POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Analyse et visualisation du processus d'écriture à l'aide des graphes

HÉLÈNE-SARAH BÉCOTTE-BOUTIN

Département de mathématiques et de génie industriel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*

Mathématiques

Juin 2019

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Cette thèse intitulée:

Analyse et visualisation du processus d'écriture à l'aide des graphes

Présentée par **Hélène-Sarah BÉCOTTE-BOUTIN**

en vue de l'obtention du diplôme de Philosophiæ Doctor

a été dûment acceptée par le jury d'examen constitué de :

Nadia LAHRICHI, Ph. D., présidente
Alain HERTZ, Doct. Ès Sc., membre et directeur de recherche
Gilles CAPOROSSI, Ph.D., membre et codirecteur de recherche
Christophe LEBLAY, Doctorat, membre et codirecteur de recherche
Diane RIOPEL, Doctorat, membre
Denis FOUCAMBERT, Ph.D., membre externe

REMERCIEMENTS

Je tiens d'abord à remercier mes directeurs de recherche, Alain Hertz, Gilles Caporossi et Christophe Leblay pour leur apport complémentaire. Vos suggestions et commentaires ont considérablement amélioré la qualité de cette thèse. Merci aussi pour votre patience et vos conseils dans les moments moins évidents de ce parcours qui n'aura pas été sans embûches.

Je tiens aussi à remercier le GERAD, autant pour la structure et les ressources à la disposition des étudiants que son personnel et les étudiants qui y gravitent. Je l'ai délaissé dans la dernière année pour des raisons logistiques, mais je suis reconnaissante des expériences professionnelles et personnelles que j'ai pu y vivre.

Merci à mes parents qui m'ont soutenue et encouragée de différentes façons. Les bons soupers et les dons de repas déjà préparés ont certainement contribué à rendre les dernières années plus faciles.

Finalement, merci à Jérôme, qui a été particulièrement compréhensif et présent pour moi dans la dernière année. Tu as fait toute la différence.

RÉSUMÉ

Lorsqu'un individu écrit un texte à l'ordinateur, il enfonce successivement des frappes sur son clavier dans le but de créer des mots et des phrases. Ces frappes peuvent être des caractères tels que des chiffres, des lettres ou des symboles. Ces frappes peuvent aussi être l'action de supprimer ces caractères. Après avoir altéré le texte de multiples fois, en effectuant ces frappes à plusieurs différents endroits, l'individu, qui est plus précisément appelé le scripteur, considérera que ce texte est terminé et arrêtera d'écrire à ce moment. L'ensemble du processus d'écriture est enregistré de telle manière qu'il est possible d'avoir accès à une liste exhaustive et détaillée de l'ensemble des opérations d'écriture effectuées par le scripteur lorsque celui-ci rédigeait le texte.

Ces fichiers sont par contre plutôt denses, sont constitués de données brutes et sans prétraitement, il est difficile pour un humain de les analyser et d'en tirer des conclusions sur le processus d'écriture. Le processus d'écriture est étudié par des chercheurs dans des domaines tels que la didactique, la linguistique et la psychologie cognitive. Bien que leurs objectifs de recherche soient différents, ils ont comme point commun de chercher à trouver des régularités dans les structures du processus d'écriture.

Plusieurs particularités rendent l'étude du processus d'écriture complexe, dont le fait que le texte est difficile à observer puisque son état change constamment. Actuellement, peu d'automatismes existent pour faciliter l'analyse du processus d'écriture. Parmi les méthodes qui ont été employées pour agréger les données du processus d'écriture, celle qui est privilégiée est la transformation de celles-ci en visualisations.

Dans le cadre de ce projet de recherche, la théorie des graphes est utilisée pour structurer l'information et les relations entre les frappes dans le but de développer des outils permettant de faciliter la recherche de tendances dans le processus d'écriture. Puisque les graphes peuvent également être visualisés, les modèles créés sont d'abord exploités en tant que visualisations.

Deux visualisations créées à partir des graphes sont présentées dans cette thèse : la *visualisation progressive* et le modèle *sans pertes*. La *visualisation progressive* vise à condenser les meilleurs attributs des visualisations existantes tout en incluant un maximum de dimensions propres à l'écriture.

La seconde visualisation, le modèle *sans pertes*, a été créée dans le but de structurer l'information de façon telle qu'il est possible d'utiliser des propriétés des graphes pour déceler certaines tendances dans les données. Ce modèle peut être utilisé en tant que visualisation, mais ses caractéristiques de présentation ne sont pas propres à celui-ci, ce qui le rend flexible.

Finalement, les mesures de proximité chronologique seront présentées. Celles-ci sont calculées à partir de la structure du graphe du modèle *sans pertes*. Elles permettent de démontrer qu'il est possible d'extraire de l'information pertinente avec le modèle *sans pertes*. Cette information est autrement difficile à obtenir.

Les modèles élaborés dans le cadre de cette thèse visent à permettre une analyse beaucoup plus facile du processus d'écriture.

ABSTRACT

When someone writes a text using a computer, he successively presses some keys on the keyboard in order to create words and sentences. These keystrokes can either be letters, numbers, symbols or the act of deleting them. After altering the text many times by inserting or deleting parts of it in just as many different places, this person, who is more precisely called the writer, will consider that this text is finished and will then stop writing. The entire process is recorded in a file containing an exhaustive and detailed list of all the writing operations performed by this writer.

These files are dense and consist of raw data. Without any preprocessing, it is difficult for a human to analyze them and to draw conclusions about the writing process. Researchers from many fields study the writing process. Although the research objectives in didactics, linguistics or cognitive psychology are different, they share the fact that they seek to find regularities in the structures of the writing process.

The process of writing a text is complicated to observe as the state of a text changes constantly. There are not a lot of methods that can aggregate the writing data and facilitate the analysis of the writing process. Among those available, the most used are visualizations of the writing process which consists of transformed data.

In this thesis, graph theory is used both to structure the data and the relations between the keystrokes in order to develop tools to help pattern analysis in the writing process. Since graphs can also be visually displayed, the models created are also exploited as visualizations.

Two visualizations created out of graph structures are presented in this thesis : the *progressive visualization* and the *lossless* model. The *progressive visualization* aims to condense the best attributes of existing visualizations while including a maximum of dimensions specific to writing.

The second visualization, the *lossless* model, was first conceived as a way to structure the information in order to be able to use graph theory properties to detect some patterns in the data. This model can be used as a visualization, but its presentation features are not unique to it, making it flexible.

Finally, the chronological proximity measures will be presented. They are calculated from the graph structure of the *lossless* model. They demonstrate that it is possible to extract easily relevant information from the lossless model. This specific information is otherwise difficult to obtain.

The models developed in this thesis were created to allow a much easier analysis of the writing process.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

GLOSSAIRE

Les concepts utilisés dans cette thèse sont listés par ordre alphabétique dans ce glossaire. La définition présentée correspond à celle qui est utilisée. Certains mots ou expressions dérivées et portant le même sens sont également présentes dans la thèse. Par exemple, les termes *chronologique* et *chronologiquement* réfèrent au concept *chronologie*.

Avant-texte : terme utilisé pour décrire les différentes versions du texte menant au texte final dans le contexte de l'écriture enregistrée. Il s'agit de l'équivalence pour le terme brouillon lorsqu'on fait référence aux différentes versions d'un texte menant à un manuscrit.

Caractère : valeur du clavier lorsque le type de frappe est une insertion.

Chronologie : il s'agit d'une simplification de la variable temporalité. La chronologie est l'ordre des opérations dans le fichier *log*. Deux frappes peuvent être chronologiques sans être consécutives.

Consécutivité : terme utilisé dans cette thèse pour désigner deux ou plusieurs frappes ayant une relation chronologique telle qu'elles ont été exécutées immédiatement l'une après l'autre. Le terme peut aussi être utilisé de manière plus large pour désigner deux événements survenant immédiatement l'un après l'autre.

Corpus d'écriture : fait référence à une collection de plusieurs fichiers *log* récoltés lors d'une même recherche

Contextuel : se dit d'une révision lorsque celle-ci se produit dans le déjà écrit.

Déplacement : opération d'écriture qui s'effectue en deux temps. D'abord une portion de texte est supprimée, puis exactement la même portion de texte est insérée à un autre endroit.

Durée de l'écriture : temps total utilisé par un scripteur pour réaliser une tâche d'écriture.

Écriture enregistrée : parfois aussi appelée écriture en-ligne. Fait référence à la rédaction d'un texte sur support informatique et à l'enregistrement du processus d'écriture dans un fichier *log*.

Fil de la plume : expression employée pour désigner la fin du texte en cours d'écriture.

Film d'écriture : reconstitution du processus d'écriture permettant de visualiser, à la manière d'un film, toutes les opérations effectuées par un scripteur, directement à partir des frappes enregistrées dans le fichier *log*.

Frappe : acte de solliciter une touche du clavier, qu'il s'agisse d'un caractère ou de la frappe suppression. Les frappes sont de type insertion ou suppression.

Genèse : terme utilisé pour parler des différentes étapes du processus d'écriture en ordonnant celles-ci dans le temps. Lorsqu'il est question des traces écrites et enregistrées des différentes versions d'un texte, les termes brouillon et avant-texte sont utilisés.

Insertion : une frappe de type insertion est l'ajout d'un caractère dans le texte.

Interruption : expression principalement utilisée dans la Notation-S. Une interruption survient lorsque le scripteur commence une révision immédiate ou différée.

Opération d'écriture : l'insertion (ajout), la suppression, le déplacement et le remplacement sont considérées comme des opérations d'écriture.

Pause : aussi appelé temps d'attente dans certains ouvrages. Il s'agit de la différence de temps entre deux frappes consécutives dans le fichier *log*.

Phase linéaire d'écriture : terme utilisé dans cette thèse pour décrire un ensemble de frappes consécutives exécutées sans que le curseur soit repositionné.

Pré-contextuel : se dit d'une révision lorsque celle-ci se produit au fil de la plume. Dans cette thèse, le terme est utilisé pour caractériser toute frappe se produisant au fil de la plume.

Réécriture : l'acte de modifier le déjà écrit. Lorsqu'il est question d'écriture enregistrée, le mot réécriture est synonyme du mot révision.

Remplacement : opération d'écriture qui s'effectue en deux temps. D'abord une portion de texte est supprimée, puis au même endroit, elle est remplacée par une autre portion de texte.

Repositionner : lorsque le curseur est repositionné, cela signifie que le scripteur déplace lui-même le curseur d'endroit, pour le positionner ailleurs dans le texte.

Révision : la révision signifie apporter un changement dans le processus d'écriture. Elle consiste soit à supprimer du texte ou à insérer du texte entre deux portions de texte déjà écrites.

Révision différée : il s'agit soit d'une suppression ou d'une insertion, lorsque celle-ci est effectuée suite à un repositionnement du curseur.

Révision immédiate : il s'agit de la suppression du déjà écrit, lorsque cette suppression est effectuée sans repositionner le curseur.

Scripteur : individu qui écrit un texte.

Scriptural : relatif à l'écriture.

Session d'écriture : si le texte a été écrit en une seule fois, il n'y a qu'une session d'écriture. Si le texte a été écrit à plusieurs moments différents comportant tous leur temps de début et de fin, il y a plusieurs sessions d'écriture.

Spatialité : concerne l'emplacement où les opérations ont été faites dans le texte.

Spatialité absolue : position de l'opération par rapport à la position de toutes les opérations qui ont été faites dans le texte.

Spatialité relative : position d'une opération telle qu'elle est enregistrée dans le fichier *log*; celle-ci concerne l'endroit où elle a été réalisée en considérant l'état du texte à ce moment précis.

Suppression : une frappe est une suppression si elle vise à effacer un caractère dans le texte en court d'écriture.

Tâche d'écriture : instructions particulières qu'un scripteur se doit de respecter lorsqu'il rédige un texte visant à être enregistré.

Temporalité : variable temporelle d'une frappe.

Temps de l'écriture : terme désigné pour décrire le processus de production d'un texte.

Tendance : fait référence à des similarités ou régularités dans les données.

Sources : Becotte-Boutin, Caporossi, Leblay, & Hertz (2019), Doquet (2003; 2014), Foucambert et Foucambert (2014), Leblay (2011; 2016), Manyika (2011), Sullivan et Lindgren (2014) et Van Waes et al. (2014).

CHAPITRE 1 INTRODUCTION

Pour plusieurs, écrire un texte de qualité exige un certain don particulier (Lebrave, 2001). Il s'agit plutôt d'une compétence qui se développerait avec l'âge et l'expérience (Midgette, Haria, & MacArthur, 2008). Bien qu'il existe aujourd'hui des logiciels avancés de traitement de texte pour aider à la rédaction, ceux-ci ont été conçus pour des usages bureautiques et non pas dans un objectif littéraire (Lebrave, 2001). Leur utilisation facilite le traitement de texte dans un milieu de travail, mais ces logiciels ne font pas de leurs utilisateurs de grands auteurs. Pour tenter de comprendre les mécanismes de création et les stratégies des grands auteurs, des psychologues, linguistes et didacticiens se sont penchés sur les manuscrits de grands auteurs tels que Flaubert, Proust, Valéry et Zola (Doquet C. , 2014). Cette discipline particulière, qui se nomme *critique génétique*, vise à comprendre le processus de la création littéraire. Pour ce faire, les chercheurs utilisaient initialement des brouillons rédigés à la main par ces auteurs dans le but de reconstruire le processus de rédaction de ces œuvres (Alamargot & Lebrave, The study of professional writing: A joint contribution from cognitive psychology and genetic criticism, 2009).

On sait relativement peu de choses de l'histoire des pratiques de production textuelle et le processus de rédaction est loin d'être connu dans le détail (Lebrave, 2001). Dès les années 90, grâce au développement des technologies informatiques, les généticiens ont pu reconstituer le film d'écriture. Cette avancée technologique a permis aux chercheurs de se concentrer sur l'analyse et non plus seulement sur la reconstitution du processus d'écriture (Doquet C. , 2014). Il est dorénavant possible, avec l'enregistrement du *temps de l'écriture*, d'accéder à une information exhaustive décrivant ce processus (Sullivan & Lindgren, 2014). Des bases de données sont constituées de fichiers *log*, ces documents où sont enregistrés les différentes opérations effectuées et le moment auquel elles l'ont été. Une collection de plusieurs fichiers *log* récoltés lors d'une même recherche est appelée *corpus d'écriture enregistrée* (Leblay C. , 2011). Les données récoltées sont nombreuses, et peu appropriées à une analyse humaine lorsqu'elles ne sont pas préalablement traitées (Caporossi & Leblay, 2014). Si le terme brouillon fait référence aux différentes versions d'un texte menant à un manuscrit, dans l'écriture enregistrée, le terme *avant-texte* est utilisé pour décrire les différentes versions du texte menant au texte final. Celles-ci peuvent être reconstruites à partir du fichier *log* (Leblay & Caporossi, 2014).

Actuellement, peu d'automatismes existent pour faciliter l'analyse du processus d'écriture. Deux aspects, qui sont d'une part les multiples dimensions dans lesquelles se situent le processus et d'autre part l'unicité de chaque rédaction par rapport aux autres, font de ce processus un cas particulier (Plane, Alamargot, & Lebrave, 2010). Le rôle de l'automatisation en linguistique a déjà été souligné. Déjà en 1997, des techniques d'exploration de données étaient utilisées dans le but « d'extraire automatiquement de l'information implicite ou inconnue à travers des données, au moyen de techniques d'apprentissage automatique (*machine learning*) et de reconnaissance statistique de tendances » (Daelemans, Berck, & Gillis, 1997). Ces techniques étaient cependant utilisées sur des textes finis et des données statiques et non pas sur le processus d'écriture qui est pour sa part composé de données dynamiques multidimensionnelles.

Alors que l'un des buts de l'étude du *temps de l'écriture* est de comprendre les mécanismes de la production d'un texte pour préciser les stratégies expertes de rédaction (Plane, Alamargot, & Lebrave, 2010), il reste difficile de comparer les *genèses* entre elles puisqu'aucune ne ressemble complètement à une autre (ITEM, 2014).

L'extraction automatique de l'information peut souvent être réalisée en modélisant les données. En effet, la modélisation et l'analyse de données permettent de déceler des tendances et motifs complexes qui auraient été auparavant difficiles à trouver (Boyd & Crawford, 2012). L'expression *motif complexe* est utilisée dans la littérature traitant de l'analyse et de la visualisation de données (Ware, 2004; Aigner, Miksch, Schumann, & Tominski, 2011; Berthold, 2011) y compris pour la visualisation des données d'écriture (Lindgren & Sullivan, 2002). Un motif est considéré simple si sa structure n'inclut aucun autre motif, autrement il s'agit d'un motif complexe (Bartolini, Ciaccia, Ntoutsi, Patella, & Theodoridis, 2004). Dans le cadre de cette thèse, un motif complexe est une information ou un ensemble d'information qui serait difficile à trouver par un chercheur en consultant directement le fichier *log*.

L'originalité de la contribution attendue réside dans la modélisation des données issues de l'enregistrement du processus d'écriture grâce à la théorie des graphes. Cette contribution permettra de fournir aux chercheurs des outils d'analyse leur permettant de trouver plus facilement des motifs complexes dans les données.

Bien qu'un glossaire soit disponible, la présentation de quelques concepts de base est de mise et sera présentée dans les prochaines sections.

1.1 Fichier *log*

Tel que mentionné précédemment, le processus d'écriture peut être enregistré dans un fichier, appelé *log,* sous la forme d'une liste d'insertions et de suppressions de caractères. Puisque ce fichier présente un type de données précises sur lequel l'ensemble de cette recherche se fonde, il est impératif de présenter sa structure et le résultat final qu'on peut en tirer.

Le tableau 1.1 présente un exemple du type d'information disponible dans ce fichier. Pour simplifier l'identification des lignes du fichier *log*, les lignes ont été numérotées en ordre croissant, ce qui correspond à la valeur *i*.

Tableau 1.1 Fichier *log* exemple 1.1

i	*Temps*	*Type*	*Pos*	*C*
1	1221	I	1	s
2	1812	I	1	u
3	3123	I	3	a
4	3959	S	1	-

Soit $AvantTexte(i)$, $Temps(i)$, $Type(i)$, $Pos(i)$ et $C(i)$ les attributs d'une frappe décrite par la i-ème ligne du fichier *log* :

- $AvantTexte(i)$ représente l'état du texte lorsque la i-ème ligne du fichier *log* a été écrite. Il ne s'agit pas d'un attribut enregistré dans le fichier puisque celui-ci peut être généré à n'importe quel moment.
- $Temps(i)$ indique le temps auquel la frappe a été effectuée.
- $Type(i)$ indique le type de frappe. S'il s'agit d'une frappe représentant l'ajout d'un caractère, le fichier indique *I*, et s'il s'agit de la frappe de suppression, le fichier indique *S*.
- $Pos(i)$ est la position dans $AvantTexte(i)$ de la frappe qui vient d'être effectuée. La plus petite position dans $AvantTexte(i)$ est 1.
    - Si $Type(i) = I$, un caractère est ajouté à la position $Pos(i)$ du texte. Ceci signifie que toutes les insertions qui sont en position $p \geq Pos(i)$ sont en position $p + 1$ après l'insertion *i*.

- Si $Type(i) = S$, le caractère qui est à la position $Pos(i)$ du texte est supprimé de la dernière version du texte. Ceci signifie que toutes les insertions qui sont en position $p > Pos(i)$ sont en position $p - 1$ après la suppression i.
- La valeur de $Pos(i)$ peut seulement être interprétée en fonction de $AvantTexte(i)$.

- $C(i)$ indique le caractère qui a été inséré dans le texte.

En traitant directement le fichier *log*, $AvantTexte(i)$ est obtenu en suivant l'algorithme suivant.

Algorithme 1 : Création de $AvantTexte$

Entrées : un fichier *log*, les n premières lignes du fichier *log*

Sorties : $AvantTexte(n)$

1 : Poser $\boldsymbol{AvantTexte} = \emptyset$
2 : **Pour toutes** les lignes *i* du fichier *log* telles que $i \leq n$ **faire**
3 : **Si** $Type(i) = I$ **alors**
4 : Ajouter $C(i)$ en position $Pos(i)$ dans $AvantTexte$
5 : **Pour toutes** les insertions *k* dans $AvantTexte$ **faire**
6 : **Si** la position de *k* est plus grande que la position de *i* **alors**
7 : Poser $Pos(k) = Pos(k) + 1$
8 : **Si** $Type(i) = S$ **alors**
9 : Trouver l'insertion *j* telle que $Pos(j) = Pos(i)$ dans $AvantTexte$
10 : Supprimer $C(j)$ de $AvantTexte$
11 : **Pour toutes** les insertions k dans $AvantTexte$ **faire**
12 : Si $Pos(k) > Pos(i)$ alors
13 : Poser $Pos(k) = Pos(k) - 1$
14 : Retourner $AvantTexte$

En reprenant cet algorithme, construisons le texte pour le fichier *log* 1.1.

Tableau 1.2 $AvantTexte(1)$

Texte	s
Pos	1

Tableau 1.3 $AvantTexte(2)$

Texte	u	s
Pos	1	2

Tableau 1.4 $AvantTexte(3)$

Texte	u	s	a
Pos	1	2	3

Tableau 1.5 $AvantTexte(4)$

Texte	s	a
Pos	1	2

La dernière et finale version du texte obtenu de ce fichier *log* est : *sa*.

1.2 Objectif de la thèse

L'objectif général de la thèse est de modéliser le fichier *log* pour en extraire de l'information. Cette information prend souvent la forme de motifs complexes.

Pour répondre à cet objectif, des nouveaux modèles du processus d'écriture seront présentés :

- La *visualisation progressive* a pour but de rassembler sur une seule visualisation l'ensemble des variables étudiées du processus d'écriture;
- Le modèle *sans pertes* a pour but de structurer l'information du fichier *log* de manière à la rendre plus accessible;
- Les mesures de proximité chronologique ont pour but d'exploiter la structure du modèle *sans pertes* et de donner accès à une information autrement difficile à trouver dans les données d'écriture.

Les modèles proposés ne consistent pas en une analyse du processus d'écriture. Il s'agit plutôt d'intermédiaires qui permettent une analyse beaucoup plus facile.

1.3 Données utilisées

Dans la littérature, les données utilisées par les chercheurs s'intéressant à l'écriture enregistrée proviennent de *corpus*, ce qui signifie l'ensemble des différents textes et processus (Mellet, 2003), qu'ils ont recueillis à partir d'études dans le cadre d'un contexte de recherche spécifique. Dès lors, une ou plusieurs tâches d'écriture sont données aux participants qui les réalisent en ayant différentes contraintes comme la *durée de l'écriture*, ou des phrases spécifiques à insérer par exemple. La création de ces *corpus* spécifiques vise à échantillonner des variables de recherche particulières.

Les données issues de l'écriture qui sont utilisées dans le cadre de cette thèse n'ont pas été recueillies à partir d'une étude. Puisque le contenu et la signification des textes ne sont pas abordés dans cette thèse, l'ensemble des processus illustrés ont été fabriqués à la suite de tests pour s'assurer d'avoir un ensemble d'exemples d'opérations d'écriture variées.

Les exemples ont été créés manuellement en utilisant le logiciel GenoGraphiX (Caporossi, 2015). Ce logiciel a été utilisé comme interface de traitement de texte et pour la génération du fichier *log*. Le but était que les fichiers *log* créés soient assez courts pour qu'ils soient interprétés manuellement par le lecteur afin que celui-ci saisisse plus facilement les concepts présentés, tout en illustrant un maximum d'opérations complexes. Dans la majorité des cas ils contiennent des mots et phrases complètes et ayant un sens pour faciliter la compréhension des explications.

1.4 Structure de la thèse

Cette thèse comprendra au Chapitre 2 une revue de littérature abordant plus en détail la nature du *temps de l'écriture* et les techniques d'analyse de données incluant une section plus détaillée sur la visualisation. Le Chapitre 3 présentera plus en détails certains outils méthodologiques importants dans le cadre de cette thèse. Le Chapitre 4 présentera une première visualisation du processus d'écriture, la *visualisation progressive*. Le Chapitre 5 présentera une seconde visualisation du processus d'écriture, le modèle *sans pertes*, qui génère un graphe possédant des propriétés structurelles pertinentes pour l'analyse du processus d'écriture. Le Chapitre 6 présente les mesures de proximité chronologiques, qui sont issues du graphe sans pertes et permettent d'avoir accès à une information autrement difficile à obtenir. Finalement, le Chapitre 7 constituera la conclusion.

CHAPITRE 2 REVUE DE LITTÉRATURE

La revue de littérature traitera à la fois des techniques utilisées dans l'analyse du *temps de l'écriture*, en linguistique et en analyse automatique du texte, et des techniques d'analyse de données en sciences et la théorie des graphes. Ce chapitre sera divisé en trois sections.

La première partie de la revue de littérature abordera la définition du *temps de l'écriture* et de ses problématiques. La deuxième section de la théorie mathématique et analytique utilisée dans cette thèse à savoir la théorie des graphes, l'analyse de données et la visualisation de données. La troisième section présentera les méthodes utilisées en analyse de l'écriture pour traiter les fichiers *log*.

Les Sections 2.2.2 et 2.3.1 sont traduites et adaptées du chapitre de livre *Writing and rewriting : The colored numerical visualization of keystroke logging* par Bécotte-Boutin et al. (2019).

2.1 Étude du *temps de l'écriture*

Le *temps de l'écriture* est le terme désigné pour décrire le processus de production d'un texte (Leblay & Caporossi, 2014). L'action de rédiger comprend les opérations d'*insertion* et de *suppression* de caractères, qui peuvent être *remplacés* par d'autres ou encore être *déplacés* à travers le texte. Certains auteurs plus minimalistes simplifient l'ensemble de ces opérations par l'opération de *remplacement* (Van Waes & Schellens, 2003; Lebrave & Grésillon, 2009). L'*insertion* devient le *remplacement* d'un espace vide par un caractère et la *suppression* est l'inverse. Dans la littérature, le terme *ajout* est généralement préféré par rapport au mot *insertion* qui lui fait référence spécifiquement au fait de modifier le déjà écrit en y insérant un caractère. Le terme *insertion* a été choisi dans cette thèse pour garder un terme propre aux données d'écriture et pour éviter toute confusion par rapport à son utilisation. Dans les différents algorithmes et descriptions présentées dans les prochains chapitres, le terme *ajout*, qui est un mot plus commun, peut signifier l'ajout de caractéristiques, de structures ou de valeurs.

Les *insertions* et *suppressions* sont caractérisées par le fait qu'elles s'opèrent en un seul temps grâce au clavier ou à la souris, tandis que le *remplacement* et le *déplacement* se font en deux temps (Caporossi & Leblay, 2011). Le *remplacement*, qui est l'action de remplacer un ensemble de frappes par un autre, peut également être considérée comme une *suppression* suivie par une

insertion, au même endroit dans le texte (Caporossi & Leblay, 2011). Pour sa part, le *déplacement* peut être considéré comme étant d'abord une *suppression*, qui est suivie d'une *insertion* identique à ce qui a été supprimé, mais à un autre endroit du texte dans ce cas-ci (Leblay C. , 2011). Cette dernière caractérisation est celle qui est utilisée dans le cadre de cette thèse.

Le processus d'écriture est particulier. Même si l'écriture d'un texte s'effectue chronologiquement, l'ordre des frappes dans le texte ne suit pas cette chronologie (Plane, Alamargot, & Lebrave, 2010). Ce qui aura été écrit au tout début de la phase de rédaction peut très bien se retrouver à la toute fin du texte. L'ensemble du processus, lorsqu'effectué sur un support numérique, peut être enregistré grâce à différents logiciels dédiés. *ScriptLog*, *JEdit*, *Translog*, *Eye and Pen* ou *InputLog* pour ne nommer que ceux-ci, permettent l'enregistrement du temps de l'écriture en accumulant une liste exhaustive des opérations effectuées dans un fichier *log* (Caporossi & Leblay, 2011).

