travail utile tout en réalisant un niveau de développement personnel. J'espère pouvoir approfondir le travail que j'ai accompli sur ce projet à l'avenir.

---

<sup>9</sup> Gray, et al., 'Hahahahaha, Duuuuude, Yeeessss!', *PLoS ONE* 15.5 (2020), p. 2.

## Références

- 'Categorising and Tagging Words', *NLTK* (2023), [https://www.nltk.org/book\\_1ed/ch05.html](https://www.nltk.org/book_1ed/ch05.html).
- 'N Queen Problem', *Geeks for Geeks* (2023), <https://www.geeksforgeeks.org/n-queen-problem-backtracking-3/>.
- Ameka, Felix, 'Interjections: The Universal Yet Neglected Part of Speech', *Journal of Pragmatics* 18 (1992), 101-118.
- Carey, John, 'Paralanguage in Computer Mediated Communication', *Association for Computational Linguistics* (1980), 67-69.
- Gray, Tyler J., et al., 'Hahahahaha, Duuuuude, Yeeessss!: A Two-Parameter Characterization Of Stretchable Words And The Dynamics Of Mistypings And Misspellings', *PLoS ONE* 15.5 (2020), 1-27.
- Luangrath, Andrea Webb, et al., 'Paralanguage Classifier (PARA): An Algorithm for Automatic Coding of Paralinguistic Nonverbal Parts of Speech in Text', *Journal of Marketing Research* 60.2 (2022), 388-408.
- Luangrath, Andrea Webb, et al., 'Textual Paralanguage and Its Implications for Marketing Communications', *Journal of Consumer Psychology* (2017), 1-37.
- Sanders, Jesper, '100+ Exclamations: The Ultimate Interjection List', *Pointerpro* (2023), <https://pointerpro.com/blog/the-ultimate-interjection-list/>.
- Walsh, Sorcha, 'Parathon: un outil pour l'annotation de la communication digitale', *Projet en Informatique* (2021), 1-10.
- Wharton, Tim, 'Interjections, Language and The Showing/Saying Continuum', *Pragmatics and Cognition* 11.1 (2003), 39-91.