La principale difficulté de l'analyse du processus d'écriture réside ici en la singularité des processus entre eux. Les caractéristiques du scripteur, son niveau de maîtrise de la langue et sa mémoire, le médium d'écriture, la tâche et le genre de la production écrite sont des variables qui influencent le temps de l'écriture (Lindgren & Sullivan, 2002).

2.2 Modélisation et analyse de données

L'étude du temps de l'écriture est caractérisée par l'acquisition d'un grand nombre de données enregistrées lors du processus d'écriture. Ces données sont trop nombreuses et complexes pour être analysées par des outils traditionnels (Caporossi & Leblay, 2011).

Dans un contexte d'analyse de données complexes, le processus simplifié par lequel les données sont acquises, traitées et analysées est repris par cette série d'actions : capture, partage, tri, analyse et visualisation des données (Manyika, et al., 2011). Certaines de ces actions ont déjà été étudiées individuellement et ont fait l'objet d'un développement logiciel visant à enregistrer le processus d'écriture et à en créer des représentations dans l'une de ses dimensions (Caporossi & Leblay, 2011). Chacun de ces logiciels est différent et est destiné à un objectif spécifique (Sullivan & Lindgren, 2014).

L'objectif des représentations de données est d'aider les chercheurs dans l'analyse, la compréhension de données et la recherche de motifs complexes. La visualisation de données est considérée comme étant un outil d'analyse (Manyika, et al., 2011). En voyant comment les données

interagissent, il est possible de découvrir et de comprendre les tendances et les changements temporels dans une base de données (Yau, 2011, p. xvi ; Minelli, Chambers, & Dhiraj, 2013, p. 110).

Plusieurs techniques de visualisation de données existent dans le but de faciliter l'utilisation et favoriser la recherche de motifs complexes. Ces techniques sont pluridisciplinaires et incluent les statistiques, les sciences cognitives, le graphisme, l'informatique et la cartographie (Kirk, 2012).

Lorsque les méthodes d'analyse de données sont appliquées aux sciences humaines, il est souvent question de la discipline des *Humanités Numériques*. Ce domaine multidisciplinaire consiste en « l'intersection de l'informatique et des disciplines des sciences humaines et des arts, des domaines interdisciplinaires de la culture et de la communication et des professions de l'éducation et des bibliothèques et des sciences de l'information » (Klein, 2014). Il est intéressant de noter que le premier sujet d'étude des Humanités Numériques fut l'étude des textes (Nyhan & Flinn, 2016).

2.2.1 Théorie des graphes

Un graphe est une représentation visuelle qui est constituée de points reliés ensemble par des lignes (Bondy & Murty, 1982, p. 1). Leur structure est utilisée dans des domaines variés puisqu'elle représente simplement des relations entre objets (Ahuja, Magnanti, & Orlin, 1993, p. 1). La terminologie est plutôt intuitive. Les points sont appelés *nœuds* tandis que les lignes sont appelées *arêtes*. Les nœuds peuvent représenter des villes sur une carte, des individus dans un réseau social ou encore simplement des informations reliées entre elles par les arêtes (Henley & Williams, 1973, p. 2). De manière plus formelle, un graphe est constitué d'un ensemble fini de *nœuds* et d'un ensemble *d'arêtes*. Un graphe peut être *orienté*, dans quel cas il a un ensemble *d'arcs* et non pas *d'arêtes* (Gross & Yellen, 1999). Un *arc* possède une origine et une destination.

De manière appliquée, en prenant l'exemple des réseaux sociaux, un graphe non orienté serait utilisé pour décrire une plateforme telle que *Facebook* pour décrire la relation mutuelle que deux individus partagent en étant amis. Un graphe orienté serait plutôt utile dans le cas de *Twitter* où la relation n'est pas nécessairement réciproque. Un usager peut très bien suivre un autre usager, sans que l'inverse soit vrai. S'il y a une relation de *précédence* entre des actions, là encore l'orientation des arcs est pertinente (Manyika, et al., 2011). À noter que la notion de précédence est plutôt

appelée *successivité* en linguistique, mais se réfère à la même notion de *suite d'événements dans le temps* (Doquet & Leblay, 2014).

Plusieurs terminologies ont été proposées dans la littérature pour les composants d'un graphe. De plus, certaines opérations, propriétés supplémentaires et invariants des graphes sont utilisés dans cette thèse.

Graphe

Soit $G = (N, A)$ un graphe, où N est l'ensemble de nœuds et A est l'ensemble d'arêtes. Soient n le nombre de nœuds et m le nombre d'arêtes. Une arête a liant les nœuds x et y est notée $a = [x, y]$.

Graphe orienté

Si $G = (N, A)$ est orienté, A est l'ensemble d'arcs et m le nombre d'arcs.

L'arc a dont l'origine est le nœud x et la destination est le nœud y sera noté (x, y) tandis qu'un arc dont l'origine est le nœud y et la destination est le nœud x sera noté (y, x).

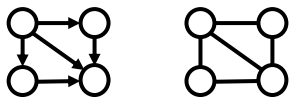

Figure 2.1 Graphe a) orienté et Graphe b) non orienté

À moins d'indications contraires, toutes les notions définies ci-après s'appliquent autant aux graphes non orientés qu'aux graphes orientés.

Adjacence et degré

Deux nœuds sont dits *adjacents*, ou *voisins*, s'ils sont reliés par une arête ou un arc.

Le *degré* d'un nœud x est le nombre de voisins que possède celui-ci (Saha Ray, 2013, p. 4). On note $d(x)$ cet entier (Agarwal & Singh, 2009, p. 4.3.2). Un nœud est dit isolé si son degré est 0 (Agarwal & Singh, 2009, p. 4.3.3). Le degré maximum d'un graphe G est noté $\Delta(G)$ (Fournier, 2010, p. 28).

Dans le cas d'un graphe orienté, les *voisins* d'un arc se séparent en deux catégories. Dans le cas où les nœuds *x* et *y* sont liés par l'arc (x, y), *x* et *y* sont bien *voisins* mais plus spécifiquement, *x* est le *prédécesseur* de *y* et *y* est le *successeur* de *x*. On note $d^-(x)$ le nombre de prédécesseurs de *x* et $d^+(x)$ le nombre de successeurs de *x*.

Graphe sous-jacent

Le graphe sous-jacent d'un graphe orienté G est le graphe qui résulte de l'omission de l'orientation des arcs tel qu'un arc devient simplement une arête, sans destination ni origine (Gross, Yellen, & Zhang, 2015, p. 1.1.1 Digraphs).

Sous graphe

Un graphe H est un sous-graphe de G si tous les nœuds et les arcs de H sont dans G. Un sous-graphe H de G est dit induit si chaque arc de G dont l'origine et la destination sont dans H est aussi un arc dans H (Balakrishnan & Ranganathan, 2012, p. 8).

Dans la figure ci-dessous, le graphe b) est sous-jacent au graphe a), le graphe c) est un sous-graphe induit du graphe b) et le graphe d) est un sous-graphe non induit de b).

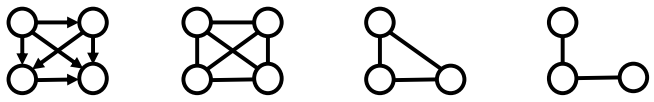

Figure 2.2 Exemple de sous-graphes

Chemin

Un chemin dans un graphe G est une suite $(x_0, a_1, x_1, \dots, a_k, x_k)$ où *k* est un entier plus grand que zéro, x_i sont des nœuds de G et a_i sont des arêtes de G telles que pour $i = 0, \dots, k-1$, x_i et x_{i+1} sont les nœuds liés par l'arête a_{i+1}. L'entier *k* représente la longueur du chemin (Fournier, 2010, p. 22).

Un chemin est dit élémentaire s'il ne passe pas deux fois par le même nœud (Fournier, 2010, p. 24).

Connexité

Un graphe G est dit connexe s'il existe un chemin entre chaque paire de nœuds (Fournier, 2010, p. 29).

Un sous-graphe *S* maximal de *G* qui respecte la propriété de connexité est appelé une composante connexe. Le terme maximal signifie qu'aucun autre sous-graphe connexe de G ne contient S (Fournier, 2010, p. 29).

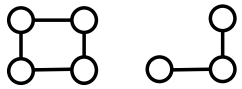

Figure 2.3 Graphe composé de 7 nœuds répartis en deux composantes connexes

Distance

La distance $d(x, y)$ entre deux nœuds *x* et *y* dans *G* est la longueur du chemin le plus court les joignant le cas échéant et représente la plus petite quantité d'arcs entre *x* et *y* (Harary, 1969, p. 14).

Diamètre

Le diamètre est la valeur de la plus longue distance entre deux nœuds d'un graphe (Gross & Yellen, 1999, p. 38).

$$diam(G) = \max_{x,y \in N} \{d(x, y)\}$$

Contraction

Deux nœuds *x* et *y* sont contractés si ces deux nœuds sont remplacés par un seul nœud *z* tel que toutes les arêtes qui étaient adjacentes à *x* et *y* sont maintenant adjacentes à *z* (Aggarwal & Wang, 2010, p. 221).

Dans la figure ci-après, les nœuds en rouge dans le graphe a) sont contractés dans le graphe b).

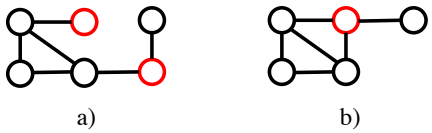

Figure 2.4 Exemple de contraction de nœuds

2.2.2 Visualisation des données comme outil d'analyse

Lorsqu'une grande quantité de données est traitée, l'utilisation de visualisations est un excellent moyen d'explorer et d'analyser une situation donnée (Yau, 2011). Les visualisations peuvent aider à identifier les tendances, les structures parmi les données, les irrégularités et les relations sur une période donnée (Minelli, Chambers, & Dhiraj, 2013). Une fois que l'analyste de données a compris et pris connaissance des données visualisées, il peut tirer des conclusions à partir des tendances découvertes (Dzemyda, Kurasova, & Zilinskas, 2013). Cette méthode d'analyse a ses défauts : la capacité humaine à traiter beaucoup de données est limitée (Manyika, et al., 2011) et parfois les chercheurs tendent à voir des relations entre des éléments aléatoires et non apparentés. Ce phénomène est appelé *apophénie* (Boyd & Crawford, 2012). La méthode à employer afin d'éviter ces biais d'interprétation, est d'utiliser en parallèle des modèles mathématiques qui permettent d'avoir les bases nécessaires pour effectuer une analyse rigoureuse tout en ayant une compréhension efficace des données (Moller, Hamann, & Russell, 2009).

Présenter des données avec des techniques graphiques peut cependant ne pas être suffisant pour aider l'utilisateur à trouver des tendances (Tory & Moller, 2004). La manière dont les gens interagissent avec les outils de visualisation et perçoivent les données affichées influence la façon dont les utilisateurs comprennent les relations et les tendances potentielles (Tory & Moller, 2004). Par conséquent, avant de créer une nouvelle visualisation, il faut considérer le facteur humain et les moyens d'améliorer son efficacité (Card, Mackinlay, & Shneiderman, 1999, p. 16).

Plusieurs recherches ont été réalisées dans le but de comprendre le support cognitif que les visualisations apportent au cerveau humain tout en essayant de comprendre visuellement les données (Tory & Moller, 2004). Afin de créer de bons outils de visualisation, deux concepts apparentés qui sont la *représentation des données* et la *présentation des données* doivent être considérés.

- Le choix d'un graphique ou du type de graphique qui illustre les variables des données est alors considéré comme l'étape de *représentation des données* (Kirk, 2012).
- Le format de livraison, l'apparence générale et la conception, y compris le choix des couleurs et les fonctions interactives font partie de la *présentation des données* (Aigner, Miksch, Schumann, & Tominski, 2011).

2.2.2.1 Représentation des données

La *représentation des données* consiste à choisir la bonne façon physique de représenter les données (Kirk, 2012). Afin de créer une version visuelle des données, la première étape est de réfléchir au type de variables qui seront représentées. Parallèlement, il faut réfléchir à l'objectif de la représentation : sera-t-elle utilisée pour l'analyse exploratoire, pour l'analyse prédictive ou simplement pour la présentation des résultats (Aigner, Miksch, Schumann, & Tominski, 2011, p. 5).

Si le temps est une variable, le fait qu'il soit considéré a) comme un nombre absolu ou b) qu'il soit interprété de façon relative, modifie la façon dont il peut être représenté.

a) Quand le temps est noté comme un nombre absolu, le plus souvent les variables représentées dans un graphique à deux axes, le temps étant affichée généralement sur l'axe des *x* tandis que la variable dépendante du temps est sur l'axe des *y* (Aigner, Miksch, Schumann, & Tominski, 2011, p. 76).

b) Dans le cas où le temps peut être interprété, des variables visuelles sont utilisées telles que la couleur et la transparence pour illustrer par exemple la durée entre deux événements (Aigner, Miksch, Schumann, & Tominski, 2011, p. 78).

Les visualisations de données sont traditionnellement statiques (Kirk, 2012). La manière d'illustrer correctement la façon dont les données interagissent est de montrer les changements dans le temps (Bartram L. , 1997). Parce que les visualisations statiques sont en deux dimensions, il est compliqué de les représenter de façon efficace avec des problèmes ayant plus de deux dimensions (Minelli, Chambers, & Dhiraj, 2013). Par conséquent, l'une des meilleures façons de visualiser les changements spatio-temporels est d'animer les données (Yau, 2011). Les représentations spatio-temporelles devraient être basées sur des sources interactives qui pourraient permettre à l'utilisateur de manipuler les données afin de permettre une compréhension plus approfondie (Ohlhorst, 2012).

Ces fonctionnalités aident l'utilisateur à s'impliquer dans une méthodologie dynamique de résolution de problèmes et vont plus loin dans son analyse. Permettre la manipulation de variables et de paramètres est un moyen de montrer plusieurs dimensions au sein d'un même affichage et de faciliter la combinaison de différentes variables lors de l'exploration de données visualisées (Kirk, 2012).

Pour aider l'utilisateur à étudier les données, certains auteurs ont suggéré d'inclure des tâches pouvant être effectuées sur les données visualisées (Shneiderman 1996, cité par (Aigner, Miksch, Schumann, & Tominski, 2011). Ajuster la vue avec les possibilités d'avoir une vue d'ensemble et/ou de zoom pour changer le degré de précision permet à l'utilisateur de voir l'ensemble de données ou une partie plus spécifique et peut être pertinente pour l'analyse des mêmes données et de leur structure selon différents objectifs. Filtrer et extraire des variables permet à l'utilisateur de supprimer des informations non désirées ou d'extraire des données avec des paramètres de requête comme le tri des données, la sélection d'une fenêtre temporelle spécifique, d'un espace géographique sélectionné ou d'autres requêtes statistiques. Parfois, l'utilisateur de ces visualisations a besoin d'obtenir des informations spécifiques pour confirmer ou infirmer une corrélation, un modèle ou une relation dans les données.

Lorsque les données représentent des informations multidimensionnelles avec des phénomènes variant dans le temps, il est utile d'accéder aux différentes structures et à leurs couches sous-jacentes. La combinaison de la visualisation et de l'analyse des données est une méthodologie complémentaire pour comprendre les structures et les relations trouvées (Moller, Hamann, & Russell, 2009).

2.2.2.2 Présentation des données

La représentation de données n'est pas le seul aspect impliqué lors de la création d'un support visuel pour décrire et comprendre les données. L'ergonomie globale de la visualisation permet de maximiser le potentiel d'analyse (Heer & Bostock, 2010).

La présentation des données implique également la conception globale de l'outil de visualisation. Bien que ce soit un aspect plus central lorsque le but de la visualisation est de présenter certains résultats, certaines caractéristiques méritent d'être considérées dans un modèle exploratoire (Unwin, Chen, & Hardle, 2008).

Selon Bartram (1997), une combinaison appropriée de formes, de symboles, de couleurs, de tailles et de positions devrait être choisie puisque ce type d'information visuelle n'exige pas d'effort cognitif (Blanchard, 2005). Une combinaison optimale de ces codes visuels augmente la rapidité et l'efficacité de l'analyste qui consulte la visualisation, lorsque celui-ci tente d'y déceler des tendances.

Alors que les couleurs peuvent aider à illustrer les variables, comme il est explicité dans la section de représentation des données, une autre considération, lors du choix de ces couleurs, serait de rendre l'outil de visualisation accessible au plus grand nombre de chercheurs dans le choix de couleurs qui prennent en compte la façon dont les personnes daltoniennes les voient (Aigner, Miksch, Schumann, & Tominski, 2011, p. 89).

2.3 Analyses et visualisations du processus d'écriture

La prochaine sous-section présentera les différentes visualisations du processus d'écriture existantes ainsi que certaines analyses qui en sont faites.

2.3.1 Visualisations

Les représentations du processus d'écriture se font majoritairement de manière statique. La seule exception est la recomposition du processus qui a été réalisée en visualisant chronologiquement les opérations effectuées par un scripteur grâce au fichier *log* (ITEM, 2014). Bien qu'animée, cette visualisation n'est autre qu'un *film* du processus, dénué de tout traitement préalable des données.

Les représentations actuelles du processus d'écriture tournent autour d'un ou plusieurs objectifs particuliers comme la révision, l'aspect temporel ou la rétrospection du scripteur (Latif, 2008). Puisque le processus d'écriture comprend un aspect spatio-temporel (Stromqvist, Holmqvist, Johansson, & Karlsson, 2006), aucune des représentations actuelles n'arrive à bien cerner le problème de manière plus complète.

Chaque frappe est caractérisée par un ensemble de variables et possède une valeur de position pour chacune de celles-ci. La *temporalité* est définie par la valeur temporelle des opérations. Par conséquent, deux opérations ne peuvent pas avoir la même position de *temporalité*. La *chronologie* est l'ordre des opérations dans le fichier *log*. Il s'agit d'une simplification de la variable temporelle. Enfin, la *spatialité* concerne l'emplacement où les opérations ont été faites géographiquement dans

le texte. Plus d'une variante de spatialité existe : ainsi, la *spatialité relative* est la position d'une opération d'écriture telle qu'elle est enregistrée dans le fichier *log*; celle-ci concerne l'endroit où elle a été réalisée en considérant l'état du texte à ce moment précis. La *spatialité absolue* est la position de l'opération d'écriture par rapport à toutes les opérations qui ont été faites dans le texte.

Dans certains cas, les chercheurs incluent également des variables qui sont exclues du processus enregistré de l'écriture, mais qui contribuent au processus de rédaction. Parmi celles-ci, une fonction de suivi des yeux permet de savoir où le scripteur regarde lorsqu'il rédige (Wengelin, et al., 2009). Bien qu'il s'agisse d'écriture à la main, le logiciel *Eye and Pen* peut enregistrer où l'œil regarde et à quel moment (Alamargot, Caporossi, Chesnet, & Ros, 2011). Dans ce cas, les variables sont encore plus complexes.

Les visualisations existantes du processus d'écriture sont bidimensionnelles. Même si la dimension spatio-temporelle du processus est un aspect à analyser et à comprendre (Stromqvist, Holmqvist, Johansson, & Karlsson, 2006), aucune des visualisations actuelles n'apporte de solutions satisfaisantes.

Un autre aspect du processus d'écriture est l'intérêt des aspects *micro* (le détail des opérations effectuées) et *macro* (la structure globale du processus) de l'avant-texte. Puisque ces deux aspects ne peuvent pas être visuellement représentés sur un même diagramme, ou graphe, au même instant, les chercheurs utilisent généralement plusieurs représentations simultanément afin de capter le processus dans son ensemble (Breetvelt, Van den Bergh, & Rijlaarsdam, 1994 ; Doquet C. , 2003 ; Van Waes & Schellens, 2003 ; Latif, 2008 ; Cox, Ortmeier-Hooper, & Tirabassi, 2009 ; Alamargot, Caporossi, Chesnet, & Ros, 2011 ; Caporossi & Leblay, 2011; Leijten & Van Waes, 2013; Southavilay, Yacef, Reimann, & Calvo, 2013).

Quand il s'agit de la représentation du processus d'écriture, deux variables surgissent nécessairement : le temps et l'espace. La représentation linéaire se concentre sur l'aspect chronologique et représente les différentes actions du scripteur sur le texte. Les représentations *SIG* (Système d'Information Géographique) pour leur part mettent délibérément l'accent sur l'espace. Entre ces deux représentations extrêmes, de nombreuses variantes peuvent exister.

2.3.1.1 Représentations linéaires

La représentation linéaire permet de voir de près les opérations effectuées lors de l'écriture du texte. De nombreuses représentations linéaires existent, mais elles ont des propriétés similaires. La variable spatiale est explicite alors que la variable chronologique est présente sans être détaillée. Bien que le terme *représentation* soit utilisé, le résultat visuel n'est pas graphique, il s'agit plutôt d'une transcription.

2.3.1.1.1 Notation-S

La *Notation-S* est un type de représentation linéaire qui est entre autres utilisé par les logiciels *Inputlog* (Lindgren & Sullivan, 2002) et *trace-it/JEdit* (Kollberg, 1997). Elle indique l'évolution du texte en démontrant la production du texte, les insertions et révisions (Lindgren & Sullivan, 2002). Elle a été produite pour permettre une meilleure compréhension des révisions et fut inspirée par une notation informelle des révisions qui avait été développée par Matsuahi en 1987 (Kollberg, 1996).

Cette représentation met l'emphase sur les actions éditées en les représentant en tant qu'opérations indépendantes. Elle ne donne cependant aucune référence à la durée des opérations. Le tableau ci-après explique le fonctionnement des symboles utilisés pour illustrer les opérations de révision (Kollberg, 1996).

Tableau 2.1 Symboles de la *Notation-S*

$\|_i$	L'interruption avec le numéro séquentiel *i*
{texte inséré}^{i}	Une insertion qui se produit après l'interruption numéro *i*
[texte supprimé]^i	Une suppression qui se produit après l'interruption numéro *i*

L'exemple d'utilisation ci-après provient de Kollberg (1997) :

I am writing a {*short*}^{1} *text.* $|_1$ *It will* [*probably*]^2 $|_3$ *be revised* [*somewhat*]^3 *later.* $|_2$ *Now* [*I am* $|_4$]^4 *it is finished.*

La version finale de ce texte est

I am writing a short text. It will be revised later. Now it is finished.

Cette représentation permet au chercheur de détecter où les interruptions et les révisions sont faites au cours de la session d'écriture. Cependant, en raison du faible niveau d'information disponible dans cette représentation, une interprétation des actions du scripteur est nécessaire pour compléter l'analyse (Kollberg, 1996).

Voici les caractéristiques de visualisation de la *Notation-S :*

Représentation des données	
Quelles sont les variables utilisées ?	Chronologie des opérations, spatialité absolue.
Comment sont-elles représentées?	C'est simplement le texte écrit avec des symboles qui indiquent où sont faites les révisions dans le texte et dans quel ordre (Kollberg, 1997) .
Quel est le degré de précision?	Précis, le texte et ses modifications sont facilement lisibles.
Présentation des données	
Quelles sont les caractéristiques de la présentation des données?	La *Notation-S* est intégrée dans un programme interactif d'analyse de révision. Il est notamment possible pour l'utilisateur de naviguer entre différentes versions des révisions et de générer des statistiques sur la session d'écriture.

2.3.1.1.2 Variation de la Notation-S

Il existe une version plus détaillée de la *Notation-S*, qui inclut des références aux sources utilisées lors de la rédaction du texte. Les portions copiées du texte sont soulignées et différentes polices de caractères sont utilisés pour désigner les sources (Perrin D. , 2003). Lorsqu'elle a été créée, cette représentation était utilisée conjointement avec le *diagramme de progression*, qui est présenté dans la Section 2.3.1.2.1 pour étudier comment les journalistes écrivent lorsqu'ils utilisent des sources, de là l'importance de l'identification des portions copiées du texte.

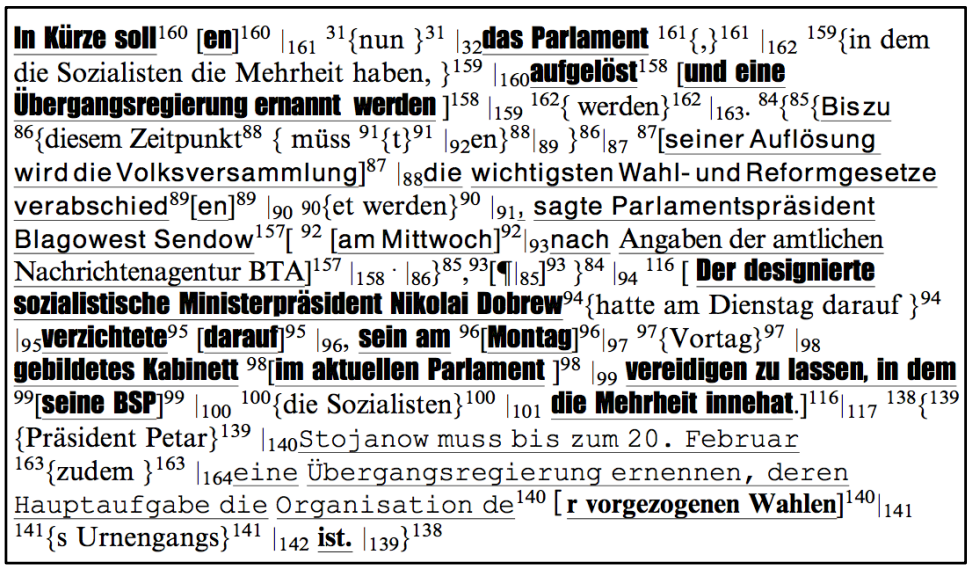

Figure 2.5 Variation de la *Notation-S* (Perrin D. , 2003)

Voici les caractéristiques de visualisation de cette variation de la *Notation-S :*

Représentation des données	
Quelles sont les variables utilisées ?	Chronologie des opérations, spatialité absolue.
Comment sont-elles représentées?	C'est simplement le texte écrit avec des symboles qui indiquent l'ordre et l'endroit dans le texte où sont faites les révisions. Dans la version avec *sources*, l'utilisation de polices différentes aide l'utilisateur à remarquer l'origine du texte affiché (Perrin D. , 2003).
Quel est le degré de précision?	Précis, le texte et ses modifications sont facilement lisibles.
Présentation des données	
Quelles sont les caractéristiques de la présentation des données?	Les différentes polices utilisées dans la version *sources* sont suffisamment différentes pour que le lecteur puisse voir la différence, mais lorsque la quantité de sources utilisées est supérieure à 3-4, cela devient difficile pour l'analyste.

2.3.1.1.3 Transcription linéaire génétique

Une troisième variation de la *Notation-S* appelée *transcription linéaire génétique* inclut l'utilisation de la souris, du pavé tactile ou des flèches par le scripteur (Leblay C. , 2009 ; Doquet C. , 2014). Les symboles de révision ne sont pas les mêmes que ceux utilisés par la *Notation-S* et certains d'entre eux sont empruntés à un logiciel appelé *JEdit* (Leblay C. , 2009). Les symboles utilisés pour générer cette représentation sont présentés dans le tableau suivant :

Tableau 2.2 Symboles de la *transcription linéaire génétique*

Conventions de transcription	Opérations de type 1, À la suite du déjà écrit	Opérations de type 2, Par retour dans le déjà écrit
Ajouts	Ajout 1 Simple	Ajout 2 (Simple +) **Gras**
Suppressions	Suppressions 1 Suppression immédiate ~~*Barrée avec italique*~~ - Suppression différée ~~Barrée sans italique~~	Suppressions 2 Suppression immédiate ~~***Barrée avec italique en gras***~~ - Suppression différée ~~**Barrée sans italique en gras**~~
Remplacements	Remplacement 1 ~~Remplacé~~remplaçant	Remplacement 2 ~~**Remplacé**~~**remplaçant**
Déplacements	Déplacement 1 ~~Déplacé~~déplaçant	Déplacement 2 ~~**Déplacé**~~**déplaçant**
Texte ^{i}	Une révision qui se produit après l'interruption numéro *i*	
🖱 ▫ ↔ ↕	Mouvements de la souris, du pavé tactile et des flèches	

TRANSCRIPTION LINEAIRE (.DOC)

Que ~~*me vionet*~~**'est-ce qui me vient en tête spontanément quand on me donne la tâche d'**~~*'*~~**e décrire un environnement idéal?**[18] Be~~**u**~~[1]**au**[2]coup de verdure←[1]→[2], peu de pollution, suffisamment d'espace pour que t~~*uot*~~out un chacun se trouve ~~*`l'i*~~à l'aise **a**[19]~~*sv***a**[4]vec soi-m^~~*me*~~ême et~~**v**~~[3] avec les autres🖱[3]🖱[4]🖱. Le climat? Je dois dire que les saisons disctinctes et variées typiques du Nord m~~**ae**~~[5] plaisent🖱[5]🖱, etv que c~~*e*~~'est vquelque chose que ~~*nje*~~ je voudrais avoir dans un **ue**[10]nvironnement idéal. ~~*Do*~~Autrement **n**[11]dit, un été chaud et ensoileillé ~~**avee**~~sans oublier évidemment les journées lumineuses et e**x**[6]trêmement longues🖱[6]🖱, ~~*`la*~~comme en Finlande en été, un hiver blanc, avec du **ds**[12]oleil et de la n~~*aige*~~~~*ei*~~~~*d*~~ge, et encore d**ueux**[13] sais~~*io*~~ons de "transmisson", de passage on peut dire, pour passer du froid,[14] au chaud ou ~~*läinverse*~~inversemen~~*rt*~~t: le pri~~*np*~~ntemps et l'a~~*i*~~utomne.

Figure 2.6 *Transcription linéaire génétique* (Leblay C. , 2009)

Voici les caractéristiques de visualisation de la *transcription linéaire génétique :*

Représentation des données

Quelles sont les variables utilisées ? Chronologie des opérations, spatialité absolue.

Comment sont-elles représentées? C'est simplement le texte écrit avec des symboles qui indiquent où dans le texte sont faites les révisions et dans quel ordre. L'utilisation de la souris, du pavé tactile ou des flèches est également indiquée par des symboles (Leblay C. , 2009). Les

	symboles de révisions ne sont pas les mêmes que ceux utilisés dans la *Notation-S*.
Quel est le degré de précision?	Précis, le texte et ses modifications sont facilement lisibles.
Présentation des données	
Quelles sont les caractéristiques de la présentation des données?	L'utilisation d'icônes pour représenter la position du curseur est très conviviale. Il est facile de lire le texte final s'il n'y a pas beaucoup de révisions. Dans le cas où une même portion de texte est retravaillée plusieurs fois, la représentation devient plutôt dense.

Cette version de la *Notation-S* a été utilisée pour étudier les traces de l'écriture en ligne, les opérations d'écriture (insertions, suppressions, remplacements et déplacements), leur chronologie et leur impact sur l'évolution du texte (Leblay, C. , 2009; Perrin D. , 2019).

2.3.1.1.4 Représentation linéaire de Scriptlog

Le logiciel *Scriptlog* possède sa propre représentation linéaire qui est d'une certaine manière une simplification du fichier *log* et peut être utilisée pour suivre le processus d'écriture au fur et à mesure (Wengelin, et al., 2009). Les frappes sont présentées en ordre chronologique. Toutes les frappes, mouvements de souris et les pauses sont affichées mais il est difficile de savoir exactement à quel endroit dans le texte elles sont effectuées (Wengelin, et al., 2009). Le but de cette représentation est de mettre en valeur les pauses et les révisions.

L'exemple suivant a été écrit en suédois et provient de Wengelin *et al*. (2009) :

<5.762>Filmen jag såg nyss handla<BACKSPACE6>bistro<BACKSPACE2>od av korta klipp<BACKSPACE4>s<BACKSPACE2>sekvenser ur några ung<3.639>domars vardag<9.930> <4.027>.Nästan<3.825><BACKSPACE9>. I princip alla sekvenser ha<BACKSPACE2>visar<BACKSPACE>de hur ungdomar utsätts för jobbiga situationer u<BACKSPACE>avsiha<BACKSPACE2>n a<BACKSPACE2>a kamrater.

La version finale de ce processus est :

> *Filmen jag såg nyss bestod av korta sekvenser ur några ungdomars vardag. I princip alla sekvenser visade hur ungdomar utsätts för jobbiga situationer av sina kamrater.*

Voici les caractéristiques de la représentation linéaire de *Scriptlog* :

Représentation des données	
Quelles sont les variables utilisées ?	Chronologie, et un certain apport de temporalité puisque la durée des pauses est indiquée.
Comment sont-elles représentées?	Les chiffres entre les parenthèses <> indiquent une pause et sa durée. <BACKSPACE> fait référence à la suppression d'une frappe tandis que <BACKSPACE6> signifierait que le scripteur ait supprimé 6 séquences de frappes puisque ce chiffre suit la mention BACKSPACE. Cette représentation du processus d'écriture ne permet pas de bien voir les retours dans le texte (Wengelin, et al., 2009).
Quel est le degré de précision?	Précis, les caractères du texte sont clairement affichés.
Présentation des données	
Quelles sont les caractéristiques de la présentation des données?	La présentation est limitée à certains symboles utilisés et l'utilisation de majuscules pour certains aspects.

Le principal désavantage de cette représentation est qu'elle ne permet pas une lecture intuitive (Wengelin, et al., 2009).

2.3.1.1.5 Représentation Timeline

Timeline est une représentation ayant comme but de démontrer ce que le scripteur fait et où il regarde lorsqu'il rédige son texte. Cette représentation utilise *Scriptlog* et des données de suivi de l'œil (Wengelin, et al., 2009). Elle n'est considérée que linéaire de par sa lecture des frappes, qui se fait linéairement.

Les deux sections du haut font référence aux données de suivi de l'œil. La section est foncée si le scripteur regarde à cet endroit. Au tout début, le scripteur regarde l'écran puisque cette section est foncée. La portion SLog indique qu'une frappe a été enfoncée. Une barre verticale indique que la frappe a été enfoncée à ce moment-là et en tout petit, le caractère associé à la frappe est identifié.

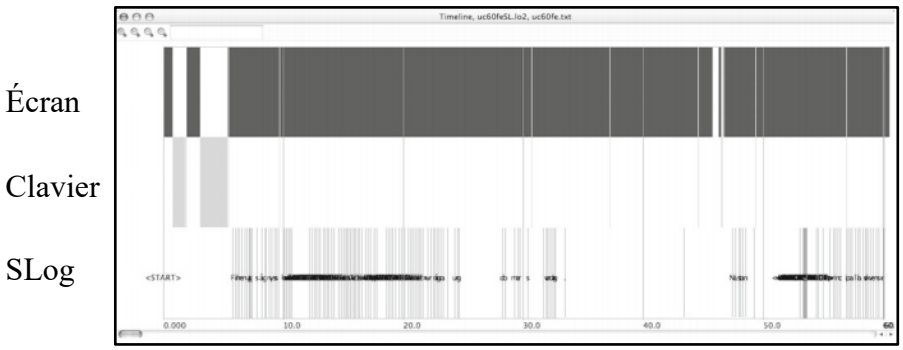

Figure 2.7 Représentation *Timeline* (Wengelin, et al., 2009)

Voici les caractéristiques de la représentation *Timeline* :

Représentation des données	
Quelles sont les variables utilisées ?	Temporalité et chronologie de chaque frappe.
Comment sont-elles représentées?	L'axe des *x* montre le temps. L'axe des *y* est séparé en 3 sections. Les sections Écran et Clavier représentent l'endroit où le scripteur regarde. Les sections sombres indiquent où porte le regard, à ce moment précis. La section SLog est dédiée aux touches enfoncées. Les opérations sont représentées par une ligne fine et le caractère exact est indiqué sur la ligne (Wengelin, et al., 2009)
Quel est le degré de précision?	Extrêmement précis. Il est possible de voir exactement l'opération d'écriture effectuée, mais il est impossible de lire ou de comprendre ce qui a été écrit.
Présentation des données	
Quelles sont les caractéristiques de la présentation des données?	La présentation est assez minimaliste. L'utilisation d'une teinte sombre à l'endroit où le rédacteur regarde est efficace. La taille et l'emplacement des frappes (caractères) peuvent cependant ne pas être optimaux, car il est difficile pour l'utilisateur de voir réellement les informations.

Cet outil de visualisation a été créé pour pallier au manque d'information de la représentation linéaire par rapport à ce que le scripteur fait lorsqu'il effectue une pause (Wengelin, et al., 2009). Conséquemment, cet outil est utilisé en complément à la représentation linéaire *Scriptlog*.

2.3.1.2 Visualisations d'ensemble du processus d'écriture

Les visualisations d'ensemble sont celles qui illustrent le flux global du processus. Il est presque impossible de savoir à quoi ressemblent le texte et les opérations dans cette perspective (Caporossi & Leblay, 2011). Ces visualisations sont principalement inspirées des *SIG* en raison des similitudes entre les domaines tels que l'identification géographique et temporelle des caractères (Lindgren & Sullivan, 2002). Dans leur domaine d'origine, les *SIG* sont utilisés à la fois pour la visualisation et l'exploration de données (Lindgren E. , Sullivan, Lindgren, & Spelman Miller, 2007).

2.3.1.2.1 Diagramme de progression

Le but de cette représentation est de mettre l'emphase sur les révisions qui sont effectuées dans le processus (Lindgren & Sullivan, 2002). Le *diagramme de progression* a été créé pour être un complément à la *Notation-S* pour permettre une meilleure analyse du processus d'écriture, notamment lorsque le scripteur utilise des sources (Perrin D. , 2003).

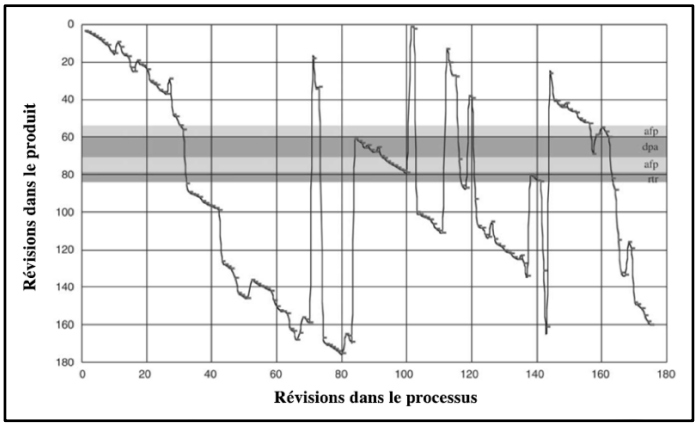

Figure 2.8 *Diagramme de progression* (Perrin D. , 2003)

L'axe des *y* est lu de haut en bas, tel qu'on lit un texte.

Voici les caractéristiques du *diagramme de progression* :

Représentation des données	
Quelles sont les variables utilisées ?	Chronologie et spatialité relative. Cependant, les variables ne sont pas les frappes, mais les révisions qui ont eu lieu durant le processus. Les sources utilisées sont également incluses.
Comment sont-elles représentées?	Sur deux axes, le nombre de révisions en cours (axe *x*) et le nombre de révisions du produit (axe *y*) : l'axe 'révisions du produit' aide le chercheur à connaître le nombre de révisions effectuées sur une section spécifique du texte.

	Les acronymes «AFP» « DPA » et « RTR » font références aux sources. Par exemple, «AFP» signifie que cette portion de texte est un copié-collé de l'Agence France Presse (Perrin D. , 2003).
Quel est le degré de précision?	Imprécis, c'est un aperçu de l'aspect révisionnel du processus. Ce n'est pas une représentation de l'ensemble du processus.
Présentation des données	
Quelles sont les caractéristiques de la présentation des données?	Les noms des axes sont bien identifiés. Les sections en gris clair et foncé se réfèrent à une section spécifique du texte (spatialité) qui a été sélectionnée sur l'ordinateur et est visualisée sur l'écran sous la forme de la *Notation-S*.

La force du *diagramme de progression* est de présenter l'ordre des révisions par rapport au texte final. Son principal défaut est qu'elle est basée seulement sur la position relative de la révision, en n'ayant aucune relation au temps (Lindgren & Sullivan, 2002).

2.3.1.2.2 Représentation Au fil de la plume

Le logiciel *Genèse du texte* fournit cette représentation appelée *Au fil de la plume* (Foucambert D. , 1995), qui appartient au même mode de représentation que les *SIG* (Caporossi & Leblay, 2011). L'accent est mis ici sur la position du curseur dans le texte et le suivi (Doquet C. , 2003).

Elle a été créée en tant qu'outil pédagogique pour aider à comprendre comment les élèves écrivent, et par la suite, pour les aider à mieux apprendre (Doquet C. , 2003).

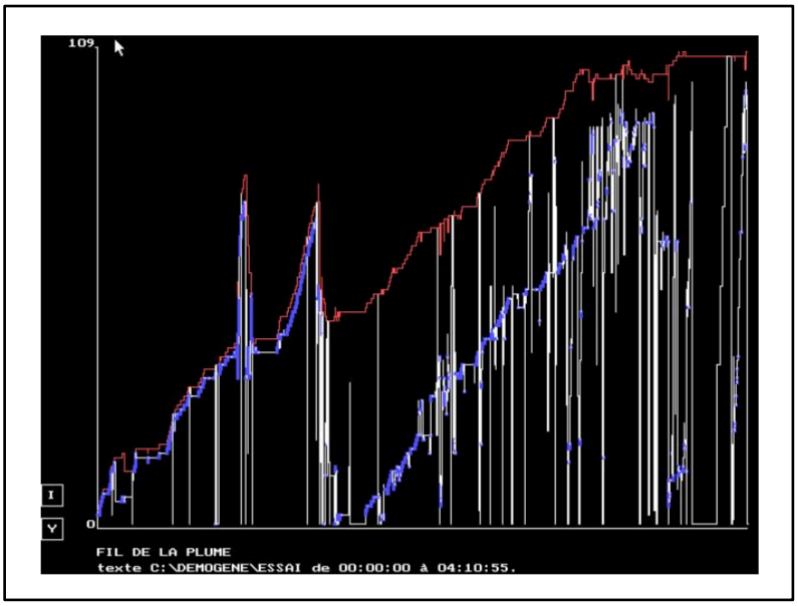

Figure 2.9 Représentation *Au fil de la plume* (Doquet C. , 2003)

Voici les caractéristiques de la représentation *Au fil de la plume* :

Représentation des données	
Quelles sont les variables utilisées ?	Temporalité et spatialité relative.
Comment sont-elles représentées?	L'axe des *x* représente le temps tandis que l'axe des *y* représente le nombre de lignes de texte (Doquet C. , 2003).
Quel est le degré de précision?	Imprécis, c'est un aperçu du processus.
Présentation des données	
Quelles sont les caractéristiques de la présentation des données?	Les noms des axes ne sont pas bien identifiés. La présentation globale n'a pas été optimisée pour le lecteur. Le fond sombre force l'œil à ne traiter qu'avec les couleurs dominantes et n'est donc pas optimal. Il est préférable d'utiliser les couleurs les plus fortes pour les données sur lesquelles l'utilisateur doit porter son attention (Kirk, 2012).

2.3.1.2.3 Graphe LS

Cette visualisation montre la même information que la *Notation-S*, mais l'utilise différemment que le *diagramme de progression* (Lindgren & Sullivan, 2002). Cette représentation a été largement inspirée par le travail de Perrin (2003) et a été par la suite utilisée en tant que modèle pour le *diagramme de progression* (Leijten & Van Waes, 2006).

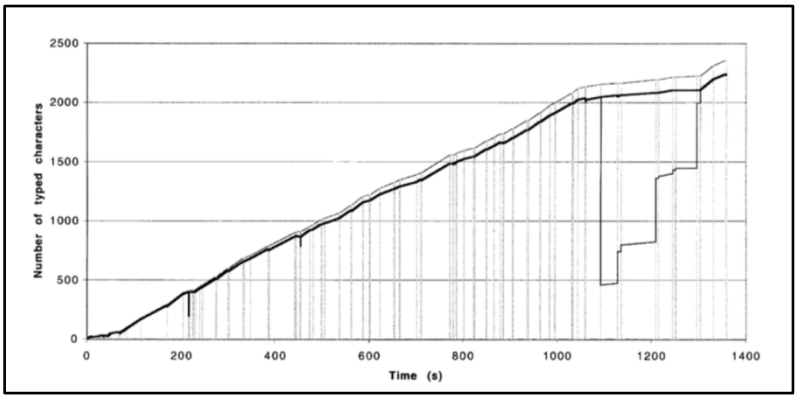

Figure 2.10 Graphe *LS* (Lindgren & Sullivan, 2002)

Voici les caractéristiques du graphe *LS* :

Représentation des données	
Quelles sont les variables utilisées ?	Temporalité et spatialité relative.
Comment sont-elles représentées?	L'axe des x représente le temps, tandis que l'axe des y représente le nombre total de caractères tapés. La ligne du haut montre la progression du nombre total de caractères tapés. La ligne centrée (la plus sombre) est la longueur du texte en fonction du temps. Enfin, la ligne inférieure est la position du curseur dans le texte (Lindgren & Sullivan, 2002).
Quel est le degré de précision?	Imprécis, c'est un aperçu du processus.
Présentation des données	
Quelles sont les caractéristiques de la présentation des données?	Les noms des axes sont bien identifiés. Il serait utile d'identifier clairement quelle couleur de ligne correspond à quel aspect du texte. Cette représentation est purement statique, la seule façon de savoir exactement ce qui se passe dans un certain point est de le rechercher manuellement dans les données et d'examiner cet événement en détail (Lindgren E. , Sullivan, Lindgren, & Spelman Miller, 2007).

La principale différence par rapport au *diagramme de progression* est la nature des opérations identifiées. Le graphe *LS* est plus général que le *diagramme de progression* et il représente le nombre d'opérations effectuées dans le temps.

Cette représentation a été créée pour comparer un ensemble de scripteurs et de tâches d'écriture tout en démontrant la densité des révisions (Lindgren & Sullivan, 2002).

2.3.1.2.4 Représentation inspirée des *Systèmes d'Information Géographiques (SIG)*

Considérant toutes les forces du *SIG*, cette visualisation a également été utilisée. Comparé au graphe *LS* qui est purement statique et pour lequel il est impossible d'avoir automatiquement des aperçus sur un point précis du graphique, le *SIG* permet au chercheur d'avoir un aperçu instantané (Lindgren E. , Sullivan, Lindgren, & Spelman Miller, 2007).

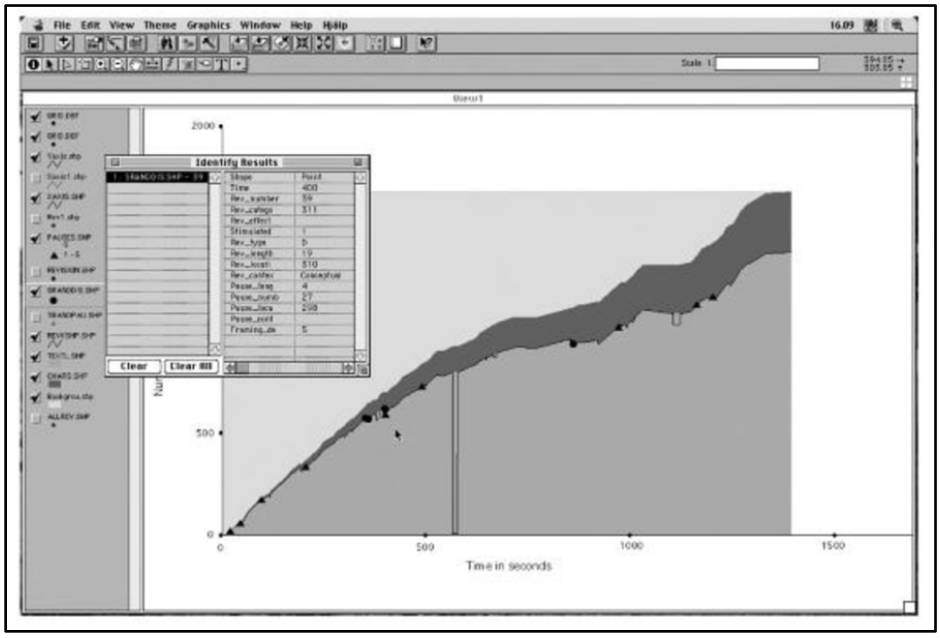

Figure 2.11 Représentation inspirée des *SIG* (Lindgren E. , Sullivan, Lindgren, & Spelman Miller, 2007)

Voici les caractéristiques de la représentation inspirée des *SIG:*

Représentation des données	
Quelles sont les variables utilisées ?	Temporalité et spatialité relative.
Comment sont-elles représentées?	L'axe des x représente le temps, tandis que l'axe des y représente le nombre total de caractères tapés. La ligne du haut montre la progression du nombre total de caractères tapés. La ligne centrée (la plus sombre) est la longueur du texte en fonction du temps. Enfin, la ligne inférieure est la position du curseur dans le texte (Lindgren E. , Sullivan, Lindgren, & Spelman Miller, 2007).
Quel est le degré de précision?	Imprécis, c'est un aperçu du processus.

Présentation des données	
Quelles sont les caractéristiques de la présentation des données?	Les noms des axes sont bien identifiés. Cette représentation est dynamique en ce sens que la fenêtre secondaire affiche des informations correspondant à une coordonnée spécifique (x, y) de données qui aide l'utilisateur à analyser le processus d'écriture (Lindgren E. , Sullivan, Lindgren, & Spelman Miller, 2007).

2.3.1.2.5 Représentation temporelle combinée à l'utilisation de sources

La représentation *LS* / *SIG* a été reprise par Leijten et Van Waes (2013) en ajoutant les éléments de source, auxquels le scripteur faisait référence en écrivant, ce qui donne une meilleure vue d'ensemble des activités du scripteur à un moment donné.

La force de cette visualisation est la combinaison de plus d'une représentation. Celle-ci permet au chercheur d'avoir un aperçu plus complet du processus d'écriture.

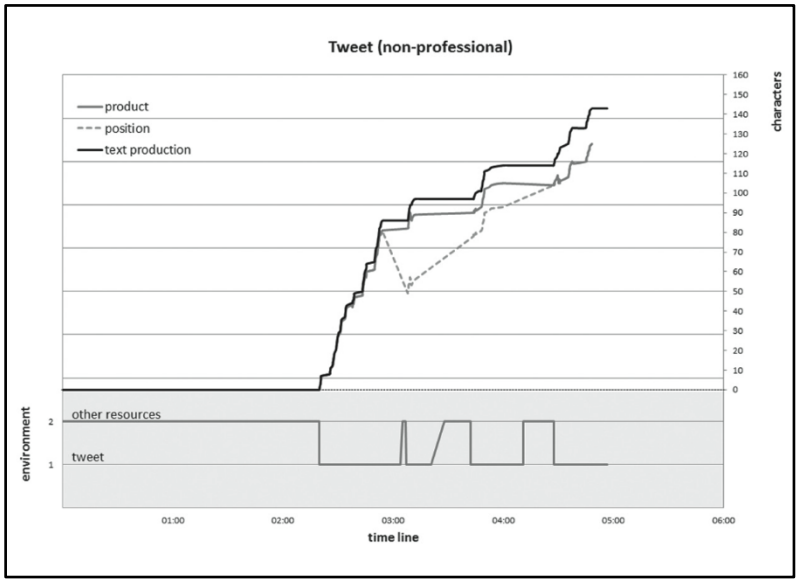

Figure 2.12 Représentation *SIG* combinée aux sources (Leijten & Van Waes, 2013)

Voici les caractéristiques de cette représentation :

Représentation des données	
Quelles sont les variables utilisées ?	Temporalité et spatialité relative.
Comment sont-elles représentées?	L'axe des x représente le temps. L'axe des y représente le nombre total de caractères tapés. La ligne supérieure montre la production de texte. La ligne centrée est la longueur réelle du texte à ce

	moment précis. La ligne inférieure est la position du curseur dans le texte (Leijten & Van Waes, 2013). L'attention du scripteur est représentée par une ligne montrant que la personne regarde les sources ou la fenêtre d'écriture (le tweet dans ce cas).
Quel est le degré de précision?	Imprécis, c'est un aperçu du processus.
Présentation des données	
Quelles sont les caractéristiques de la présentation des données?	Les noms des axes sont bien identifiés. Cette représentation utilise une combinaison de teintes de gris pour séparer les sections.

2.3.1.2.6 Représentation temporelle combinée à l'illustration des pauses

Cette représentation est tirée directement d'une démonstration du logiciel *Inputlog* (Inputlog, 2009) et consiste en la représentation *LS* / *SIG* en incorporant une référence aux pauses que le scripteur a effectuées parallèlement à sa rédaction.

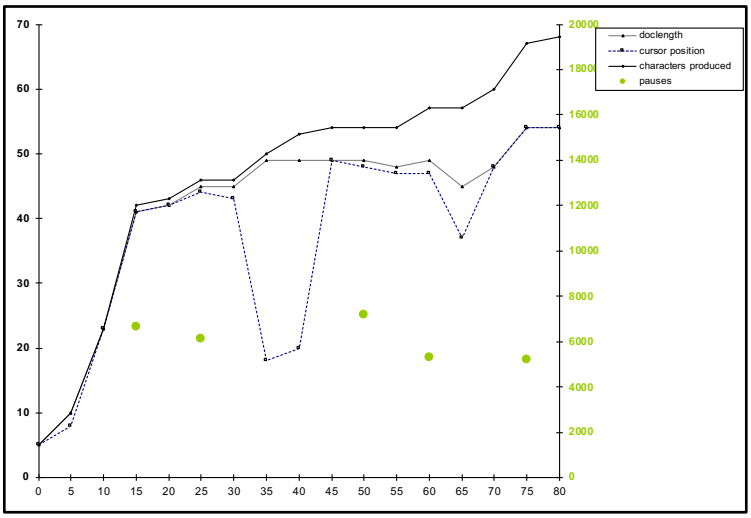

Figure 2.13 Représentation *SIG* avec pauses (Inputlog, 2009)

Voici les caractéristiques de cette représentation :

Représentation des données	
Quelles sont les variables utilisées ?	Temporalité et spatialité relative.
Comment sont-elles représentées?	L'axe des *x* représente le temps. Il y a deux axes des *y*. Celui de gauche représente le nombre total de caractères tapés tandis que

	celui de droite représente la durée de la pause. La ligne supérieure montre la production de texte. La ligne centrée est la longueur réelle du texte à ce moment précis. La ligne inférieure est la position du curseur dans le texte. Les points de couleurs représentent les pauses du scripteur.
Quel est le degré de précision?	Imprécis, c'est un aperçu du processus.
Présentation des données	
Quelles sont les caractéristiques de la présentation des données?	Les noms des lignes sont bien identifiés. Plusieurs formes et couleurs différentes sont utilisées afin de séparer les différentes variables.

2.3.1.3 Représentation par les graphes

Cette représentation permet de bien voir l'aspect dynamique de l'écriture (Caporossi & Leblay, 2011). Elle est axée sur la progression linéaire de la production écrite et représente la chronologie du processus d'écriture (Leblay C. , 2011). L'une des particularités est de bien gérer la transformation du texte grâce aux nœuds représentant des parties de texte (ajoutés, supprimés) reliés ensemble par des arêtes définissant leur relation, soit chronologique ou spatiale (Southavilay, Yacef, Reimann, & Calvo, 2013). Il est possible de voir le contenu des nœuds.

Tel que mentionné par Caporossi et Leblay (2011), cette représentation manque néanmoins l'aspect temporel de la rédaction.

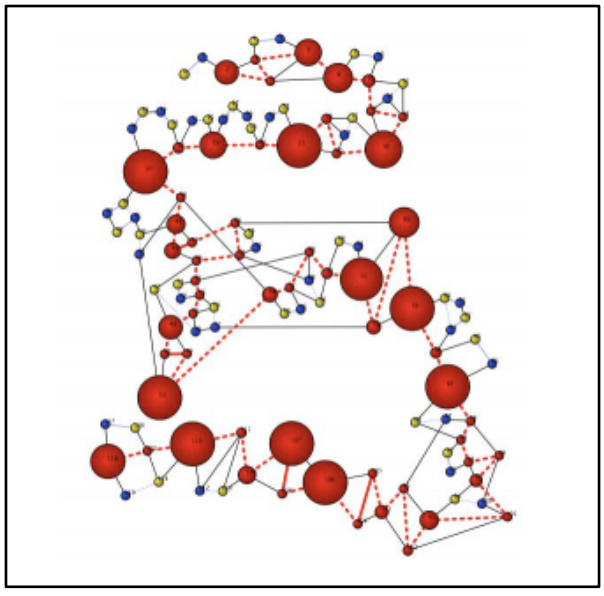

Figure 2.14 *Représentation par les graphes* (Caporossi & Leblay, 2011)

Voici les caractéristiques de la *représentation par les graphes* :

Représentation des données	
Quelles sont les variables utilisées ?	Chronologie et spatialité semi-absolue. Seuls les nœuds représentant les insertions encore présentes dans le texte final sont liés par leur relation spatiale.
Comment sont-elles représentées?	Les nœuds représentent une séquence d'opérations consécutives (dans le temps et dans l'espace) de même nature (insertions/ suppression). Les arêtes représentent une relation spatiale ou temporelle entre les nœuds. La position des nœuds n'est pas importante dans la définition du graphique et est laissée à l'utilisateur (Caporossi & Leblay, 2011 ; 2014 ; 2015).
Quel est le degré de précision?	Intermédiaire : il est possible d'obtenir un aperçu précis en voyant le contenu des nœuds qui contiennent des frappes tandis que l'ensemble du processus est visible par la structure générale du graphe.
Présentation des données	
Quelles sont les caractéristiques de la présentation des données?	L'utilisation des couleurs aide l'utilisateur à connaître la nature des opérations groupées. Le rouge est utilisé pour le texte présent dans le texte final. Le jaune est pour les ajouts qui ont été supprimés et le bleu représente les opérations de suppression. L'utilisation de différents rendus de lignes pour les arêtes en fonction du fait que les opérations sont effectuées consécutivement (ligne noire) ou qu'elles sont successivement chronologiques dans le texte final (ligne pointillée rouge) aide l'utilisateur à mieux comprendre la nature de la relation entre deux nœuds. Le fait qu'il soit possible de déplacer les nœuds et d'en connaître le contenu en cliquant dessus le rend interactif. Permettre à l'utilisateur d'interagir avec une visualisation est un moyen d'augmenter les probabilités de trouver des tendances dans les données (Kirk, 2012). Par contre, puisqu'aucune position n'est attribuée aux nœuds, ceux-ci sont, au départ, placés les uns sur les autres dans le logiciel GenoGraphiX (Caporossi, 2015) et l'utilisateur doit nécessairement prendre le temps de replacer par lui-même les nœuds avant de pouvoir visualiser le graphe dans son ensemble, ce qui peut être long si le graphe comprend beaucoup de nœuds.

Cette représentation est à mi-chemin entre les représentations *SIG* et linéaires. L'aspect dynamique du processus d'écriture est mis en évidence (Caporossi & Leblay, 2011) et la progression linéaire représente la chronologie (Leblay C. , 2011). Une de ses forces est de bien montrer la relation chronologique entre les opérations. Un autre avantage de cette visualisation du processus d'écriture est qu'elle peut gérer la position du texte en mouvement (Southavilay, Yacef, Reimann, & Calvo, 2013).

2.3.1.4 Synthèse des visualisations

Trois grandes familles de visualisations sont utilisées dans l'étude du processus d'écriture : les représentations linéaires, les *SIG* et la *représentation par les graphes*.

Les représentations linéaires sont utiles pour permettre de voir de près les frappes exécutées dans l'ensemble du texte rédigé et présentent pour la plupart la dimension *chronologie*.

Les représentations *SIG* ont l'avantage de bien représenter l'interaction entre la *temporalité* et *spatialité* (Lindgren & Sullivan, 2002). Leur faiblesse est de ne montrer qu'une vue générale du processus. Il est impossible de savoir ce qui a été écrit, le contexte et quelles sont les opérations exactes qui ont été écrites (Caporossi & Leblay, 2011). Puisque la position utilisée est celle qui est enregistrée dans le fichier *log* au moment où la révision a été effectuée, il est impossible de connaître quelle partie du texte est modifiée par cette opération.

La *représentation par les graphes* est utile pour comprendre le lien spatial entre les frappes présentes dans le texte final opérations et la chronologie des opérations (Caporossi & Leblay, 2014). Par contre, il est difficile de saisir la temporalité des opérations puisqu'aucune information ne traite spécifiquement du moment exact où les frappes ont été enfoncées.

2.3.2 Outils d'analyse et mesures utilisées

Les analyses et mesures utilisées se divisent en deux grandes catégories :

- comme attendu, sont retrouvées les mesures générées directement par les logiciels de traitement du processus d'écriture ;
- alternativement, certains chercheurs ont utilisé des techniques statistiques extérieures à ces logiciels afin d'avoir accès à des mesures différentes et orientées sur leur sujet de recherche.

2.3.2.1 Mesures issues de logiciels de traitement du processus d'écriture

La majorité des logiciels de traitement du processus d'écriture permettent d'enregistrer l'écriture d'un texte, de générer le film d'écriture et l'une des visualisations précédemment présentées. Peu d'entre eux génèrent des mesures d'analyses et les représentations produites sont orientées autour de l'analyse des pauses et/ou des révisions.

L'analyse des pauses consiste à repérer l'emplacement, la durée et la fréquence des pauses (Van Waes L. , Leijten, Wengelin, & Lindgren, 2012) et à en présenter les statistiques.

L'analyse des révisions consiste à présenter les insertions et suppressions. Il y a un certain travail manuel supplémentaire à faire pour les chercheurs qui souhaitent affiner les analyses automatiques (Leijten & Van Waes, 2006) et ajouter de nouvelles variables telles que le but de la révision (correction orthographique, grammaticale, du contenu) par exemple (Van Waes & Schellens, Writing Profiles : The Effect of the Writing Mode on Pausing and Revision Patterns of Experienced Writers, 2003). Plus spécifiquement, le logiciel *Inputlog* traite les données pour en extraire le début de la révision, l'imbrication de la révision, la position du curseur durant la révision et la fin de la révision. Ces données sont présentées sous le format d'une liste (Van Waes & Schellens, Writing Profiles : The Effect of the Writing Mode on Pausing and Revision Patterns of Experienced Writers, 2003). Il ne s'agit donc pas d'une analyse à proprement parler, mais plutôt une présentation des données traitées.

De façon générale, peu de détails sur les calculs effectués sont présentés dans la littérature ou dans les manuels d'utilisation de ces logiciels (Leijten & Van Waes, 2013; Baaijen, Galbraith, & Glopper, 2012; Van Waes, Leijten, Wengelin, & Lindgren, 2012).

2.3.2.2 Mesures issues de techniques statistiques

Plusieurs techniques statistiques ont été utilisées dans des ouvrages analysant le processus d'écriture enregistrée. Parmi celle-ci, notons l'*analyse factorielle* (Deane, Using writing process and product features to assess writing quality and explore how those features relate to other literacy tasks, 2014) et ses variantes telles que l'*analyse discriminante* (Van Waes & Schellens, Writing Profiles : The Effect of the Writing Mode on Pausing and Revision Patterns of Experienced Writers, 2003), l'*analyse en composantes principales* (Grabowski, 2008 ; Baaijen, Galbraith, & Glopper, 2012 ; Almond, Deane, Quinlan, Wagner, & Sydorenko, 2012 ; Foucambert & Foucambert, 2014 ; Van Waes & Leijten, 2015) ainsi que l'*analyse par grappes* (*cluster analysis*) (Van Waes & Schellens, Writing Profiles : The Effect of the Writing Mode on Pausing and Revision Patterns of Experienced Writers, 2003). Ces techniques sont souvent utilisées pour synthétiser de grands ensembles de données et identifier les facteurs qui décrivent le mieux possible les tendances dans les variables définies.

Dans certains cas, les auteurs utilisent plutôt des modèles de régression pour traiter leurs données (Miller, Lindgren, & Sullivan, 2008 ; Deane & Zhang, 2015).

Pour l'ensemble de ces méthodes, plusieurs variables sont déterminées puis l'analyse statistique vise à mettre en relation ces variables. Ainsi, l'analyse ne s'effectue pas directement sur les fichiers *log*. Les chercheurs définissent eux-mêmes les variables utilisées en fonction de leurs besoins de recherche. Ces variables sont parfois composées de plusieurs types d'opérations qui sont en fait une combinaison de deux opérations. Par exemple, si on considère la pause comme une opération, une variable définie comme une pause qui est suivie d'une insertion différée (Foucambert & Foucambert, 2014), est une combinaison d'opérations. Parmi les variables utilisées, sont souvent mentionnées la durée de l'écriture, le nombre de sessions d'écriture et la quantité de mots produits par heure pour n'en nommer que quelques-unes. Dans ce cas, plus d'informations quant à la nature des calculs menant aux mesures est disponible (Foucambert & Foucambert, 2014). Dans d'autres cas par contre, certaines des variables utilisées sont qualitatives et requièrent l'intervention humaine (Van Waes & Schellens, Writing Profiles : The Effect of the Writing Mode on Pausing and Revision Patterns of Experienced Writers, 2003). Pour une même variable, on retrouve souvent la moyenne et des mesures de *dispersion* telles que l'*écart-type*, la *valeur minimale et maximale*. Les auteurs Deane et Zhang (2015) soulignent que l'utilisation de telles variables, qui correspondent finalement à des valeurs qui résument l'information se trouvant dans un fichier *log*, indique très peu de détails sur le processus d'écriture. Ils suggèrent qu'une analyse plus nuancée permettrait d'extraire de l'information plus riche et de confirmer des relations plus complexes dans les données.

L'ensemble de ces méthodes ont en commun le fait qu'elles ne traitent pas les données de la même façon. Ceci est entre autres dû aux variables utilisées, qui ne sont pas standardisées et qui dépendent des objectifs de recherche des chercheurs.

2.3.2.3 Synthèse des outils d'analyse et mesures utilisées

Il existe assez peu d'outils quantitatifs élaborés pour l'analyse du processus d'écriture.

Les mesures issues de techniques statistiques sont uniquement utilisées sur un ensemble de fichiers *log*, lesquels sont transformés en un ensemble de variables synthétisant le contenu du fichier *log*. Cette transformation des fichiers *log* en variables intermédiaires, demande un certain pré-traitement des données.

Les seules mesures pouvant être utilisées sur un seul fichier *log* sont celles issues de logiciels de traitement du processus d'écriture. Peu de détails sont donnés quant au calcul de ces mesures. (Leijten & Van Waes, 2013; Baaijen, Galbraith, & Glopper, 2012; Van Waes, Leijten, Wengelin, & Lindgren, 2012).

2.4 Conclusion

En plus d'établir les bases du contexte et de la théorie utilisée dans le cadre de cette thèse, la revue de littérature a permis de faire un survol des techniques d'analyse du processus d'écriture utilisées par les chercheurs. Certaines lacunes des outils d'analyse du processus d'écriture ont pu être observées.

Bien que la visualisation des données reste un outil très utilisé dans l'analyse du processus d'écriture, il n'en existe aucune qui illustre l'ensemble des variables inhérentes au processus d'écriture (temporalité, chronologie, spatialité relative, spatialité absolue).

Les outils quantitatifs d'analyse du processus d'écriture pour leur part sont peu nombreux et ne permettent pas d'automatiser la recherche de tendances complexes dans les données. Bien que l'utilisation de méthodes statistiques soit pertinente, les résultats obtenus et présentés dans la littérature sont dépendants de la question de recherche et des variables intermédiaires établies par les auteurs, ce qui rend les résultats difficiles à comparer puisque non standardisés. De plus, ces méthodes, si elles permettent dans une certaine mesure de déceler des tendances plus complexes, sont limitées par la définition de leurs variables intermédiaires et ne sont pas appropriées lors de l'analyse d'un seul fichier *log*.

Face à un tel constat, il est donc cohérent de proposer de nouveaux outils d'analyse pour faciliter le traitement et l'analyse du processus d'écriture.

Le chapitre suivant présente des outils méthodologiques fréquemment utilisés dans cette thèse.

CHAPITRE 3 OUTILS MÉTHODOLOGIQUES

Le but de cette thèse consiste entre autres à faciliter le traitement et l'analyse du processus d'écriture. Pour répondre à cet objectif, deux nouvelles visualisations et une mesure sont proposées. Au Chapitre 4, la *visualisation progressive* est présentée et au Chapitre 5 il s'agit du modèle *sans pertes*. Finalement le Chapitre 6 présente les mesures de proximité chronologique.

Certains outils vus dans la revue de littérature sont utilisés dans ces trois chapitres. Il s'agit plus spécifiquement de la représentation linéaire *Notation-S*, de la représentation inspirée des *SIG* et de la *représentation par les graphes*. Ces visualisations, qui ont été présentées dans la Section 2.3.1, servent de base ou d'inspiration aux modèles présentés dans les prochains chapitres. Elles sont aussi utilisées pour comparer les propriétés des nouvelles outils développés et à démontrer plus clairement les contributions de cette thèse.

Ce chapitre présentera ces trois visualisations plus en détail. La construction de chacun de ces modèles sera d'abord expliquée. Par la suite, un exemple sera utilisé pour comparer l'interprétation et la découverte de tendances possibles dans chacune des visualisations. Ceci permettra au lecteur de mieux comprendre les limites et avantages de celles-ci.

3.1 Construction des visualisations du processus d'écriture

Dans le but de bien comparer la construction des visualisations, le même exemple sera utilisé pour la *Notation-S*, le *SIG* et la *représentation par les graphes.* Le fichier *log* 3.1 est présenté dans le tableau 3.1. L'*AvantTexte* correspondant à chaque itération du texte est également présenté.

Tableau 3.1 Fichier *log* 3.1

i	*Temps*	*Type*	*Pos*	*C*	*AvantTexte*
1	1974	I	1	t	t
2	5830	I	2	a	ta
3	6030	I	3	s	tas
4	6310	I	4	s	tass
5	8086	I	5	e	tasse
6	8342	S	5	-	tass
7	8518	S	4	-	tas
8	8702	I	4	t	tast
9	8966	S	2	-	tst
10	12118	S	1	-	st
11	13614	I	1	T	Tst
12	13863	I	2	e	Test

La version finale du texte issu de ce fichier *log* est donc :

Test

3.1.1 Construction de la représentation linéaire *Notation-S*

La *Notation-S* a été choisie parmi toutes les représentations linéaires présentées dans la Section 2.3.1.1 en raison de sa simplicité, autant pour sa construction que son interprétation.

Afin de démontrer le processus de construction de la *Notation-S*, voici un court exemple. Pour consulter les symboles utilisés pour illustrer les opérations de révision (Kollberg, 1996), ceux-ci ont été préalablement présentés dans la Section 2.3.1.1.1.

La construction de la *Notation-S* est présentée ici en suivant la nomenclature et les détails donnés par Horenbeeck et al. (2012). À cette fin, la valeur *position* issue du fichier *log* est utilisée, ainsi que trois autres concepts : la *production pré-contextuelle*, la valeur *comptePosition* et la *révision* qui sont définies ci-après.

Production pré-contextuelle : Il s'agit de la production de texte, c'est-à-dire de l'insertion de caractères, à la fin du document, au *fil de la plume*. La fin du document est soit après la dernière position du document ou à une position telle que les seuls caractères suivants sont des espaces. La production pré-contextuelle n'est marquée d'aucune interruption.

ComptePosition : Valeur accordée à chaque caractère dans le texte. La valeur 1 est donnée pour tous les caractères non supprimés du texte. Un caractère supprimé se fait attribuer une valeur de 0. Les symboles d'interruptions ne possèdent pas de valeur de position.

Révision : Toute action qui suit une interruption. Les *suppressions immédiates*, les *suppressions différées* et les *insertions différées* qui ne sont pas dans la production pré-contextuelle sont considérées comme des *révisions*.

a) Les cinq premiers caractères sont écrits en *production pré-contextuelle* :

tasse

Ordre des caractères dans la *Notation-S*	t	a	s	s	e
ComptePosition	1	2	3	4	5

b) Interruption 1 : Suppression des caractères en position 5 et 4 :

$$\boldsymbol{tas[se]^1|_1}$$

Ordre des caractères dans la *Notation-S*	t	a	s	s	e
ComptePosition	1	2	3	0	0

c) Insertion de la lettre *t* en *production pré-contextuelle* (donc pas d'interruption) en position 4 :

$$\boldsymbol{tas[se]^1|_1t}$$

Ordre des caractères dans la *Notation-S*	t	a	s	s	e	t
ComptePosition	1	2	3	0	0	4

d) Interruption 2 : Suppression des caractères en position 2 et 1 :

$$\boldsymbol{[ta]^2s[se]^1|_1t|_2}$$

Ordre des caractères dans la *Notation-S*	t	a	s	s	e	t
ComptePosition	0	0	1	0	0	2

e) Interruption 3 : Insertion des deux caractères *Te* sans repositionner le curseur suite à l'interruption 2, soit en position 1 et 2.

$$\boldsymbol{[ta]^2|_3\{Te\}^3s[se]^1|_1t|_2}$$

Ordre des caractères dans la *Notation-S*	t	a	T	e	s	s	e	t
ComptePosition	0	0	1	2	3	0	0	4

3.1.2 Construction de la représentation inspirée des *SIG*

Les *SIG* sont composés d'un graphique en deux dimensions (Lindgren & Sullivan, 2002). L'axe des abscisses représente généralement le temps d'une frappe i mais pourrait aussi représenter la chronologie de la frappe i. L'axe des ordonnées représente le nombre total de caractères tapés. La ligne du haut, illustrée ici en vert, montre la progression du nombre total de caractères tapés. La ligne centrée, en orange, est la longueur de l'$AvantTexte(i)$, c'est-à-dire au moment où la frappe i

a été exécutée. Enfin, la ligne inférieure est la valeur $Pos(i)$, c'est-à-dire la position du curseur dans le texte lorsque la frappe i est exécutée.

Le tableau 3.2 reprend le détail du fichier *log* 3.1 en y adjoignant les valeurs des lignes orange et verte, qui sont dans les colonnes à droite.

Tableau 3.2 Valeurs *SIG* du fichier *log* 3.1

i	*Temps*	*Type*	*Pos*	*C*
1	1974	I	1	t
2	5830	I	2	a
3	6030	I	3	s
4	6310	I	4	s
5	8086	I	5	e
6	8342	S	5	-
7	8518	S	4	-
8	8702	I	4	t
9	8966	S	2	-
10	12118	S	1	-
11	13614	I	1	T
12	13863	I	2	e

Nombre de caractères dans $AvantTexte(i)$	*Nombre cumulatif de frappes de type I*
1	1
2	2
3	3
4	4
5	5
4	5
3	5
4	6
3	6
2	6
3	7
4	8

La figure 3.1 présente le *SIG* correspondant au fichier *log* 3.1.

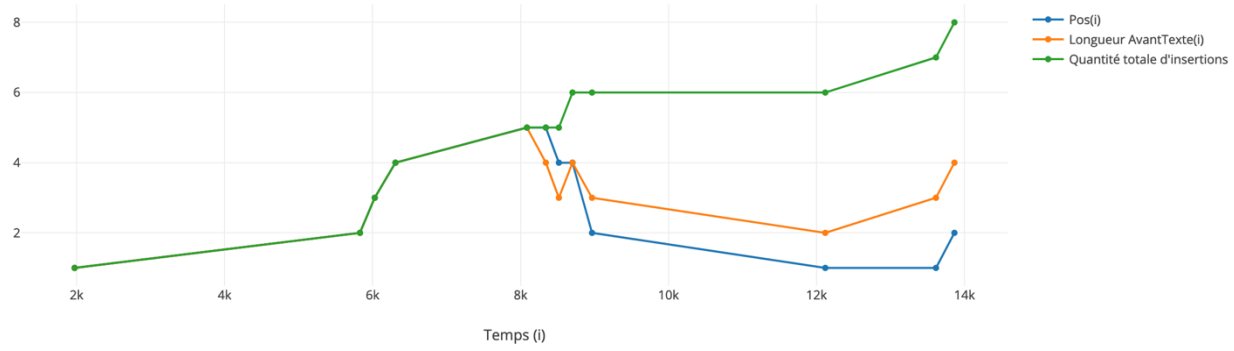

Figure 3.1 *SIG* correspondant à l'exemple 3.1

3.1.3 Construction de la *représentation par les graphes*

Chaque nœud regroupe les séquences de caractères de même nature si celles-ci sont également contiguës dans le temps (Caporossi & Leblay, 2011). Ces séquences contiennent soit des *insertions* ou des *suppressions*. Le graphe et le contenu de ses nœuds évoluent à mesure que les lignes du fichier *log* sont traitées pour tenir compte des interruptions. Deux nœuds *temporellement* consécutifs sont reliés par une arête et deux nœuds *spatialement* consécutifs sont également reliés par une arête, seulement si ces derniers sont présents dans le texte final.

Afin d'expliciter le processus de transformation du graphe, les grandes étapes du processus d'écriture du fichier *log* 3.1 décrites dans la Section 3.1.1 seront reprises.

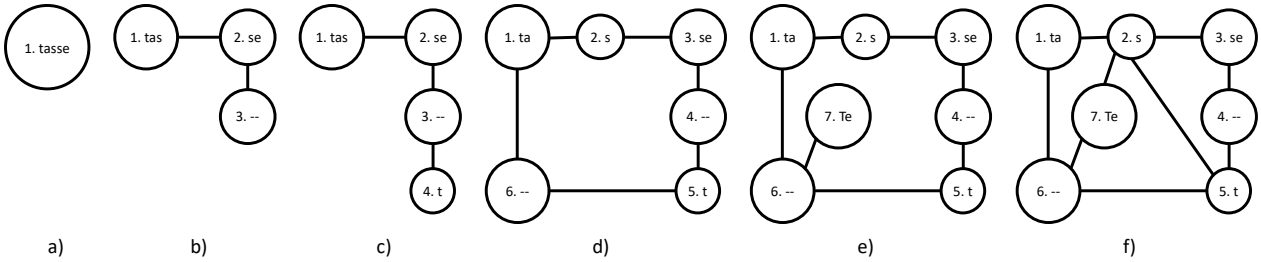

Figure 3.2 Construction de la *représentation par les graphes* du fichier *log* 3.1

Voici un rappel sommaire des étapes :

a) Les cinq premiers caractères sont écrits en *production pré-contextuelle.*

b) Interruption 1 : Suppression des caractères en position 5 et 4. Puisqu'il y a une interruption dans la *séquentialité* des opérations, les caractères en position 5 et 4 sont mis à part dans un deuxième nœud et un troisième nœud représente ces deux caractères supprimés.

c) Insertion de la lettre *t* en *production pré-contextuelle* soit à la fin du texte.

d) Interruption 2 : Suppression des caractères en position 2 et 1.

e) Interruption 3 : Insertion de deux caractères *Te* sans repositionner le curseur suite à l'interruption 2, soit en position 1 et 2.

f) Des arêtes sont créées pour lier les nœuds représentant le texte final.

Afin de différencier la nature de la relation entre deux nœuds et le type de lien entre ceux-ci, des codes de couleurs sont utilisés. Les tableaux ci-après détaillent les conventions des couleurs des cellules et les conventions de représentation des arêtes tels que décrits par Caporossi et Leblay (2011; 2014).

Tableau 3.3 Convention de couleur des nœuds de la *représentation par les graphes*

Type de séquence	**Couleur**
Insertion présente dans le texte final	Rouge
Insertion supprimée	Jaune
Suppression	Bleu

Tableau 3.4 Convention de couleur des arêtes de la *représentation par les graphes*

Type d'arête	**Rouge**	**Noir**	**Bleu**
Continu		Représente deux nœuds qui se suivent chronologiquement, mais pas nécessairement spatialement.	Représente le lien entre l'acte de supprimer et le texte qu'il supprime, lorsque la suppression se produit successivement chronologiquement à l'insertion. Il s'agit d'une *suppression immédiate*.
Pointillé	Lien entre des nœuds qui se suivent dans le texte final, mais qui ne se suivent pas chronologiquement.		Représente le lien entre l'acte de supprimer et le texte qu'il supprime, lorsque la suppression ne se produit pas successivement chronologiquement. Il s'agit d'une *suppression différée*.

Considérant ces conventions, le graphe final préalablement présenté ressemble à ceci :

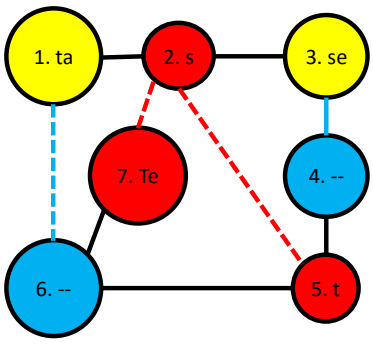

Figure 3.3 Version colorée de la *représentation par les graphes* du fichier *log* 3.1

3.2 Interprétation des visualisations du processus d'écriture

Tout comme dans le cas de la construction des visualisations, le même fichier *log* sera utilisé pour démontrer comment interpréter ces visualisations. Afin d'illustrer des motifs complexes, un exemple de plus grande taille sera présenté. Cet exemple, le fichier *log* 3.2, est de taille modeste si on le compare aux fichiers généralement étudiés qui peuvent posséder jusqu'à 2000 frappes pour seulement 15 minutes de rédaction (Wengelin A. , 2014).

L'écriture de cet exemple se fait en 5 grandes étapes.

1. Frappes 1 à 18 :

   le scripteur écrit d'abord *Voici une* puis supprime le dernier caractère soit le *e* de *une*. Il rédige ensuite *exemple* de manière à ce que le texte final à ce moment précis soit ***Voici un exemple***.

2. Frappes 19 à 33 :

   le scripteur revient au début de son texte pour supprimer le mot *Voici* et le remplace par *Cee*, supprime le dernier e et insère immédiatement après *ci est* tel que le texte à ce moment-là est ***Ceci est un exemple***.

3. Frappes 34 à 44 :

   le scripteur change son curseur d'endroit pour se positionner devant l'espace précédant le mot *exemple* et y insère *tout petit*. Le texte à ce moment est ***Ceci est un tout petit exemple***.

4. Frappes 45 à 49 :

   le scripteur supprime ensuite le mot *tout* ainsi que l'espace précédant celui-ci de manière à ce que le texte devienne ***Ceci est un petit exemple***.

5. Frappe 50

   le scripteur insère un point à la toute fin du texte. C'est ainsi que le texte prend sa forme finale soit ***Ceci est un petit exemple.***.

3.2.1 Interprétation de la *Notation-S*

La *Notation-S* a été utilisée par Kollberg (1996) pour établir une définition formelle des concepts de révision élémentaire et d'épisodes de révision. Une révision élémentaire consiste soit en une révision sans bouger le curseur, ou encore en une insertion ou en une suppression différée. Un épisode de révision consiste en une séquence de révisions élémentaires (Kollberg, 1996).

Les révisions immédiates sont effectuées à une distance nulle, c'est-à-dire que le scripteur effectue la révision exactement à la position de l'opération précédente, sans bouger son curseur. Les révisions différées se produisent lorsque le scripteur bouge le curseur pour réviser dans une autre portion du texte, soit à une distance qui n'est pas égale à zéro (Kollberg, 1997).

La *Notation-S* du fichier *log* 3.2 est présenté ci-après.

$$[Voici]^{2}\ |_{3}\{Ce[e]^{4}|_{5}\{ci\ est\}^{5}|_{6}\}^{3}|_{4}un[[e]^{1}|_{1}\{tout]^{7}\ \ petit\}^{6}|_{7}\ exemple\ |_{2}.$$

La suite de symboles $]^{1}|_{1}$ nous indique que cette révision est immédiate. Le point d'inscription de l'interruption ($|_{1}$) et l'identification de la suppression portant le même numéro séquentiel ($]^{1}$) se suivent.

Toutes les autres révisions sont différées. Cet exemple est très dense, ce qui est dû à un grand nombre de révisions effectuées sur une courte portion de texte. Pour avoir le détail des autres révisions, il faudrait analyser une par une les interruptions. Dans un tel cas, il n'est pas évident de repérer visuellement des tendances.

3.2.2 Interprétation du *SIG*

Tel que décrit dans la Section 3.1.2, l'axe des abscisses du *SIG* représente généralement la temporalité des frappes d'un fichier *log*. Par contre, certains motifs sont plus facilement repérables lorsque l'axe des abscisses représente la chronologie des frappes. Pour cette raison, le *SIG* du fichier *log* 3.2, ayant la chronologie comme variable indépendante sera d'abord présenté à la figure 3.4. L'interprétation du *SIG* du fichier *log* 3.2, ayant la temporalité comme variable indépendante sera présenté par la suite.

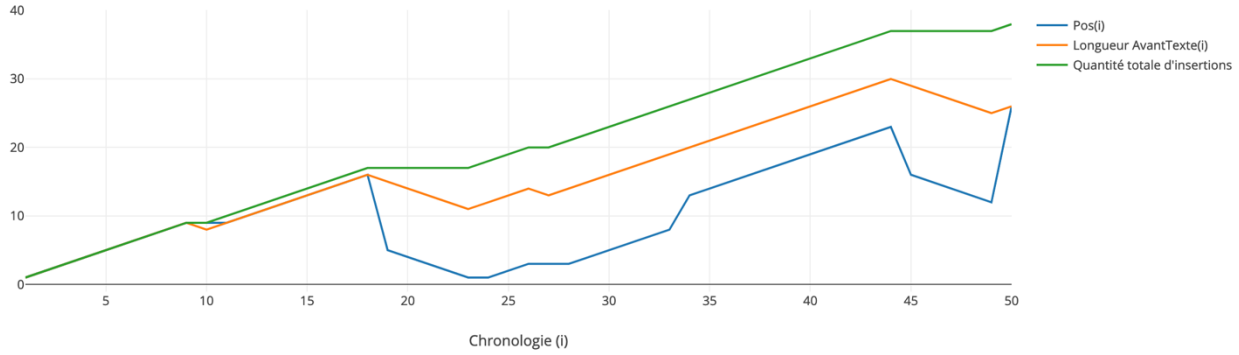

Figure 3.4 *SIG* du fichier *log* 3.2 avec la chronologie comme variable indépendante

Le premier motif qu'il est possible de repérer dans le *SIG* concerne les épisodes d'insertion et de suppression.

Lorsque le scripteur insère des caractères, la courbe de la longueur de l'$AvantTexte(i)$ croît. Lorsque le scripteur supprime des caractères, au contraire, la courbe de la longueur de l'$AvantTexte(i)$ décroît. Ce motif peut aussi être identifié en observant la courbe de la quantité totale d'insertions, qui est en vert. Lorsque le scripteur insère des caractères, la courbe de la quantité totale d'insertions croît. Lorsque le scripteur supprime des caractères, la courbe de la quantité totale d'insertions est parallèle à l'axe des abscisses.

Le second motif qu'il est possible de repérer dans le *SIG* concerne la position du curseur. Par définition, la valeur de position d'une frappe i de type insertion qui est exécutée dans le pré-contextuel est $Pos(i) = |AvantTexte(i)|$. Conséquemment, lorsqu'une frappe de type insertion est exécutée dans le pré-contextuel, la courbe orange et la courbe bleue sont exactement les mêmes. Par définition, la valeur de position d'une frappe i de type suppression qui est exécutée dans le pré-contextuel est $Pos(i) = |AvantTexte(i)| + 1$. Conséquemment, lorsqu'une frappe de type suppression est exécutée dans le pré-contextuel, la courbe bleue est d'une unité au-dessus de la courbe orange et les deux courbes sont parallèles. Lorsque la courbe orange et la courbe bleue sont correspondent pas à ces attributs, ceci signifie que la frappe est exécutée dans le contextuel.

Le troisième motif qu'il est possible de repérer dans le *SIG* concerne le repositionnement du curseur. Lorsque la variation de $Pos(i)$ est plus grande que la simple variation due à l'accumulation d'insertions ou de suppressions, on peut observer un repositionnement du curseur.

Dans la figure 3.5, les suites de suppressions sont identifiées en gris et les suites d'insertions sont laissées en blanc. Le repositionnement du curseur, pour sa part, est identifié par des lignes verticales noires.

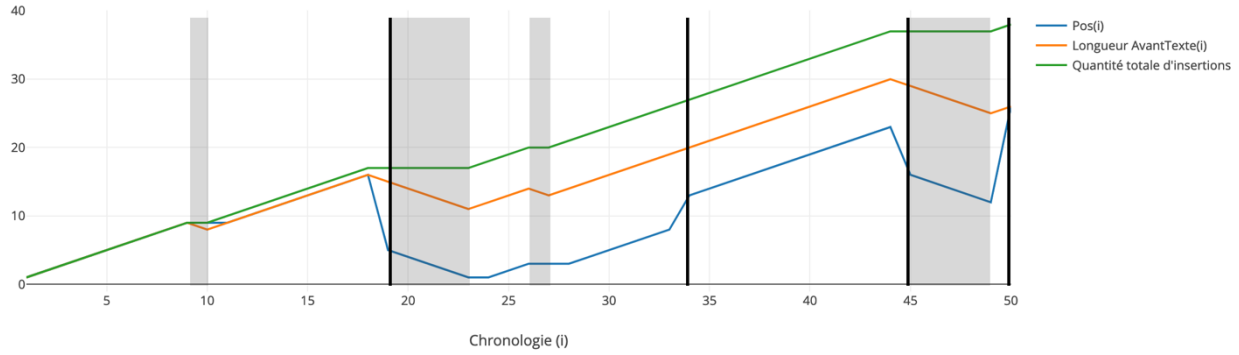

Figure 3.5 *SIG* du fichier *log* 3.2 avec insertions et repositionnements du curseur

On remarque qu'il y a cinq suites d'insertions et quatre suites de suppressions. On remarque aussi que le curseur a été repositionné à quatre reprises.

Soulignons qu'une suite d'insertions ne correspond pas nécessairement à un épisode d'insertion. Les frappes 28 à 44 sont toutes des insertions. Par contre, le curseur est repositionné à la frappe 34. Il s'agit donc de deux épisodes d'insertions distincts, le premier étant constitué des frappes 28 à 33 et le second, des frappes 34 à 44.

Le scripteur effectue les frappes 1 à 18 ainsi que la frappe 50 dans le pré-contextuel. Toutes les autres frappes sont effectuées dans le contextuel.

Lorsque l'axe des abscisses du *SIG* représente le temps, les motifs sont parfois plus difficiles à identifier. Puisque les courbes ne croissent ou décroissent pas à un rythme régulier, un faible repositionnement du curseur est moins évident à trouver.

La figure 3.6 présente le *SIG* du fichier *log* 3.2 lorsque l'axe des abscisses représente le temps. L'avantage de cette version du *SIG* est de bien mettre en évidence les pauses (Lindgren & Sullivan, 2002).

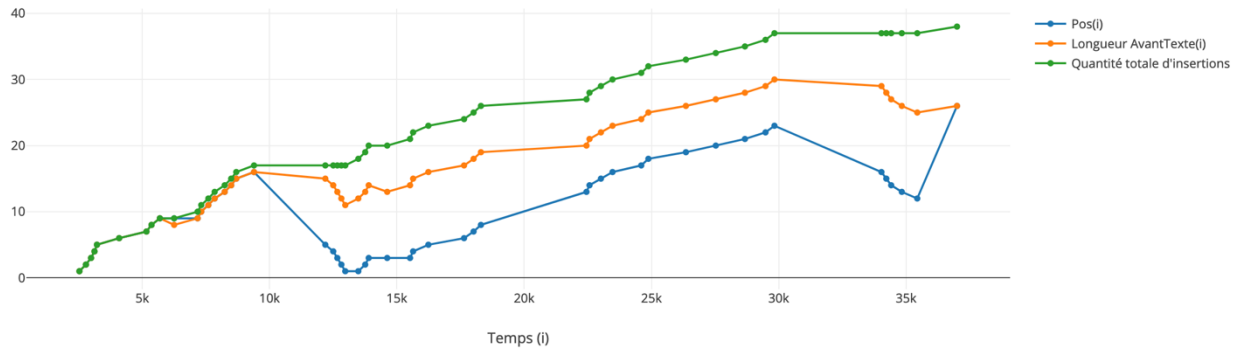

Figure 3.6 *SIG* du fichier *log* 3.2 lorsque l'axe des abscisses représente le temps

Les visualisations de type *SIG* comportent deux défauts principaux. Tout d'abord, il est impossible de savoir avec exactitude le contenu d'un point dans le graphique. Ensuite lorsqu'il y a un retour dans le texte, il est souvent difficile de connaître quelle partie du texte a été modifiée par cette opération puisque la position utilisée est celle qui est enregistrée dans le fichier *log* au moment où la révision a été effectuée.

3.2.3 Interprétation de la *représentation par les graphes*

La figure 3.7 illustre la représentation par les graphes du fichier *log* 3.2.

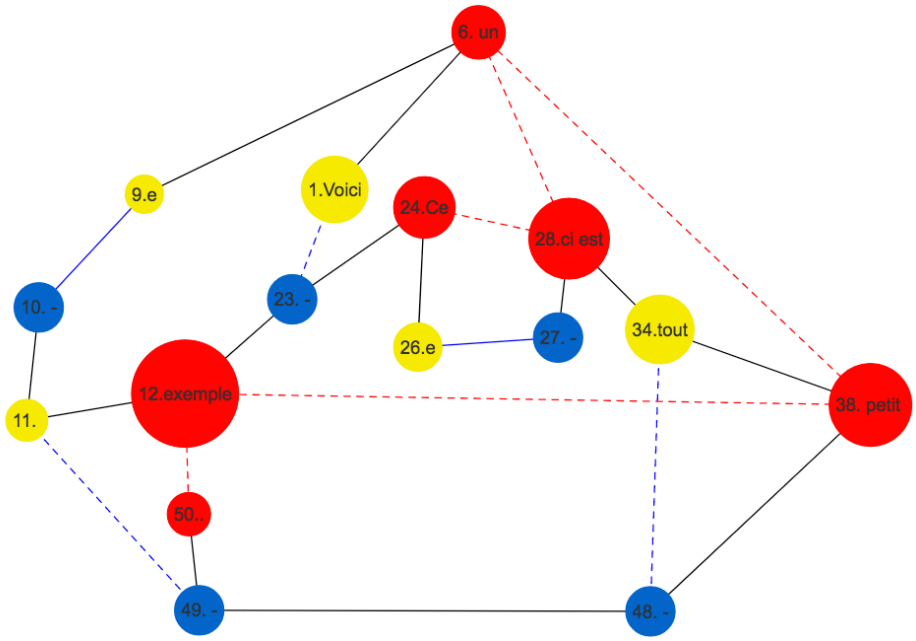

Figure 3.7 *Représentation par les graphes* du fichier *log* 3.2

Plusieurs motifs représentant des opérations classiques du processus d'écriture sont identifiables dans la *représentation par les graphes*.

Un atout de la *représentation par les graphes* est d'illustrer avec précision si une suppression est immédiate ou différée. Cette information est reconnaissable par le trait reliant le nœud insertion et le nœud suppression. Un trait solide indique que la suppression est immédiate tandis qu'un trait pointillé indique que la suppression est différée (Caporossi & Leblay, 2011). Dans la figure 3.7 on remarque qu'il y a deux suppressions immédiates (nœuds 10 et 27) et trois suppressions différées (nœuds 27, 48 et 49).

Il est aussi possible d'identifier quelles sont les insertions qui sont toujours présentes dans le texte. Les nœuds en rouge sont reliés entre eux par des arêtes rouges. Le sous-graphe composé uniquement de cet ensemble de nœuds et d'arêtes forme un chemin. L'ordre des nœuds dans ce chemin correspond à l'ordre des caractères qu'ils représentent dans le texte final (Caporossi & Leblay, 2011). Par contre, à moins de lire le contenu des nœuds, il n'est pas possible de déterminer dans quel sens lire les nœuds sur ce chemin.

Un autre désavantage de la *représentation par les graphes* est qu'il est impossible de retracer les différentes versions du texte à partir du graphe final. Le graphe résultant n'est pas unique à un processus spécifique.

L'exemple suivant démontre clairement cette situation. Les trois tableaux ci-dessous présentent des fichiers *log* légèrement différents. L'$AvantTexte$ correspondant est également présenté.

Tableau 3.5 Fichier *log* 3.3

i	*Type*	*Pos*	*C*
1	I	1	*a*
2	I	1	*c*
3	I	3	*t*
4	S	1	-

AvantTexte
a
ca
cat
at

Tableau 3.6 Fichier *log* 3.4

i	*Type*	*Pos*	*C*
1	I	1	*a*
2	I	2	*c*
3	I	3	*t*
4	S	2	-

AvantTexte
a
ac
act
at

Tableau 3.7 Fichier *log* 3.5

i	*Type*	*Pos*	*C*
1	I	1	*a*
2	I	2	*c*
3	I	1	*t*
4	S	3	-

AvantTexte
a
ac
tac
ta

Ces trois processus contiennent l'insertion de trois caractères (*a, c, t*) et la suppression du caractère *c* en tant que dernière opération. La dernière version du texte est la même pour les fichiers *log* 3.3 et 3.4 mais le processus et $AvantTexte(i)$ ne sont pas les mêmes. La dernière version du texte du fichier *log* 3.5 est par contre différente des deux premiers. Malgré ces différences, la *représentation par les graphes* de ces trois fichiers *log* est exactement la même.

La figure 3.8 représente les fichiers *log* 3.3, 3.4 et 3.5 sous la *représentation par les graphes*.

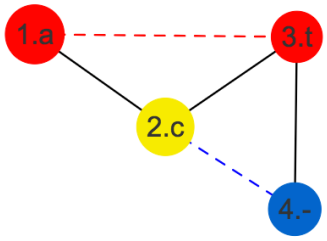

Figure 3.8 *Représentation par les graphes* des fichiers *log* 3.3, 3.4 et 3.5

La dimension spatiale est donc présente seulement partiellement puisque seuls les nœuds représentant des frappes qui sont spatialement contiguës dans le texte final seront liées ensemble dans la *représentation par les graphes*.

3.3 Conclusion

Ce chapitre a présenté la construction et l'interprétation de la représentation linéaire *Notation-S*, de la représentation inspirée des *SIG* et de la *représentation par les graphes*. Ces visualisations et outils d'analyse du processus d'écriture sont utilisés dans les prochains chapitres puisqu'ils servent d'inspiration aux modèles présentés. Des comparaisons entre ces visualisations et les nouveaux modèles seront également effectuées. Le but de cette thèse est de modéliser le fichier log et permettre un repérage plus facile des motifs complexes dans les données. La présentation en détails de ces trois visualisations du processus d'écriture nous permettra, dans les prochains chapitres, de démontrer plus clairement les contributions de cette thèse.

Le chapitre suivant présente la première visualisation élaborée dans le cadre de cette thèse, la *visualisation progressive*.

CHAPITRE 4 *VISUALISATION PROGRESSIVE*

L'un des outils privilégiés pour l'étude du processus de l'écriture est la visualisation de données. Lorsqu'utilisées dans un contexte d'analyse de données, les visualisations aident à identifier les tendances, structures et relations entre les données. Il existe plusieurs types et versions de visualisations du processus d'écriture. Celles-ci ont été présentées dans la Section 2.3.1 et sont détaillées davantage dans le Chapitre 3. Tel que mentionné, ces visualisations peuvent être regroupées en trois grandes familles : les représentations linéaires, les représentations inspirées des *SIG* et la *représentation par les graphes*. Chacune de celles-ci s'oriente autour d'une ou de deux dimensions suivantes : la *chronologie*, la *temporalité* et/ou la *spatialité,* qu'elle soit *relative* ou *absolue*.

Aucune de ces visualisations ne comprend l'ensemble des dimensions. Conséquemment, les chercheurs ont souvent recours à une utilisation conjointe de deux de celles-ci (Lindgren & Sullivan, 2002).

L'objectif de la *visualisation progressive*, qui est présentée dans ce chapitre, est d'être une solution intégrative qui permette de visualiser le processus d'écriture en considérant l'ensemble des dimensions. La solution proposée jumelle les avantages des méthodes *SIG* avec la structure de la *représentation par les graphes* tout en considérant l'ordre des opérations tel qu'il est présenté par les représentations linéaires. De plus, puisque le but est de représenter le processus d'écriture de façon à mettre en relief l'aspect *temporel*, cette représentation est dynamique : il est possible d'interagir avec elle. Cet aspect permet également au chercheur d'alterner entre une vue d'ensemble (*macro*) et une vue rapprochée du processus (*micro*), ce qui permet de contextualiser des frappes spécifiques dans le flux global du processus d'écriture.

Ce chapitre présentera d'abord la construction de la *visualisation progressive*. Il y aura ensuite la présentation d'une version simplifiée et plus compacte. Finalement, les différents avantages et inconvénients de la *visualisation progressive* seront présentés et comparés à ceux des trois visualisations qui lui sont parentes.

Une version préliminaire de la *visualisation progressive* a été présentée dans trois publications (Bécotte-Boutin, Caporossi, & Hertz, 2015; Bécotte-Boutin, Caporossi, Hertz, & Leblay, 2016; Bécotte-Boutin, Caporossi, Leblay, & Hertz, 2019). Dans ces publications, seule une description

sommaire de la visualisation et une figure exemple ont été présentés. L'ensemble des exemples présentés dans cette thèse sont originaux et n'ont pas été publiés. La Section 4.3.5 de cette thèse est une adaptation du tableau 4.3.B publié dans (Bécotte-Boutin, Caporossi, Leblay, & Hertz, 2019). La *visualisation progressive* présentée dans cette thèse a été améliorée, détaillée et formalisée au moyen d'algorithmes, théorèmes et preuves. La majorité du contenu présenté dans ce chapitre est donc unique à cette thèse.

4.1 Description du modèle de la *visualisation progressive*

La *visualisation progressive* est principalement inspirée de la *représentation par les graphes*. Contrairement à cette dernière, où chaque nœud est composé d'un ensemble d'insertions consécutives, chaque caractère individuel est d'abord considéré comme un nœud dans la *visualisation progressive*. Une autre distinction est le fait que le graphe de la *visualisation progressive* est orienté alors que celui de la *représentation par les graphes* n'est pas orienté.

Dans un souci de cohérence, la nomenclature des arcs et des couleurs de nœuds est calquée sur celle utilisée dans la *représentation par les graphes*.

Les nœuds du graphe sont positionnés selon un ensemble des coordonnées (x, y). La variable x représente la *spatialité absolue*. La variable y représente la *temporalité*, c'est-à-dire le moment exact auquel l'opération d'écriture a été effectuée. Le graphe peut être inclus dans un graphique à deux axes mais ces axes ne sont pas nécessaires puisque la position des nœuds est suffisante à la lecture de la visualisation.

Contrairement aux *SIG*, cette visualisation doit être lue de haut en bas pour imiter la lecture occidentale d'un texte. Une autre différence est la position illustrée. Dans les *SIG*, la position de la frappe est relative et correspond exactement à la valeur de position issue du fichier *log*. Dans le cas de la *visualisation progressive*, l'ordre *spatial* est le même que l'ordre des caractères de la *Notation-S* tel qu'explicité par Van Horenbeeck et al. (2012). Il ne peut pas y avoir plus d'une frappe insertion qui possède la même valeur de position. Dans le cas de la *visualisation progressive*, les actions de suppressions sont également illustrées, ce qui n'est pas le cas pour la *Notation-S*. Afin d'attribuer une coordonnée *x* aux nœuds de type *S*, la règle suivante est suivie :

> si une insertion *i* a été par la suite supprimée, l'action de suppression *j* aura la même valeur de coordonnée *x* que *i,* mais *i* et *j* auront chacun leur valeur temporelle *y* respective.

Ainsi, même si une portion du texte est supprimée, elle est conservée sur la visualisation.

Puisque la *visualisation progressive* est un outil élaboré pour aider les chercheurs qui étudient le processus d'écriture à trouver des tendances et structures dans les données d'écriture, les aspects de représentation et de présentation des données doivent être considérés afin de la rendre la plus propice à l'analyse humaine.

Tel que mentionné à la Section 2.2.2, le fait d'interagir avec les données permet au cerveau humain de mieux détecter les tendances et de comprendre les relations entre les variables (Kirk, 2012). Pour cette raison, l'aspect dynamique d'une représentation visuelle est intégré dans la *visualisation progressive*. Dans ce modèle, dans un premier temps, cet aspect provient de la propriété à s'approcher ou s'éloigner des composants. Cette fonction permet à l'utilisateur d'explorer les détails du processus tout en comprenant sa dynamique dans son ensemble. Au lieu de devoir recourir à deux représentations différentes pour avoir accès à cette information, le chercheur peut n'en utiliser qu'une, ce qui contribue aussi à accélérer son analyse.

4.1.1 Transformation du fichier *log* en graphe

Le graphe *progressif* représentant le processus d'écriture est généré à partir du fichier *log*.

L'algorithme 2 décrit la construction de ce graphe.

- Soit λ le nombre de lignes dans le fichier *log* et $TexteFinal = AvantTexte(\lambda)$.
- Chaque frappe i est un nœud dans le graphe. Les valeurs $Type(i)$, $Temps(i)$ et $C(i)$ sont automatiquement attitrées à chaque nœud.
- Pour chaque nœud x tel que $Type(x) = I$, dénotons $p_i(x)$ la position de $C(x)$ dans $AvantTexte(i)$ pour tous les i tel que $x \leq i \leq \lambda$. Si $C(x)$ a été supprimé, alors $p_i(x) = -1$.
- Pour chaque nœud x tel que $Type(x) = S$ on a que $p_i(x) = -1$ pour tous les i tel que $x \leq i \leq \lambda$.
- Soit $coordonnéeX_i(x)$ un attribut de position de chaque nœud $x \leq i$ qui reproduit l'ordre des opérations de la *Notation-S* dans $AvantTexte(i)$. Les nœuds de type *S* possèdent la même valeur de $coordonnéeX_i$ que l'insertion qu'ils suppriment.

- Un arc est créé entre chaque paire de nœuds consécutifs $(x, x+1)$.
- Si un nœud x de type I a été supprimée par un nœud y de type S, l'arc (x, y) est créé.

La transformation s'effectue en deux étapes. D'abord le graphe est créé. Tous les nœuds de celui-ci contiennent au départ les mêmes attributs qui sont associés à l'opération d'écriture dans le fichier *log* soit le type d'opération, son temps, la frappe associée et sa position.

Pour terminer la représentation, chaque nœud i du graphe est positionné selon ses coordonnées (x, y) où $x = coordonnéeX_{\lambda}(i)$ et $y = Temps(i)$. Les axes peuvent ou non être générés en complément.

Afin d'illustrer la transformation du *log* en graphe progressif, l'exemple 3.1 ainsi que ses cinq grandes étapes de rédaction seront reprises.

En haut de chaque nœud x se retrouvent entre parenthèses les valeurs $(p_i(x), coordonnéeX_i(x))$ où i représente le nœud le plus grand dans le graphe en construction.

a) Les cinq premiers caractères sont écrits en *production pré-contextuelle.*

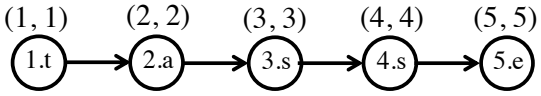

Figure 4.1 Étape a) de la construction de la *visualisation progressive*

b) Interruption 1 : suppression des caractères en position 5 et 4.

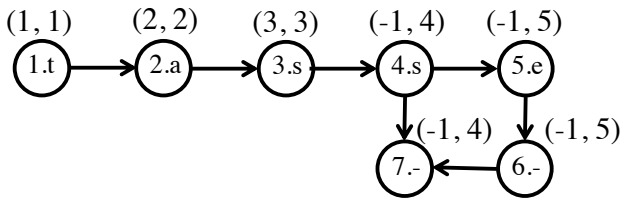

Figure 4.2 Étape b) de la construction de la *visualisation progressive*

c) Insertion de la lettre *t* à la fin du texte.

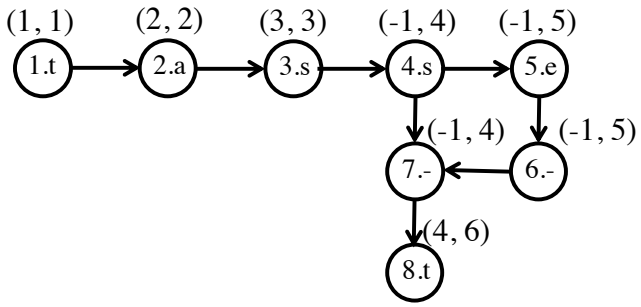

Figure 4.3 Étape c) de la construction de la *visualisation progressive*

d) Interruption 2 : suppression des caractères en position 2 et 1 :

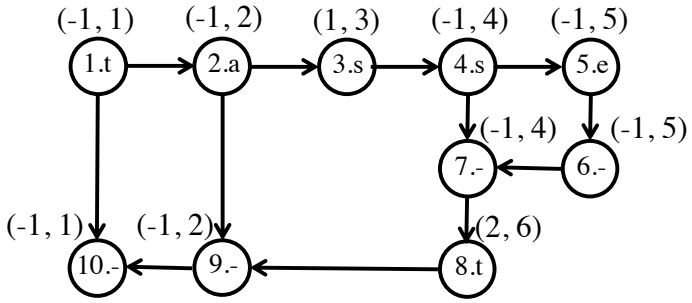

Figure 4.4 Étape d) de la construction de la *visualisation progressive*

e) Interruption 3 : Insertion de deux caractères *Te* sans lever le curseur suite à l'interruption 2, soit en position 1 et 2.

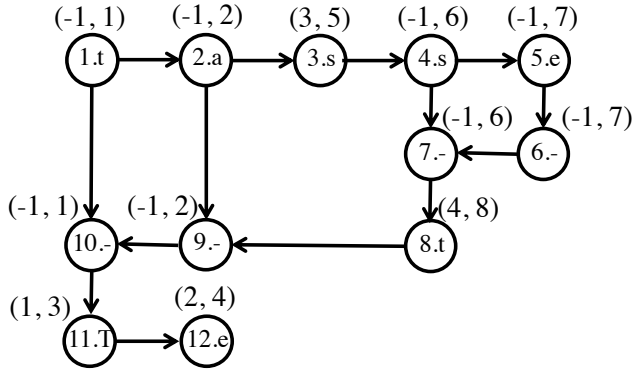

Figure 4.5 Étape e) de la construction de la *visualisation progressive*

f) Création des arcs liant les nœuds des frappes encore présentes dans le texte final, dans l'ordre du texte final.

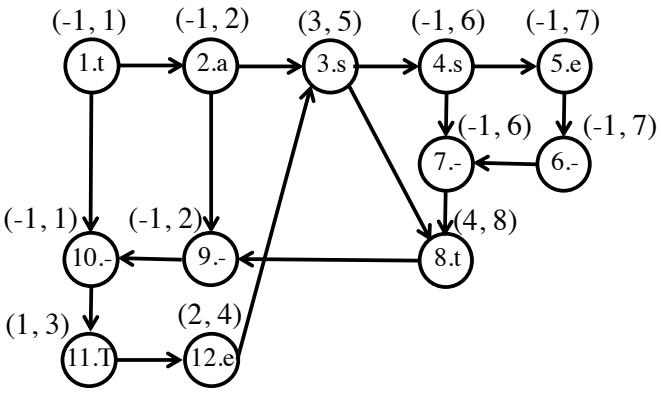

Figure 4.6 Étape f) de la construction de la *visualisation progressive*

Une fois le graphe construit, il suffit de positionner les nœuds selon leurs coordonnées (x, y) et d'appliquer les conventions de couleurs.

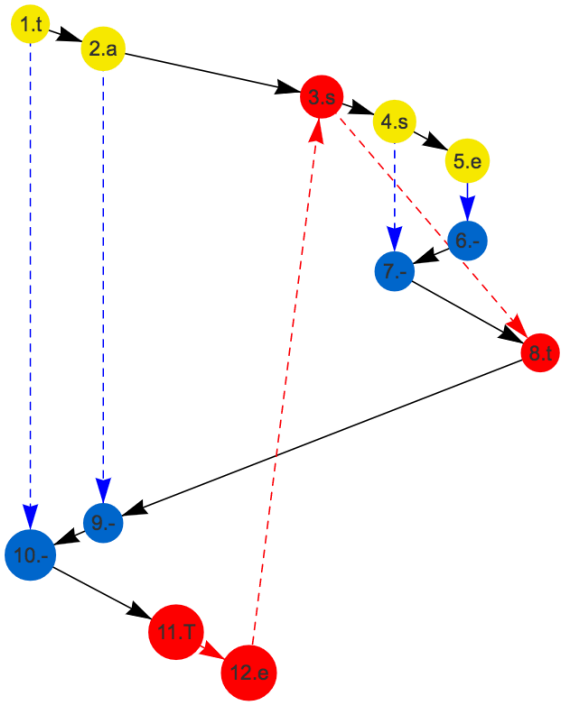

Figure 4.7 *Visualisation progressive* du fichier *log* 3.1

Ce graphe comporte 12 nœuds et 16 arcs.

La version finale du texte peut être lue en suivant les arcs et les nœuds rouges. Ce texte est :

Test

Grâce aux coordonnées (x, y) des nœuds, il est possible de repérer les versions intermédiaires du texte en regardant la visualisation. Pour ce faire, il faut sélectionner seulement les nœuds qui sont avant le temps *t* désiré, ou avant le nœud représentant la frappe *i* désirée.

Supposons que nous nous intéressons à l'état du texte correspondant à la 8ème frappe. Revoici la *visualisation progressive* du fichier *log* 3.1 pour laquelle toutes les frappes plus grandes que 8 sont cachées sous une portion gris foncé.

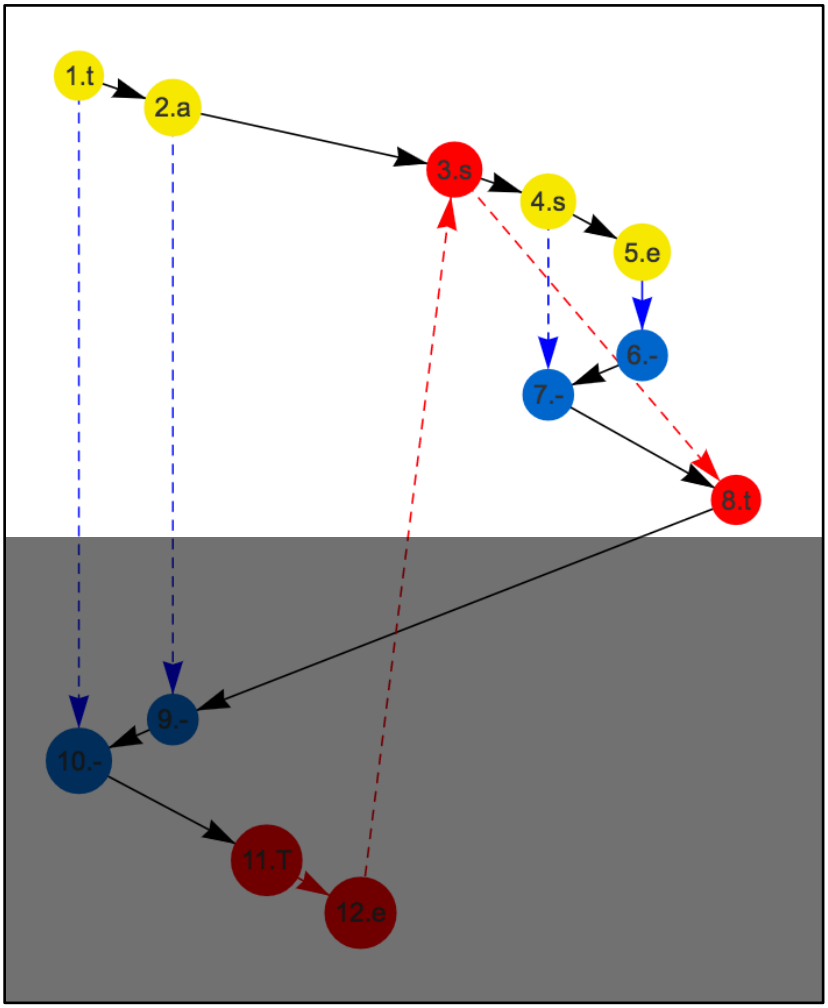

Figure 4.8 Emphase sur une portion de la *visualisation progressive*

Afin de lire les différents $AvantTexte$, il faut suivre les nœuds de gauche à droite en considérant leur couleur. Il faut considérer uniquement les nœuds qui sont dans la portion claire.

Dans le cas où un nœud est jaune, si le nœud bleu qui est son successeur est visible, on omet le caractère inséré dans la version du texte puisque cela signifie qu'il a déjà été supprimé dans cet $AvantTexte$. Si le nœud bleu qui est son successeur n'est pas visible, on inclut ce caractère dans la version du texte. C'est le cas pour les deux premiers caractères, qui forment les lettres *ta*. Les nœuds en rouge sont par défaut inclus dans le texte. Le caractère suivant (3.s) est donc rajouté et le texte est *tas.* Les caractères 4 et 5 sont les suivants (lettres s et e), sont en jaune et on voit clairement leurs nœuds bleus correspondants ce qui signifie que ces caractères ont été supprimés et n'apparaissent pas dans cette version du texte qui est toujours *tas.* Finalement, le dernier caractère (8.t) est en rouge donc il faut l'ajouter au texte qui est maintenant *tast*. Ce texte correspond bien à l'$AvantTexte(8)$ présenté dans le tableau 3.1.

Afin de créer la version finale de la visualisation progressive, deux algorithmes sont utilisés.

Le premier, sert à construire la base de la visualisation progressive, en créant les nœuds, les arcs *chronologiques*, les arcs entre une suppression et l'insertion qu'elle supprime, ainsi qu'en attribuant la valeur de coordonnée y aux nœuds. Le deuxième algorithme, lie les insertions qui sont encore présentes dans le texte final, selon leur ordre spatial.

Algorithme 2 : Structure de base du graphe de la *visualisation progressive*

Entrées : un fichier *log*

Sorties : un graphe *G = (N, A)* représentant un processus d'écriture

1 : Soit *G* un graphe vide
2 : **Pour toute** ligne $i = 1$ jusqu'à λ **faire**
3 : **Si** $Type(i) = I$ **alors**
4 : **Si** il existe un nœud *j* un nœud tel que $p_{i-1}(j) = Pos(i)$ **alors**
5 : Poser $posx = coordonnéeX_{i-1}(j)$
6 : **Sinon**
7 : Poser $posx = quantité\ de\ noeuds\ de\ type\ I\ dans\ G$
8 : **Pour tous** les nœuds k dans le graphe **faire**
9 : **Si** $coordonnéeX_{i-1}(k) \geq posx$ **alors**
10 : Poser $coordonnéeX_i(k) = coordonnéeX_{i-1}(k) + 1$

11 : **Si** $p_{i-1}(k) \geq 1$ **alors**
12 : Poser $p_i(k) = p_{i-1}(k) + 1$
13 : **Sinon**
14 : Poser $p_i(k) = p_{i-1}(k)$
15 : **Sinon**
16 : Poser $coordonnéeX_i(k) = coordonnéeX_{i-1}(k)$
17 : Poser $p_i(k) = p_{i-1}(k)$
18 : Créer le nœud i et l'ajouter dans G
19 : Poser $p_i(i) = Pos(i)$ et $coordonnéeX_i(i) = posx$
20 : Ajouter l'arc $(i-1, i)$
21 : **Si** $Type(i) = S$ **alors**
22 : Trouver le nœud j tel que $p_{i-1}(j) = Pos(i)$
23 : **Pour tous** les nœuds x dans G **faire** :
24 : Poser $coordonnéeX_i(x) = coordonnéeX_{i-1}(x)$
25 : **Si** $p_{i-1}(x) > Pos(i)$ **alors**
26 : Poser $p_i(x) = p_{i-1}(x) - 1$
27 : **Sinon**
28 : Poser $p_i(x) = p_{i-1}(x)$
29 : Créer le nœud i et l'ajouter dans G
30 : Poser $p_i(i) = -1$, $p_i(j) = -1$ et $coordonnéeX_i(i) = coordonnéeX_i(j)$
31 : Ajouter l'arc (j, i)
32 : **Si** $j \neq i - 1$ **alors** ajouter aussi l'arc $(i-1, i)$
33 : Retourner G

Algorithme 3 : Ajout des arcs reliant les nœuds du texte final dans la *visualisation progressive*

Entrées : un graphe *G = (N, A)* représentant un processus d'écriture

Sorties : un graphe *G = (N, A)* représentant un processus d'écriture comprenant les arcs reliant les nœuds du texte final

1 : Poser $\boldsymbol{x = 1}$
2 : Poser n_t le nombre de nœuds i dans G tels que $p_\lambda(i) \geq 1$
3 : **Tant que** $x < n_t$ **faire**
4 : Trouver le nœud i tel que $p_\lambda(i) = x$
5 : Trouver le nœud j tel que $p_\lambda(j) = x + 1$
6 : Ajouter l'arc (i, j) s'il n'existe pas
7 : Poser $x = x + 1$
8 : Retourner G

L'objectif de la *visualisation progressive* est d'illustrer l'ensemble des variables du processus d'écriture pour extraire plus facilement de l'information. La capacité de cette visualisation à ne perdre aucune information et à donner accès aux $AvantTexte$ fait partie de cet objectif. Le théorème 1 démontre qu'il est possible de recréer les différentes versions du texte grâce à la *visualisation progressive*, en considérant un temps ou une ligne spécifique du fichier *log*. À noter que ce théorème utilise seulement l'identifiant i des nœuds ainsi que la valeur $coordonnéeX_\lambda(i)$ de ceux-ci, considérant que λ exprime le nombre total de lignes dans le fichier *log*. Dans cet algorithme, les arcs de la *visualisation progressive* ne sont pas utilisés, ce qui est cohérent avec la description informelle du processus de reconstruction du texte présentée après la figure 4.8, plus haut dans cette même section.

Théorème 1 *La visualisation progressive permet de reconstruire* $AvantTexte(i)$.

PREUVE

Les caractères dans $AvantTexte(i)$ sont des insertions qui ont été effectuées avant i ou à ce moment et qui n'ont pas été supprimées au moment i.

Posons N_{I_i} l'ensemble de nœuds de type I dont l'identifiant est inférieur ou égal à i et N_{S_i} l'ensemble de nœuds de type S dont l'identifiant est inférieur ou égal à i.

Pour trouver $AvantTexte(i)$, il faut d'abord identifier l'ensemble N_{A_i} qui est tel que $N_{A_i} \subseteq N_{I_i}$ et qui comprend uniquement les nœuds de N_{I_i} qui n'ont pas été supprimés au moment i.

Pour chaque nœud j dans N_{I_i} :

- s'il existe un nœud k dans N_{S_i} tel que $coordonnéeX_\lambda(j) = coordonnéeX_\lambda(k)$, alors le caractère du nœud j n'est pas dans $AvantTexte(i)$.
- s'il n'existe pas un tel nœud k, alors le nœud j est dans $AvantTexte(i)$ et fait partie de l'ensemble N_{A_i}.

Considérant l'ordre induit par les valeurs $coordonnéeX_\lambda$, il suffit de trier les nœuds de l'ensemble N_{A_i} selon leur valeur $coordonnéeX_\lambda$ en ordre croissant pour reconstruire $AvantTexte(i)$. ■

Remarque 1 *Toutes les informations du fichier log peuvent être déduites à partir de la visualisation progressive.*

Le fichier *log* contient pour chacune de ses lignes i les informations suivantes : $Temps(i)$, $Type(i)$, $Pos(i)$ et $C(i)$.

Les valeurs $Temps(i)$, $Type(i)$ et $C(i)$ ne sont pas modifiées par les algorithmes 2 et 3 et sont donc allouées à chaque nœud tel quel et inversement, il est possible de les récupérer telles quelles.

La valeur $Pos(i)$ peut être déduite de la valeur $coordonnéeX_\lambda(i)$. En effet, selon sa définition, cette valeur nous permet de connaître toutes les frappes qui ont été insérées spatialement, sans être nécessairement temporellement, avant la frappe i dans le texte.

Pour chaque nœud i, il suffit de déterminer :

- n_I le nombre de nœuds de type I dont l'identifiant est inférieur à i et qui possèdent une valeur $coordonnéeX_\lambda$ inférieure à celle de i ;
- n_S le nombre de nœuds de type S dont l'identifiant est inférieur à i et qui possèdent une valeur $coordonnéeX_\lambda$ inférieure à celle de i.

Le calcul de $Pos(i)$ dépend du type du nœud.

Si $Type(i) = I$, alors on aura que $Pos(i) = n_I + n_S + 1$.

Si $Type(i) = S$, alors on aura que $Pos(i) = n_I + n_S$. ■

Bien que les arcs ne soient pas utilisés dans le théorème 1 ni dans la remarque 1, ceci signifie simplement que la relation qu'ils représentent est implicite et peut être déduite à partir du numéro des nœuds. Par contre, ceci n'implique pas nécessairement que les arcs soient facultatifs dans la *visualisation progressive*. En effet, les arcs permettent de constater clairement la relation entre les nœuds sans qu'il soit nécessaire de la déduire. Dans l'optique où cette visualisation vise à être

utilisée par un humain, l'application de codes visuels pour exprimer les relations structurelles est privilégiée (Bartram D. , 1980). Le cerveau gère ce type de données beaucoup plus rapidement que l'énumération de valeurs numériques (Ware, 2004, p. 151).

Il est à noter que la redondance de l'information ne doit pas nécessairement être évitée. En effet, il a été démontré qu'elle ne nuit pas à la précision et ni à la vitesse de perception de l'utilisateur (Chun, 2017). La redondance contribuerait à améliorer la compréhension des tendances dans les données puisque celles-ci sont présentées plus clairement (Borkin, et al., 2016).

4.2 Simplification du modèle – niveau 2

Il est possible de simplifier le modèle en diminuant le nombre d'opérations représentées. En contractant les opérations similaires, consécutives et non coupées par d'autres opérations, on obtient un graphe similaire à celui produit par la *représentation par les graphes*. Par rapport à cette dernière, l'avantage de la *visualisation progressive* est que chaque nœud possède sa propre position. D'une part, cet attribut fait en sorte que le graphe est plus rapide à consulter puisqu'il ne nécessite pas que l'utilisateur déplace manuellement les nœuds; et d'autre part, la position (x, y) de chaque nœud permet de reconstituer visuellement les *AvantTexte*.

Pour simplifier les explications, considérons à partir de maintenant que :

- Si un nœud i est tel que $Type(i) = I$ et $p_\lambda(i) = -1$, ce noeud représente une insertion qui a été supprimée. Pour simplifier les explications, elle sera maintenant dénotée $Type(i) = IS$.
- Le nœud j correspondant à l'acte de suppression de la frappe représentée par le nœud i sera dénoté $sup(i) = j$.

Lorsque deux nœuds de type I, i et $(i-1)$, ont été tapés consécutivement et qu'ils possèdent une $coordonnéeX$ consécutive telle que $coordonnéeX_\lambda(i) = coordonnéeX_\lambda(i-1) + 1$, cela

signifie qu'en plus d'être consécutifs dans le temps, ils sont toujours consécutifs dans le texte final. Dans ce cas uniquement, i et $(i-1)$ sont contractés.

Lorsque i et $(i-1)$ sont des nœuds de type *IS* et qu'ils possèdent également une $coordonnéeX$ consécutive telle que $coordonnéeX_\lambda(i) = coordonnéeX_\lambda(i-1)+1$, ils sont des candidats pour une contraction. Avant de procéder, il est nécessaire de vérifier que les actions de suppressions ont également été consécutives puisque ce n'est pas le cas par défaut. En pratique, deux insertions consécutives peuvent être supprimées à des moments non consécutifs. Les nœuds i et $(i-1)$ peuvent uniquement être contractés s'il existe deux nœuds j et $(j+1)$ de type S tels que les arcs (i,j) et $(i-1,j+1)$ existent. Si cette condition est remplie, les nœuds i et $(i-1)$ sont contractés et les nœuds j et $(j+1)$ sont également contractés.

Algorithme 4 : Simplification du graphe – niveau 2

Entrées : un graphe $G = (N, A)$ représentant un processus d'écriture

Sorties : un graphe $G' = (N', A')$ représentant un processus d'écriture en version simplifiée

1 : Poser *G'* une copie du graphe *G*
2 : Poser *i* une valeur correspondant au nombre de nœuds dans G
3 : **Tant que** $i > 0$ **faire**
4 : **Si** $Type(i) = I$ et $Type(i-1) = I$ **alors**
5 : **Si** $coordonnéeX_\lambda(i) = coordonnéeX_\lambda(i-1) + 1$ **alors**
6 : Contracter les nœuds *i* et $i-1$
7 : **Pour tout** nœuds *j* dans *G'* **faire**
8 : **Si** $coordonnéeX_\lambda(j) > coordonnéeX_\lambda(i)$ **alors**
9 : Poser $coordonnéeX_\lambda(j) = coordonnéeX_\lambda(j) - 1$
10 : **Si** (
11 : $Type(i) = IS$ et $Type(i-1) = IS$ et
12 : $coordonnéeX_\lambda(i) = coordonnéeX_\lambda(i-1) + 1$ et
13 : les nœuds j et $j+1$ tels que $Type(j) = S$ et $Type(j+1) = S$ existent et
14 : les arcs (i,j) et $(i-1,j+1)$ existent
15 :) **alors**
16 : Contracter les nœuds i et $(i-1)$
17 : Contracter les nœuds $(j+1)$ et j
18 : **Pour tout** nœuds *k* dans le graphe *G'* **faire**
19 : **Si** $coordonnéeX_\lambda(k) > coordonnéeX_\lambda(i)$ **alors**
20 : **Si** $coordonnéeX_\lambda(i) > coordonnéeX_\lambda(j+1)$ **alors**
21 : Poser $coordonnéeX_\lambda(k) = coordonnéeX_\lambda(k) - 2$
22 : **Sinon**
23 : Poser $coordonnéeX_\lambda(k) = coordonnéeX_\lambda(k) - 1$
24 : Retourner *G'*

La figure 4.9 présentée ci-après illustre l'exemple 3.1 dans sa version simplifiée :

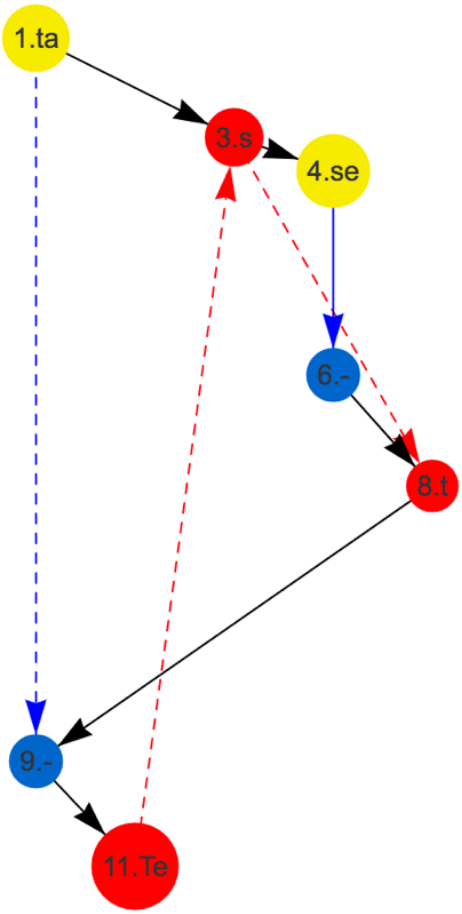

Figure 4.9 *Visualisation progressive* de niveau 2 du fichier *log* 3.1

La simplification permet de réduire le graphe à 7 nœuds et 10 arcs. Il est possible de lire les versions intermédiaires du texte de la même manière que décrit précédemment, en considérant l'ensemble du texte des nœuds.

Afin d'illustrer la reconstruction des *AvantTexte*, reprenons les nœuds représentant les frappes 1 à 8.

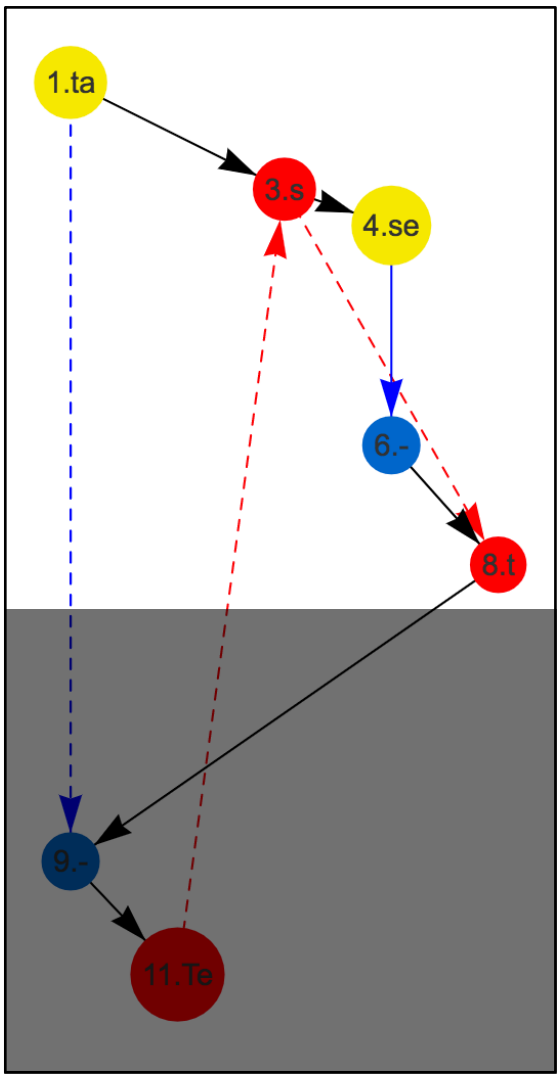

Figure 4.10 Emphase sur une portion de la *visualisation progressive* de niveau 2 du fichier *log* 3.1

Afin de lire les différents *AvantTexte*, il faut respecter les mêmes indications que celles fournies dans la Section 4.1.1.

Le premier nœud, soit celui contenant les lettres *ta* est jaune mais son nœud bleu correspondant n'est pas visible donc on inclut ces caractères dans la version de ce texte. Le nœud suivant, soit le nœud 3 est rouge donc le caractère qu'il contient est rajouté et le texte est *tas.* Le nœud 4 contient les caractères *se* mais puisqu'il est jaune et qu'on voit clairement son nœud bleu correspondant, ceci signifie que ces caractères ont été supprimés et n'apparaissent pas dans cette version du texte

qui est toujours *tas.* Finalement, le dernier caractère (8.t) est en rouge donc il faut l'ajouter au texte qui est maintenant *tast*. Ce texte correspond bien à l'$AvantTexte(8)$ présenté dans le tableau 3.1 et le résultat est le même que celui obtenu avec le graphe de niveau 1.

4.3 Avantages et inconvénients de la *visualisation progressive*

Les sous-sections suivantes détailleront les avantages et inconvénients de la *visualisation progressive*, par rapport spécifiquement aux trois visualisations qui l'ont inspirée. Chaque section abordera plus en détail les défauts de chacune, les solutions qu'apporte la *visualisation progressive* par rapport à ces défauts et les inconvénients de la *visualisation progressive* par rapport aux visualisations originales.

Par la suite, la *visualisation progressive* du fichier *log* 3.2 sera présentée pour permettre une comparaison plus réaliste des différents outils.

Finalement, une synthèse des avantages et inconvénients de la *visualisation progressive* sera présentée.

4.3.1 Avantages et inconvénients par rapport à la *représentation par les graphes*

L'exemple du fichier *log* 3.1 est assez simple et le lecteur peut se demander à cette étape-ci en quoi cette représentation possède plus d'avantages que son plus proche parent, la *représentation par les graphes* de Caporossi et Leblay (2011 ; 2014).

Un aspect primordial de la critique génétique est l'intérêt d'étudier les différentes versions d'un texte (Cislaru, 2015). La *représentation par les graphes* aide à capturer la dynamique du processus d'écriture mais elle est incomplète dans le sens qu'il est impossible de retracer les différentes versions du texte à partir du graphe final. Non seulement il y a une perte d'information mais le graphe résultant n'est pas unique à un processus spécifique. En effet, dans la Section 3.2.3, il a été démontré que la représentation par les graphes n'est pas unique à un processus d'écriture.

À des fins de comparaisons, la figure 4.11 présente les *visualisations progressives* des fichiers *log* 3.3, 3.4 et 3.5 respectivement. La *visualisation progressive* produit bien une visualisation unique à chaque fichier *log* contrairement à la *représentation par les graphes*.

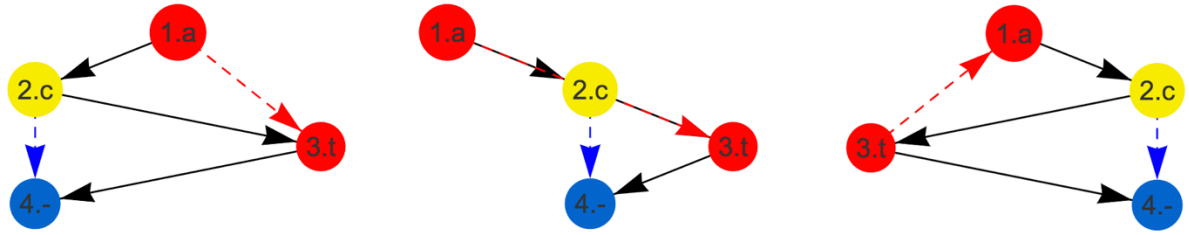

Figure 4.11 *Visualisation progressive* des fichiers *log* 3.3, 3.4 et 3.5

Bien que la structure du graphe dans la *visualisation progressive* soit la même que celle dans la *représentation par les graphes*, le positionnement des nœuds selon une méthode de coordonnées (x, y) permet plusieurs choses :

- incorpore la variable de spatialité absolue ;
    - permet de retrouver l'ensemble des relations spatiales des différentes frappes ;
    - permet de reconstruire les différents $AvantTexte$;
- inclut la variable de temporalité ;
- permet d'obtenir une seule visualisation propre à un processus.

La *visualisation progressive* permet donc de valoriser la structure de la *représentation par les graphes* en lui ajoutant de nombreux avantages.

4.3.2 Avantages et inconvénients par rapport au *SIG*

Les informations contenues dans le *SIG* peuvent être déduites de la *visualisation progressive*. Par contre, puisqu'elles ne sont pas directement représentées, un certain calcul doit être effectué par

l'utilisateur, ce qui demande un travail cognitif supplémentaire. Dans le cas où un utilisateur cherche spécifiquement à comparer l'emplacement dans le texte de la frappe *i* au moment où celle-ci a été exécutée ($Pos(i)$), le nombre de caractères dans $AvantTexte(i)$ et le nombre total de frappes de type *I* exécutées jusqu'à ce moment, le *SIG* reste l'outil le plus simple.

Autrement, la *visualisation progressive* permet de mettre en valeur la position spatiale des frappes de manière plus précise et contextualisée que ce que permet le *SIG*.

4.3.3 Avantages et inconvénients par rapport à la *représentation linéaire*

La *représentation linéaire* tire sa force de sa relative simplicité et de la possibilité de lire le processus d'écriture à la même manière qu'un texte.

Tel que décrit dans la Section 4.2, la *visualisation progressive* contient une version éclatée de la *représentation linéaire*. En effet, l'ordre des frappes est exactement la même dans les deux visualisations. *A priori* la même information est présente dans les deux visualisations. Par contre, puisque les mots et phrases sont déconstruits en nœuds dans la *visualisation progressive*, ceci fait en sorte que le texte soit moins rapide à lire que dans la *représentation linéaire*.

Par contre, la *visualisation progressive* inclut aussi la notion temporelle et permet d'illustrer autant le flux global du processus que de voir de près les opérations effectuées lors de l'écriture du texte tandis que la *représentation linéaire* permet seulement un point de vue de près.

4.3.4 Exemple comparatif

Afin d'illustrer plus clairement les avantages et inconvénients de la *visualisation progressive* par rapport aux visualisations existantes, reprenons le fichier *log* 3.2. Ceci permettra au lecteur de se faire une meilleure idée de l'utilisation de la *visualisation progressive*.

Voici la *visualisation progressive* de ce fichier *log* :

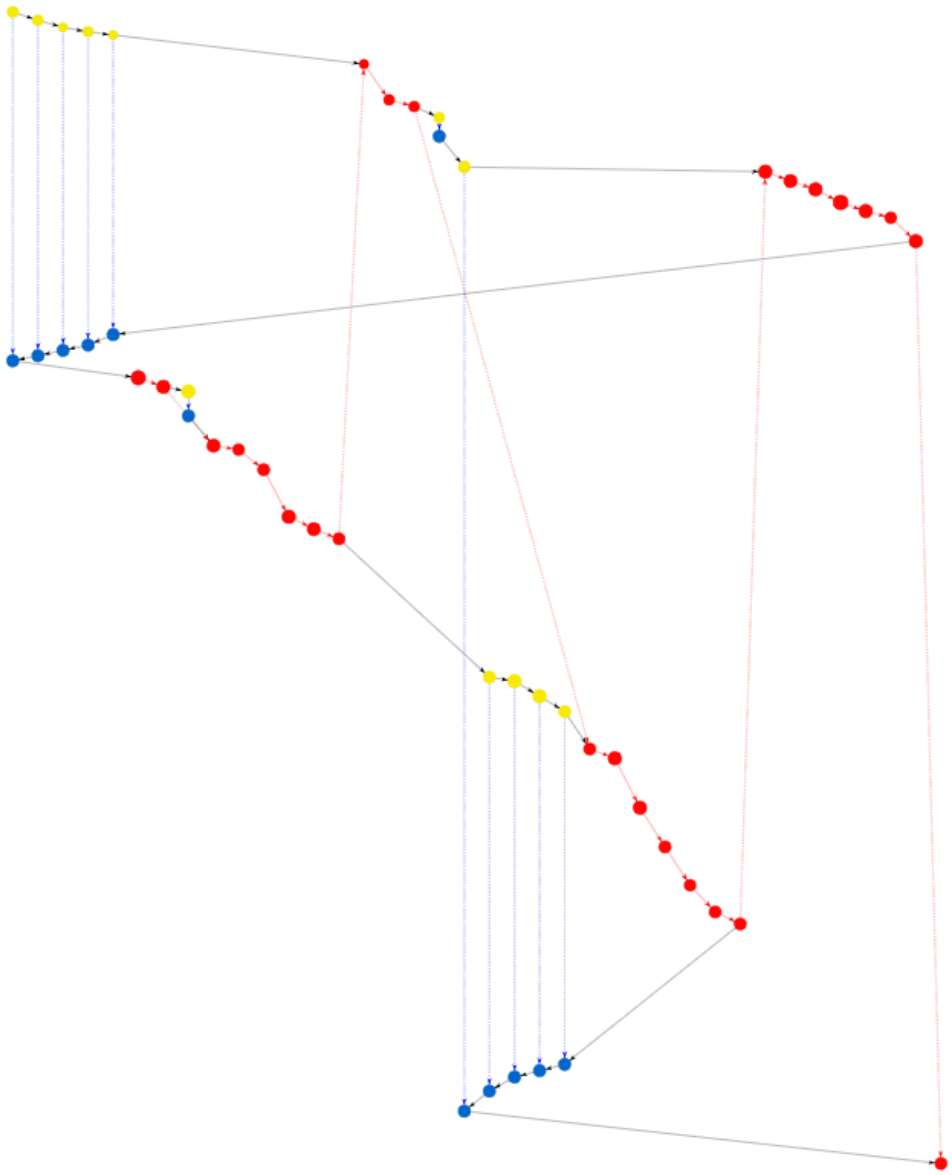

Figure 4.12 *Visualisation progressive* du fichier *log* 3.2

Ce graphe comporte 50 nœuds et 64 arcs. Plus le graphe est gros, plus il est difficile de visualiser l'ensemble des frappes et leur contenu. Ce problème est également présent dans l'ensemble des autres visualisations, à l'exception des *SIG*. Le recours à la fonction zoom et à la version simplifiée de niveau 2 du graphe permettent d'une part de permettre au chercheur d'avoir accès à une vue rapprochée du processus et d'autre part, de diminuer la quantité d'information présentée.

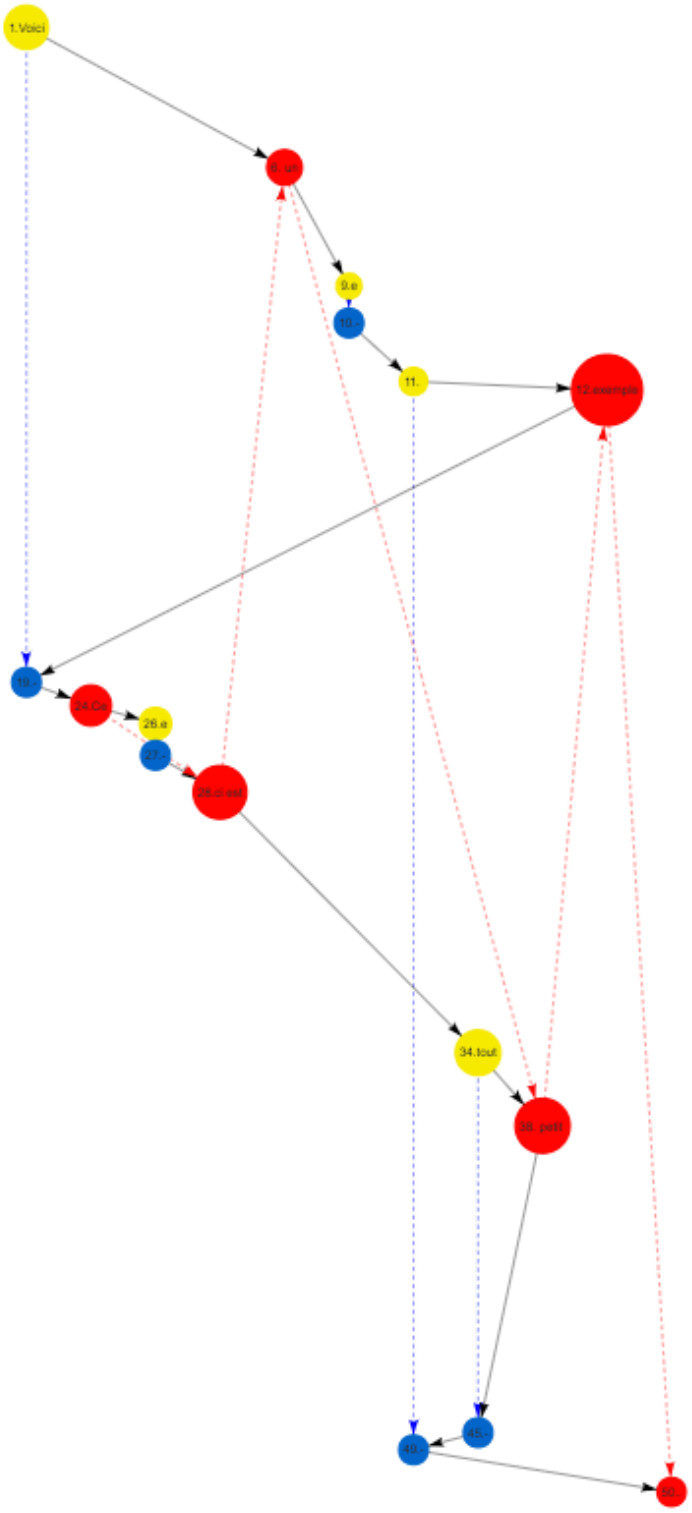

Figure 4.13 Version simplifiée de niveau 2 du fichier *log* 3.2

La simplification permet de réduire le graphe de 50 à 16 nœuds et de 64 à 23 arcs.

Le lecteur peut bien observer sur cet exemple les propriétés décrites précédemment. La *visualisation progressive* de niveau 1 est facilement comparable avec le *SIG* puisque chaque point/nœud représente une frappe du fichier *log*. Le *SIG* correspondant au fichier *log* 3.2 a été présenté dans le Chapitre 3 à la figure 3.6. Remarquons que par rapport aux axes représentant la temporalité dans chacune des visualisations, la position des nœuds de la courbe verte du *SIG* est similaire à la position des nœuds dans la *visualisation progressive*. L'avantage de la *visualisation progressive* quant à la mise en contexte des frappes est plus explicite par rapport au *SIG*.

La *représentation par les graphes* et la *représentation linéaire* sont plus facilement comparables avec la *visualisation progressive* de niveau 2.

Les nœuds et arêtes de la *représentation par les graphes* sont exactement les mêmes que les nœuds et arcs de la *visualisation progressive*. Concernant la *représentation linéaire*, la similarité peut être constatée au-delà de la position des frappes, qui correspond à la position des nœuds sur l'axe des x dans la *visualisation progressive*. La *Notation-S* du fichier *log* 3.2 a été présenté dans la Section 3.2.1. Les crochets et accolades représentant les interruptions permettent de constater implicitement les nœuds de la *visualisation progressive*. En effet, tout ensemble de caractères n'étant pas interrompu par un crochet ou une accolade représente un nœud de type *I* ou *IS* dans la *visualisation progressive* de niveau 2.

Dans le cas où un chercheur souhaite tout de même utiliser conjointement plusieurs visualisations dans son processus d'analyse du processus d'écriture, cette correspondance d'information qui est sous-jacente à la *visualisation progressive* en fait un outil versatile.

4.3.5 Synthèse des caractéristiques de la *visualisation progressive*

Afin d'uniformiser les caractéristiques de cette visualisation avec les précédentes détaillées dans la Section 2.3, voici les caractéristiques de la représentation et de la présentation de la *visualisation progressive* présentées dans un format similaire.

Représentation des données	
Quelles sont les variables utilisées ?	Temporalité, chronologie, spatialité relative et absolue.
Comment sont-elles représentées?	Similaire au *SIG*, chaque frappe possède sa coordonnée (x, y). L'axe des x représente le temps et l'axe des y représente la spatialité absolue des opérations.
Quel est le degré de précision?	Précis ou imprécis en fonction du niveau de la vue sélectionnée du processus.
Présentation des données	
Quelles sont les caractéristiques de la présentation des données?	L'utilisation des couleurs aide l'utilisateur à connaître la nature de l'opération unique. Le même code que pour la *représentation par les graphes* est utilisé. Le rouge est utilisé pour le texte présent dans le texte final. Le jaune est pour les ajouts qui ont été supprimés et le bleu pour l'opération de suppression. Concernant les arcs, les couleurs et les textures de ceux-ci permettent à l'utilisateur de bien cerner la relation entre deux nœuds. Il est également possible d'effectuer un zoom avant ou arrière, d'avoir une vue très précise du processus ou de l'afficher sous forme de vue d'ensemble, comme dans le cas du *SIG*.

Afin d'offrir un comparatif des avantages et inconvénients juste de la *visualisation progressive* par rapport aux visualisations existantes, rappelons que celle-ci possède des aspects de représentation et de présentation des données, et qu'elle n'est pas qu'un modèle puisque l'objectif de la création de visualisations est d'aider l'utilisateur à reconnaître des structures dans les données étudiées.

Dans le cas où un scripteur effectue plusieurs retours dans le texte et que le fichier *log* résultant est volumineux, il peut être difficile de manœuvrer dans le graphe et de reconstituer le fil de l'écriture.

Par contre, l'avantage principal de cette visualisation est que non seulement les variables temporelles/chronologiques sont représentées, mais que la variable spatiale soit aussi présente dans ses deux déclinaisons : relative et absolue. Finalement, ses propriétés dynamiques permettent autant d'étudier une portion spécifique du processus que d'en capter le flot global.

4.4 Conclusion

Le *SIG*, la *représentation linéaire* et la *représentation par les graphes* sont trois outils utilisés par les chercheurs pour comprendre le processus d'écriture. Aucune de ces trois visualisations n'arrive

à la fois à prendre en compte la temporalité, la chronologie, la spatialité relative et absolue. Pour avoir accès à l'ensemble des variables et aux relations entre les données d'écriture, les chercheurs utilisent généralement plus d'une visualisation dans leurs analyses.

La *visualisation progressive* arrive à bien amalgamer les forces de ces trois visualisations et à illustrer l'ensemble des variables inhérentes au processus d'écriture. De plus, la *visualisation progressive*, étant dynamique, permet aux chercheurs de s'intéresser autant au processus dans son ensemble que sur une portion spécifique du texte. Considérant que l'objectif de cette thèse est de faciliter la découverte de tendances complexes dans les données, le fait que les chercheurs n'aient plus à recourir à plus d'une visualisation contribue certainement à simplifier le processus d'analyse.

Le chapitre suivant présente une seconde visualisation inspirée de la théorie de graphes, qui s'appelle le modèle *sans pertes*. Il s'agit d'un graphe représentant le processus d'écriture, possédant des propriétés structurelles et ne nécessitant pas de positionnement de ses nœuds pour qu'il soit possible de retrouver les différents *AvantTexte*.

CHAPITRE 5 MODÈLE *SANS PERTES*

La *visualisation progressive*, présentée au Chapitre 4, a été créée pour inclure toutes les dimensions du processus d'écriture dans une seule visualisation, dans un contexte où aucune visualisation du processus d'écriture dans la littérature ne présente de telles propriétés. Conséquemment, la *visualisation progressive* permet de mieux mettre en relief les tendances complexes dans les données puisque toutes les relations entre les frappes du fichier *log* sont présentées.

La structure des graphes est souvent utilisée pour mettre en relief la relation entre plusieurs objets (Ahuja, Magnanti, & Orlin, 1993, p. 1). La *visualisation progressive*, bien qu'elle comprenne des nœuds et des arcs, ne permet pas de bien exploiter les propriétés structurelles des graphes puisque la relation entre les nœuds doit être observée à partir de leurs coordonnées (x, y) .

Une seconde visualisation, qui s'appelle le modèle *sans pertes*, a été créée pour modéliser le fichier *log*. La structure de ce graphe est telle qu'il est possible d'analyser structurellement celui-ci en utilisant entre autres des mesures propres à la théorie des graphes. Cette contribution correspond à l'objectif global de la thèse qui est de modéliser le fichier *log* de manière à extraire plus facilement des motifs complexes.

Bien que cette nouvelle visualisation soit également créée à partir d'une structure de graphe, le graphe résultant n'est pas le même que celui de la *visualisation progressive*.

Ce chapitre présentera d'abord la méthode de construction du modèle *sans pertes*. Par la suite, des aspects de présentation et représentation des données qui peuvent être appliqués seront discutés. Il y aura ensuite la présentation d'une version simplifiée et plus compacte. Finalement, les différents avantages et inconvénients du modèle *sans pertes* par rapport aux autres visualisations seront détaillés.

Des mesures découlant de la structure du graphe *sans pertes* seront présentées au Chapitre 6.

5.1 Description du modèle *sans pertes*

Le graphe *sans pertes* représentant le processus d'écriture est généré à partir du fichier *log*.

Pour mieux illustrer la création du graphe, voici le processus de création du graphe *sans pertes* du fichier *log* 3.3. Ce fichier *log* a été présenté dans la Section 3.2.3.

Dans les graphes suivants, le numéro du nœud x est identifié à l'intérieur du cercle tandis que le numéro à l'extérieur du cercle représente la position $p_i(x)$ du nœud, considérant que i est le numéro de la dernière ligne du fichier *log* considérée, ce qui signifie que i correspond à la plus grande valeur de nœud dans le graphe. Il est à noter que $p_i(x)$ est une valeur différente de $Pos(x)$ et les détails concernant $p_i(x)$ seront donnés dans les prochains paragraphes. Rappelons que $Pos(x)$ est la position dans $AvantTexte(i)$ de la frappe lorsque celle-ci a été effectuée et que cette valeur provient directement du fichier *log*.

La première étape de construction du modèle *sans pertes* commence par la création d'un graphe comportant deux nœuds : un nœud -1 qui correspond au début du texte et un nœud 0 qui correspond à la fin du texte et l'arc $(-1, 0)$. Le début du texte, soit le nœud -1 possède toujours la position 0 donc $p_i(-1) = 0$, pour tout $i = 0, 1, \dots, \lambda$. La figure illustrant le graphe est ci-après.

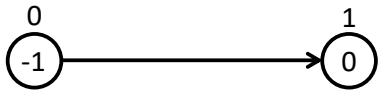

Figure 5.1 Première étape de la construction de tout graphe *sans pertes* du fichier *log* 3.3

Ensuite, lorsqu'une ligne représentant une frappe *i* est modélisée dans le graphe, un nœud *i* représentant cette frappe est créé.

Lorsque $Type(i) = I$, le nœud *i* représentant cette frappe est relié aux nœuds *j* tel que $p_{i-1}(j) = Pos(i) - 1$ et *k* tel que $p_{i-1}(k) = Pos(i)$ par les arcs (j, i) et (i, k). Par la suite, pour chaque nœud $x \neq i$ tel que $p_{i-1}(x) \geq Pos(i)$, cette valeur de position est augmentée d'une unité tel que $p_i(x) = p_{i-1}(x) + 1$. Finalement, le nœud i est tel que $p_i(i) = Pos(i)$.

Lorsque $Type(i) = S$, le nœud *i* représentant cette frappe est relié aux nœuds *j* tel que $p_{i-1}(j) = Pos(i) - 1$ et *k* tel que $p_{i-1}(k) = Pos(i) + 1$ par les arcs (j, i) et (i, k). Il existe un nœud u tel que $p_{i-1}(u) = Pos(i)$. La valeur de $p_{i-1}(u)$ devient $p_i(u) = -1$. Par la suite, pour chaque nœud $x \neq i$ tel que $p_{i-1}(x) > Pos(i)$, cette valeur de position est diminuée d'une unité tel que $p_i(x) = p_{i-1}(x) - 1$. Finalement, le nœud i est tel que $p_i(i) = -1$.

La construction du graphe *sans pertes* du fichier *log* 3.3 se poursuit comme suit :

La ligne 1 du fichier *log* consiste en une insertion ayant la position $Pos(1) = 1$. Puisque $p_0(-1) = 0$ et $p_0(0) = 1$, cette frappe doit être reliée aux nœuds -1 et 0 par les arcs $(-1, 1)$ et $(1, 0)$ puisque $p_0(-1) = 0$ et $p_0(0) = 1$. Le seul nœud x ayant une position $p_0(x)$ plus grande ou égale à 1 est le nœud 0 donc $p_1(0) = p_0(0) + 1 = 1 + 1 = 2$ de manière à former le graphe suivant :

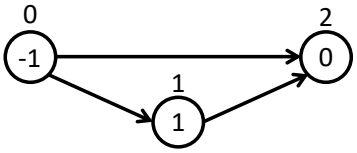

Figure 5.2 Deuxième étape de la construction du graphe *sans pertes* du fichier *log* 3.3

La ligne 2 du fichier *log* consiste en une insertion ayant la position $Pos(2) = 1$. Puisque $p_1(-1) = 0$ et $p_1(1) = 1$, cette frappe doit être reliée aux nœuds -1 et 1 par les arcs $(-1, 2)$ et $(2, 1)$. Deux nœuds ont une position plus grande ou égale à 1 : 1 et 0. Donc $p_1(1)$ devient $p_2(1) = p_1(1) + 1 = 1 + 1 = 2$ et $p_2(0) = p_1(0) + 1 = 2 + 1 = 3$ de manière à former le graphe suivant :

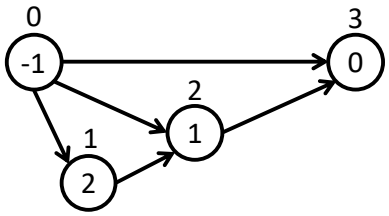

Figure 5.3 Troisième étape de la construction du graphe *sans pertes* du fichier *log* 3.3

La ligne 3 du fichier *log* consiste en une insertion ayant la position $Pos(3) = 3$. Cette frappe doit être reliée aux nœuds 1 et 0 par les arcs $(1, 3)$ et $(3, 0)$ puisque $p_2(1) = 3 - 1 = 2$ et $p_2(0) = 3$. Le seul nœud x ayant une position $p_2(x)$ plus grande ou égale à 3 est le nœud 0 donc $p_3(0) = p_2(0) + 1 = 3 + 1 = 4$ de manière à former le graphe suivant :

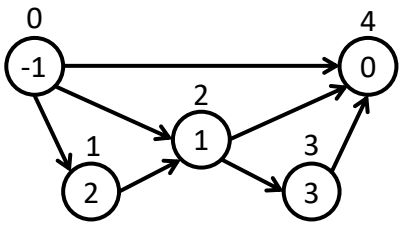

Figure 5.4 Quatrième étape de la construction du graphe *sans pertes* du fichier *log* 3.3

Finalement la ligne 4 du fichier *log* consiste en une suppression ayant la position $Pos(4) = 1$. Cette frappe doit être reliée aux nœuds -1 et 1 par les arcs $(-1, 4)$ et $(4, 1)$ puisque $p_3(-1) = 0$ et $p_3(1) = 2$. Le nœud ayant une position égale à $Pos(4) = 1$ est le nœud 2 donc $p_4(2) = -1$ et $p_4(4) = -1$. Il y a trois nœuds ayant une position p_3 plus grande ou égale à 1 : le nœud 1, le nœud 3 et le nœud 0. Donc $p_4(1) = p_3(1) - 1 = 2 - 1 - 1$ tandis que $p_4(3) = p_3(3) - 1 = 3 - 1 = 2$ et $p_4(0) = p_3(0) - 1 = 4 - 1 = 3$ de manière à former le graphe suivant :

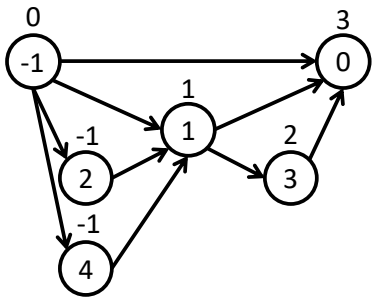

Figure 5.5 Cinquième étape de la construction du graphe *sans pertes* du fichier *log* 3.3

L'algorithme 5 décrit de façon plus formelle la construction de ce graphe.

- Soit λ le nombre de lignes dans le fichier *log* et $TexteFinal = AvantTexte(\lambda)$.
- Chaque frappe i est un nœud dans le graphe. Les valeurs $Type(i)$, $Temps(i)$ et $C(i)$ sont automatiquement attitrées à chaque nœud.
- Deux nœuds supplémentaires sont créés : un nœud -1 qui correspond au début du texte et un nœud 0 qui correspond à la fin du texte. Conséquemment, le nombre de nœuds dans G est égal à $\lambda + 2$ et les nœuds sont numérotés de -1 à λ.

- Pour chaque nœud x tel que $Type(x) = I$, dénotons $p_i(x)$ la position de $C(x)$ dans $AvantTexte(i)$. Si $C(x)$ a été supprimé par la frappe représentée par le nœud y, alors $p_i(x) = -1$, pour tout $i \geq y$.
- Pour chaque nœud x tel que $Type(x) = S$, $p_i(x) = -1$, pour tout $i \geq x$.
- Si $AvantTexte(i) = \emptyset$, on note que $p_i(-1) = 0$ et $p_i(0) = 1$.

Algorithme 5 : Modèle *sans pertes*

Entrées : un fichier *log*

Sorties : un graphe *G = (N, A)* représentant un processus d'écriture

1 : Créer un graphe $\boldsymbol{G}$ qui contient les nœuds $\mathbf{-1}$ et $\mathbf{0}$; et l'arc $(\mathbf{-1}, \mathbf{0})$
2 : Poser $p_0(-1) = 0$ et $p_0(0) = 1$
3 : **Pour toute** ligne $i = 1$ jusqu'à λ **faire**
4 : Trouver le nœud j tel que $p_{i-1}(j) = Pos(i) - 1$
5 : Trouver le nœud k tel que $p_{i-1}(k) = Pos(i)$
7 : **Si** $Type(i) = I$ **alors**
8 : Poser $p_i(i) = Pos(i)$
9 : **Pour tous** les nœuds x dans le graphe **faire**
10 : **Si** $p_{i-1}(x) \geq Pos(i)$ **alors**
11 : Poser $p_i(x) = p_{i-1}(x) + 1$
12 : **Sinon**
13 : Poser $p_i(x) = p_{i-1}(x)$
14 : Créer le nœud i et l'ajouter dans G
15 : **Si** $Type(i) = S$ **alors**
16 : Poser $p_{i-1}(i) = -1$
17 : **Pour tous** les nœuds x dans le graphe **faire**
18 : **Si** $p_{i-1}(x) > Pos(i)$ **alors**
19 : **Si** $p_{i-1}(x) = Pos(i) + 1$ **alors**
20 : Poser $p_i(k) = -1$ et $k = x$
21 : **Sinon**
22 : Poser $p_i(x) = p_{i-1}(x) - 1$
23 : **Sinon**
24 : Poser $p_i(x) = p_{i-1}(x)$
25 : Ajouter les arcs (j, i) et (i, k)
26 : Retourner G

L'algorithme 6 présente comment recréer les différentes versions du texte, en considérant un temps ou une ligne spécifique du fichier *log*.

Algorithme 6 : Reconstruction du texte après avoir lu *i* lignes du *log*, à partir du graphe *sans pertes*

Entrées : un graphe $G = (N, A)$ représentant un processus d'écriture, une ligne *i* d'un fichier *log* de λ lignes

Sorties : $Texte(i)$

1 : Poser $\boldsymbol{G'}$ le sous-graphe induit de $\boldsymbol{G}$ contenant uniquement les nœuds $\boldsymbol{j \leq i}$
2 : Poser $Texte = \emptyset$ et $j = -1$
3 : Tant que $j \neq 0$ faire
4 : Soit V une liste contenant les successeurs du nœud j
5 : Poser k le nœud dans V qui possède l'identifiant ayant la valeur la plus élevée
6 : **Si** $Type(k) = I$ **alors**
7 : Ajouter le caractère $C(k)$ à la fin de $Texte(i)$
5 : Poser $j = k$
8 : Retourner $Texte(i)$

Nous pouvons maintenant prouver le théorème qui décrit qu'il y a un seul texte qui peut être obtenu à partir des *i* premières lignes du fichier *log* et qu'il est possible de recréer exactement ce texte à partir d'un chemin dans le graphe correspondant au même fichier *log*.

Théorème 2 *Pour tous les i de 0 à λ,* $Texte(i)$ *est le texte qui serait obtenu en considérant uniquement les i premières lignes du fichier log.*

PREUVE Nous allons démontrer par induction que $AvantTexte\,(i) = Texte(i)$ pour $i = 0, 1, \ldots, \lambda$

Pour $i = 0$, le résultat est trivial.

Supposons que le résultat soit vrai pour les premières i lignes du fichier *log*. Nous allons maintenant démontrer que le résultat est aussi vrai pour les $i + 1$ premières lignes.

- Si $Type(i+1) = S$, $AvantTexte(i+1)$ est obtenu en supprimant le caractère en position $Pos(i+1)$ dans $AvantTexte(i)$.

  Par hypothèse d'induction, $Texte(i) = AvantTexte(i)$ et le graphe résultant de l'algorithme après i instructions du fichier *log* a trois nœuds u, v et w tels que $p_i(u) = Pos(i+1) - 1$, $p_i(v) = Pos(i+1)$ et $p_i(w) = Pos(i+1) + 1$.

  Après avoir atteint le nœud u, l'algorithme va suivre un chemin qui lui permettra d'atteindre v. Tous les nœuds sur ce chemin sont de type *S*. Après avoir atteint le nœud v, l'algorithme va suivre un chemin qui lui permettra d'atteindre w et tous les nœuds sur ce chemin sont de type *S*.

  Lorsque la $i+1$ ième ligne du fichier *log* sera considérée par l'algorithme, il y aura l'ajout du nœud $i+1$ et des arcs $(u, i+1)$ et $(i+1, w)$. Ainsi, le nœud $i+1$ devient le successeur de u ayant l'identifiant le plus élevé.

  Après avoir atteint le nœud u, $AvantTexte(i)$ et $AvantTexte(i+1)$ sont les mêmes. L'algorithme de construction de $AvantTexte(i+1)$ se rend de u à $i+1$ sans ajouter de caractère puisqu'il s'agit d'un nœud de type *S* puis à w puisqu'il s'agit du seul successeur de $i+1$.

- Si $Type(i+1) = I$, alors $AvantTexte(i+1)$ est obtenu en insérant le caractère $C(i+1)$ en position $Pos(i+1)$ dans $AvantTexte(i)$.

  Par hypothèse d'induction, $Texte(i) = AvantTexte(i)$ et le graphe résultant de l'algorithme après i instructions du fichier *log* a deux nœuds u et v tels que $p_i(u) = Pos(i+1) - 1$ et $p_i(v) = Pos(i+1)$.

  Après avoir atteint le nœud u, l'algorithme va suivre un chemin qui lui permettra d'atteindre v. Tous les nœuds sur ce chemin sont de type *S*.

  Lorsque la $i+1$ ième ligne du fichier *log* sera considérée par l'algorithme, il y aura l'ajout du nœud $i+1$ et des arcs $(u, i+1)$ et $(i+1, v)$. Le nœud $i+1$ devient le successeur de u ayant l'identifiant le plus élevé.

  Après avoir atteint le nœud u, $AvantTexte(i)$ et $AvantTexte(i+1)$ sont les mêmes.

- Si $v = 0$, alors $AvantTexte(i)$ contient strictement $Pos(i)$ caractères et $AvantTexte(i+1)$ est obtenu en rajoutant $C(i+1)$ en dernière position dans $AvantTexte(i)$, ce qui signifie que $Texte(i+1) = AvantTexte(i+1)$.
- Si $v \neq 0$, l'algorithme de construction de $AvantTexte(i+1)$ se rend de u à $i+1$ en ajoutant le caractère $C(i+1)$ à $AvantTexte(i+1)$ puis l'algorithme se rend à v puisqu'il s'agit du seul successeur de $i+1$. La fin du texte à partir de v est la même dans $AvantTexte(i)$ et $AvantTexte(i+1)$.

  On a donc bien que $AvantTexte(i+1)$ est obtenu en rajoutant $C(i+1)$ entre les positions $Pos(i+1)-1$ et $Pos(i+1)$ dans $AvantTexte(i)$, ce qui signifie que $Texte(i+1) = AvantTexte(i+1)$. ■

Une fois la construction du graphe terminée, si la seule utilité du graphe est de servir de visualisation, le nœud $i = 0$ devient superflu. Dans le but d'alléger la représentation visuelle, ce nœud et tous ses arcs entrants sont généralement omis. Dans le cas où des calculs sont effectués sur la structure du graphe, le nœud $i = 0$ est par contre utile, ce qui sera détaillé au Chapitre 6.

La figure suivante présente le fichier *log* 3.2 sous la forme du graphe *sans pertes*, en incluant le nœud 0. Ce fichier *log* est présenté dans la Section 3.2 et il a aussi été représenté grâce à la *visualisation progressive* dans la Section 4.3.

Notons qu'à l'exception des nœuds -1 et 0, les nœuds du graphe *sans pertes* sont les mêmes que les nœuds de la *visualisation progressive*, puisque dans les deux cas, une frappe du fichier *log* est représentée par un nœud.

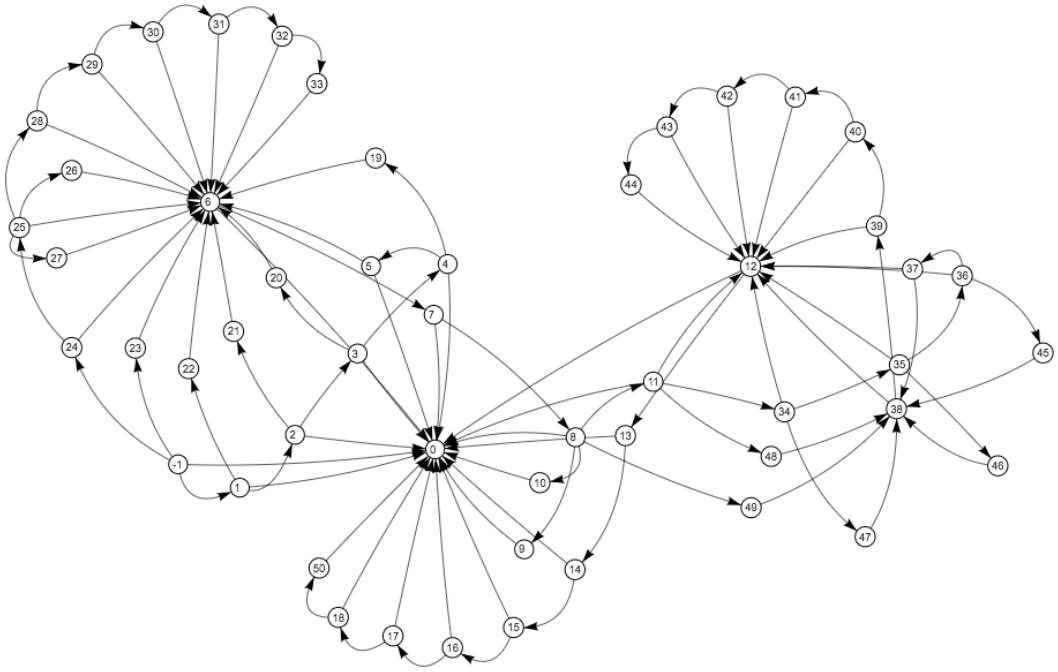

Figure 5.6 Graphe *sans pertes* du fichier *log* 3.2

5.2 Présentation et représentation du graphe

Les propriétés découlant de ce modèle sont très intéressantes d'un point de vue structurel et seront abordées dans le chapitre suivant. Par contre, dans l'optique où le graphe est une représentation visuelle des données et si celle-ci est utilisée en tant qu'outil visuel par l'utilisateur, il est pertinent de garder en tête les concepts de représentation et présentation de données introduits dans la revue de littérature.

Selon Bartram (1997), une combinaison appropriée de formes, de symboles, de couleurs, de tailles et de positions devrait être choisie parce que ce type d'information visuelle n'exige pas d'effort cognitif et qu'elle est plutôt traité par le système visuel pré-attentif. L'affichage d'une combinaison complète de codes visuels et de dimensions augmente l'efficacité et la rapidité de l'analyste en regardant la représentation (Aigner, Miksch, Schumann, & Tominski, 2011). Les couleurs et les formes sont généralement privilégiées pour distinguer différentes variables (Tufte, 2001, p. 154).

Afin d'améliorer la facilité d'utilisation de la visualisation *sans pertes*, les nœuds peuvent être colorés ou représentés par des formes différentes en fonction de leur type.

Dans leur version de la *représentation par les graphes*, Caporossi et Leblay (2011) utilisent des couleurs pour identifier en un coup d'œil le type de frappe. Pour une insertion, la couleur rouge est utilisée, une insertion supprimée est en jaune et enfin le bleu l'est pour une suppression. Ces couleurs ont été d'ailleurs reprises dans la *visualisation progressive* présentée au Chapitre 4.

La figure suivante montre à quoi pourrait ressembler le graphe avec des nœuds colorés.

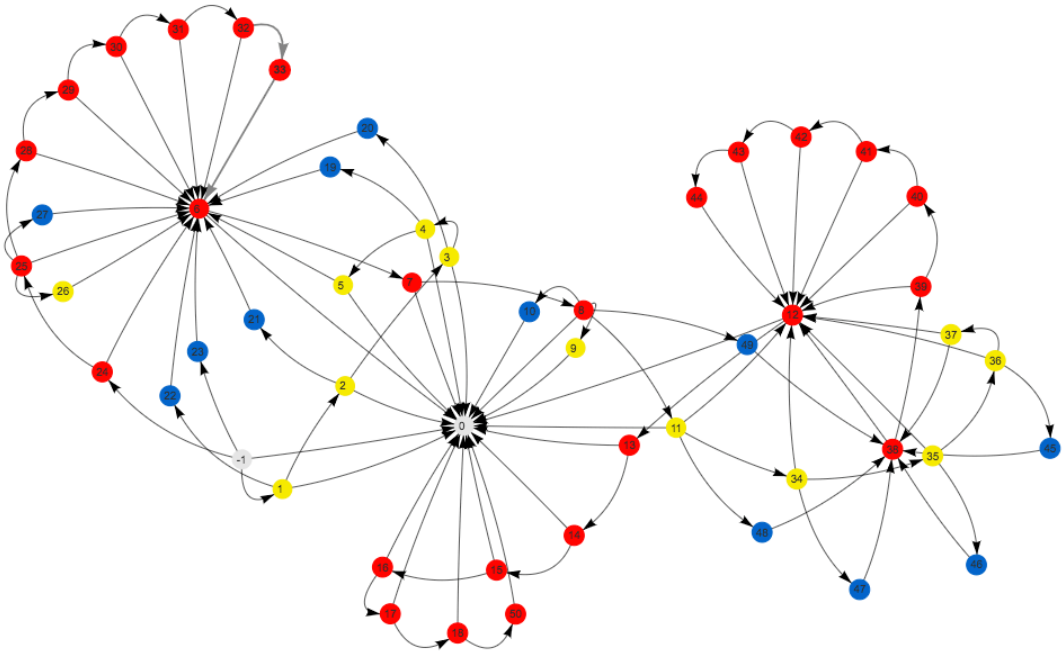

Figure 5.7 Modèle *sans pertes* de niveau 1 coloré du fichier *log* 3.2

5.3 Simplification du modèle – niveau 2

Tel que présenté dans le cas de la *visualisation progressive*, les graphes résultants peuvent être assez denses. En pratique, le graphe du modèle *sans pertes* comporte deux nœuds de plus que la *visualisation progressive* pour le même fichier *log*. Puisque l'un des objectifs de la transformation du fichier *log* en graphe est de visualiser les données afin de mieux comprendre, de mieux identifier

des modèles et des stratégies d'écriture plus détaillées, le modèle *sans pertes* s'éloigne de ces objectifs lorsqu'il devient trop dense.

5.3.1 Création du graphe *sans pertes* de niveau 2

La régularité la plus simple dans les données du processus d'écriture est la *consécutivité* d'une séquence d'opérations dans le temps et l'espace et a déjà été employée dans la première représentation graphique du processus d'écriture (Caporossi & Leblay 2011) ainsi que dans la *visualisation progressive* présentée au Chapitre 4.

Le but du graphe *sans pertes* de niveau 2 est de contracter tous les nœuds de même type qui sont consécutifs dans le temps et dont la *consécutivité* n'est pas brisée par d'autres frappes et ce, à partir du graphe *sans pertes*. Il peut être pratique de se rappeler que dans le graphe *sans pertes*, chaque frappe (caractère inséré ou suppression) représente un nœud dans le graphe.

Pour illustrer le fonctionnement de l'algorithme, voici deux exemples.

5.3.1.1 Exemple du fichier *log* 5.1

Si un scripteur écrit de façon continue « le test », le graphe au départ contient un nœud par caractère, soit 7 en incluant la frappe espace, le nœud -1 et le nœud 0 ce qui fait un total de 9 nœuds. Par contre, puisqu'il n'y aucune interruption entre ces caractères et qu'ils ont été écrits de manière consécutive, lors de la contraction, toutes les lettres entre guillemets seront contractées ensemble. Si le fichier *log* ne contient pas plus de frappes que celles déjà mentionnées, le graphe contracté contient maintenant seulement 3 nœuds : -1, 0 et la contraction de « le test ».

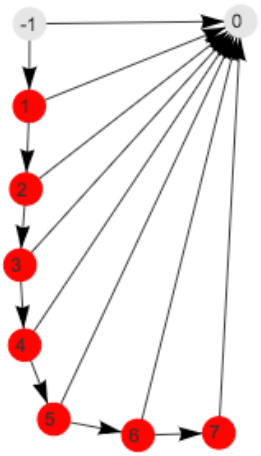

Figure 5.8 Graphe *sans pertes* du fichier *log* 5.1

La version contractée du graphe du fichier *log* 5.1 est présentée dans la figure 5.9.

Figure 5.9 Graphe *sans pertes* de niveau 2 du fichier *log* 5.1

5.3.1.2 Exemple du fichier *log* 5.2

Si par contre le scripteur écrit « le test », qu'il supprime par la suite « test » et qu'il insère à cet endroit « texte », le processus d'écriture est alors segmenté en quatre séquences : « le », « test », la suppression de « test », « texte ». Le graphe au départ contient 3 + 5 + 5 = 12 nœuds insertions, 4 nœuds suppressions, le nœud -1 et le nœud 0 ce qui fait un total de 18 nœuds. Même si « le test » a été écrit en un jet, l'action de supprimer le mot « test » brise cette suite. Chacune des 4 séquences est composée de frappes de même type qui ont été écrites chronologiquement. Si le fichier *log* ne contient pas plus de frappes que celles déjà mentionnées, le graphe contracté contiendrait 6 nœuds : -1, 0, « le », « test », suppression de « test », « texte ». Le graphe présenté ci-après correspond au modèle *sans pertes*. Les nœuds 1-2-3 sont « le », les nœuds 4-5-6-7 sont « test », les nœuds 8-9-10-11 sont la suppression de « test », et les nœuds 12-13-14-15-16 sont « texte ».

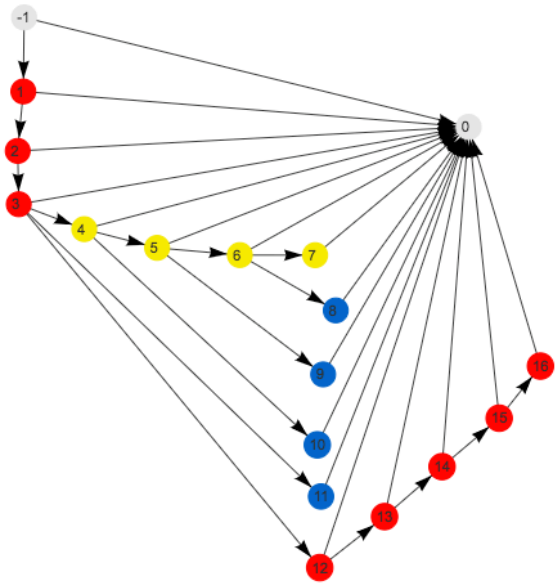

Figure 5.10 Graphe *sans pertes* du fichier *log* 5.2

En résumé, si deux nœuds dans le graphe sont de même type, qu'ils ont été exécutés de façon consécutive sans qu'aucun nœud ne vienne briser leur séquence, on les contracte.

La version contractée du graphe du fichier *log* 5.2 est présentée dans la figure 5.11.

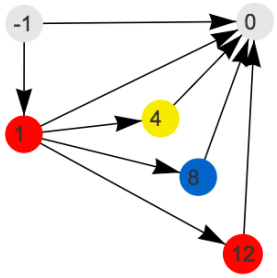

Figure 5.11 Graphe *sans pertes* de niveau 2 du fichier *log* 5.2

5.3.1.3 Algorithme de création du niveau 2

L'algorithme 7 détaille la manière dont ces opérations sont effectuées.

Pour simplifier les explications, considérons à partir de maintenant que :

- Si un nœud i est tel que $Type(i) = I$ et $p_\lambda(i) = -1$, ce noeud représente conséquemment une insertion qui a été supprimée. Pour simplifier les explications, elle sera maintenant dénotée comme $Type(i) = IS$.
- Le nœud j correspondant à l'acte de suppression de la frappe insertion représentée par le nœud i sera considérée comme $sup(i) = j$.

Algorithme 7 : Niveau 2 du modèle *sans pertes*

Entrées : un graphe *G = (N, A)* représentant un processus d'écriture

Sorties : un graphe contracté *G' = (N', A')* représentant le même processus d'écriture

1 : Poser *G'* une copie du graphe *G*
2 : Poser $i = |A| - 2$
3 : **Tant que** $i > 0$ **faire**
4 : **Si** $Type(i) = I$ et $Type(i-1) = I$ **alors**
5 : **Si** l'arc $(i-1, i)$ existe et que $p_\lambda(i) = p_\lambda(i-1) + 1$ **alors**
6 : Poser S_{i-1} l'ensemble contenant tous les successeurs du nœud $i-1$
7 : **Si** $|S_{i-1}| = 2$ **alors**
8 : Trouver le nœud k faisant partie de l'ensemble S_{i-1} tel que $k \neq i$
9 : **Si** les arcs $(i-1, k)$ et (i, k) existent **alors**
10 : Contracter les nœuds $i-1$ et i
11 : Ajouter le caractère $C(i)$ au début de la suite de caractères $C(i-1)$
12 : **Si** $Type(i) = IS$ et $Type(i-1) = IS$ **alors**
13 : **Si** $sup(i-1) = sup(i) + 1$ et l'arc $(i-1, i)$ existe **alors**
14 : Poser S_{i-1} l'ensemble contenant tous les successeurs du nœud $i-1$
15 : **Si** $|S_{i-1}| = 3$ **alors**
16 : Trouver le nœud *k* faisant partie de l'ensemble S_{i-1} tel que $k \neq i$ et $k \neq sup(i)$
17 : Poser $S_{sup(i)}$ l'ensemble contenant tous les successeurs du nœud $sup(i)$
18 : Poser $S_{sup(i-1)}$ l'ensemble contenant tous les successeurs du nœud $sup(i-1)$
19 : **Si** les arcs $(i-1, k)$ et (i, k) existent et que $S_{sup(i)} = S_{sup(i-1)}$ **alors**
20 : Supprimer l'arc $(i-1, sup(i))$
21 : Contracter les nœuds $(i-1)$ et i
22 : Ajouter le caractère $C(i)$ au début de la suite de caractères $C(i-1)$
23 : Contracter les nœuds $sup(i-1)$ et $sup(i)$
24 : Poser $i = i - 1$
25 : Retourner *G'*

La figure 5.12 montre le modèle *sans pertes* de niveau 2 du fichier *log* 3.2 dans sa version colorée.

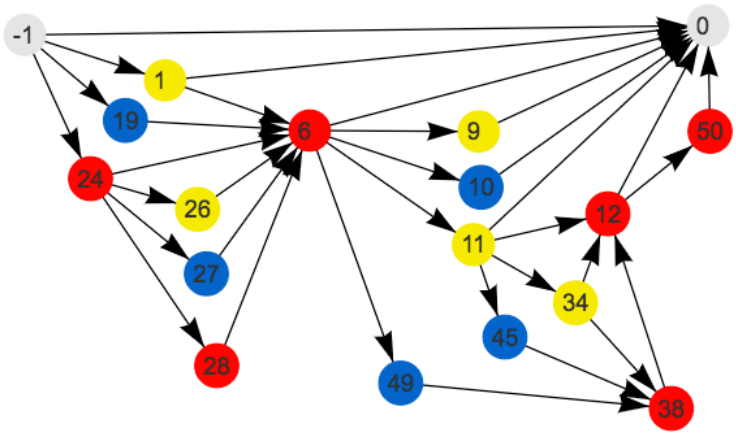

Figure 5.12 Modèle *sans pertes* de niveau 2 du fichier *log* 3.2

5.4 Avantages et inconvénients du modèle

Les sous-sections suivantes détailleront les avantages et inconvénients du modèle *sans pertes*, d'abord par rapport à la *visualisation progressive*, puis en synthétisant les caractéristiques du modèle *sans pertes*.

5.4.1 Avantages et inconvénients par rapport à la *visualisation progressive*

A priori le modèle *sans pertes* peut être utilisé en tant que visualisation mais ce n'en est pas une au sens du terme défini dans la Section 2.2.2 où les attributs de présentation et de représentation sont sélectionnés spécifiquement pour faciliter son utilisation.

Pour tout de même offrir un certain type de comparaison par rapport à la *visualisation progressive* et rapprocher cette comparaison de celle faite à la Section 4.3.1, les fichiers *log* 3.3, 3.4 et 3.5 seront utilisés. Les nœuds sont illustrés avec le même code de couleur et le même contenu présenté dans ceux-ci, à savoir le numéro de la frappe et le caractère associé s'il y a lieu. Notons que le fichier *log* 3.3 a aussi servi à expliquer le processus de la création du modèle *sans pertes* dans la Section 5.1.

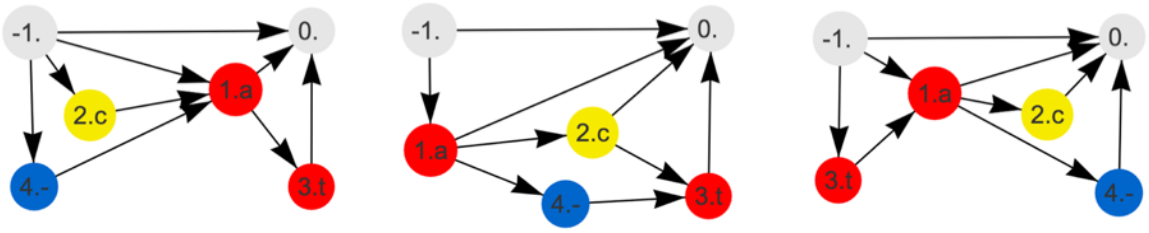

Figure 5.13 Modèle *sans pertes* des fichiers *log* 3.3, 3.4 et 3.5

Dans le cas où le modèle *sans pertes* serait utilisé en tant que visualisation, celui-ci se trouve donc désavantagé par rapport à la *visualisation progressive* parce qu'il n'a pas été spécifiquement conçu à cet effet. De plus, la variable temporelle n'est pas présente visuellement dans la version du modèle *sans pertes* décrite dans ce chapitre. C'est donc dire que par rapport à la *visualisation progressive*, le modèle *sans pertes* est moins complet au niveau des variables utilisées.

Le grand avantage du modèle *sans pertes* est que l'ensemble des informations du processus d'écriture soit contenu dans sa structure. Conséquemment, il est possible s'utiliser des mesures propres à la théorie des graphes sur le modèle *sans pertes*, ce que la *visualisation progressive* ne permet pas. En effet, les propriétés de conservation de l'information de la *visualisation progressive* découlent de la position des nœuds et non pas de la structure du graphe.

5.4.2 Synthèse des caractéristiques du modèle *sans pertes*

Les aspects de présentation des données et même certains aspects de la représentation des données ne sont pas inhérents au modèle. Conséquemment, ceux-ci peuvent être ajustés au besoin.

Voici les caractéristiques de la représentation et de la présentation du modèle *sans pertes* :

Représentation des données	
Quelles sont les variables utilisées ?	Chronologie et spatialité. Les deux types de spatialité (absolue et relative) sont illustrés.
Comment sont-elles représentées?	Chaque frappe ou ensemble de frappes ayant des caractéristiques similaires est représenté par un nœud qui est voisin d'un ou plusieurs nœuds qui sont spatialement voisins (de façon absolue et/ou relative).
Quel est le degré de précision?	Précis ou imprécis en fonction du niveau de la vue sélectionnée du processus.
Présentation des données	

Quelles sont les caractéristiques de la présentation des données?	*A priori* aucun attribut de présentation des données n'est spécifique à cette visualisation sauf la possibilité d'effectuer un zoom avant ou arrière. Il est par contre possible d'emprunter les conventions de couleur à d'autres modèles.

Les avantages principaux du modèle *sans pertes* seront démontrés dans le Chapitre 6, où des nouvelles mesures découlant de la structure du graphe *sans pertes* seront présentées.

5.5 Conclusion

La structure du graphe *sans pertes*, permet notamment de retrouver l'information contenue dans le fichier *log* et implicite à celui-ci, sans égard à la position des nœuds. Cette caractéristique favorise sans aucun doute la détection de motifs complexes dans les données d'écriture.

Le but premier du modèle *sans pertes* est la création d'un graphe dans lequel on peut structurellement trouver des tendances en utilisant certaines propriétés des graphes, ce qui n'a pu être évalué dans ce chapitre.

Le prochain chapitre présente les mesures de proximité chronologique, qui sont calculées à partir de la structure du graphe *sans pertes*. Ces nouvelles mesures permettent de trouver des irrégularités ou tendances qui sont difficiles à repérer par les visualisations existantes du processus d'écriture.

CHAPITRE 6 MESURES DE PROXIMITÉ CHRONOLOGIQUE

L'un des moyens privilégiés pour découvrir des tendances intéressantes rapidement dans les données d'écriture est l'utilisation de visualisations. Le *SIG* permet d'observer de manière chronologique la quantité de retours dans le texte et les épisodes d'insertions et de suppressions (Lindgren E. , Sullivan, Lindgren, & Spelman Miller, 2007). La *représentation par les graphes*, et par extension, la *visualisation progressive*, permet d'observer la dynamique spatiale d'un processus d'écriture (Caporossi & Leblay, 2011). La *Notation-S*, bien qu'elle permette de contextualiser chronologiquement les différentes révisions, nécessite un certain niveau d'analyse pour son interprétation et permet plus difficilement la recherche rapide de motifs complexes dans les données d'écriture (Kollberg, 1996).

Un processus d'écriture peut comprendre, par exemple, plusieurs épisodes d'insertions qui sont effectués au même endroit dans le texte, mais à des moments différents. Dans ce cas, il est possible de savoir à quel endroit dans l'*AvantTexte* les frappes sont exécutées. Cette information est d'ailleurs affichée telle quelle dans le fichier *log* et est reprise dans le *SIG*. Par contre, il existe peu de moyens de connaître quels sont les caractères spatialement contigus aux frappes lorsque celles-ci sont exécutées. Une telle information permettrait notamment de contextualiser les révisions avec plus de précision et de trouver des irrégularités ou tendances intéressantes.

La *visualisation progressive* et le graphe *sans pertes*, permettent d'avoir accès à cette information. Par contre, il faut la déduire individuellement pour chaque frappe. Cette information n'est donc pas déjà calculée, ni disponible en format synthétisé ou visuel.

Ce chapitre vise à présenter les mesures de proximité chronologique. Ces mesures sont calculées directement à partir du graphe *sans pertes*. Ces mesures indiquent si une frappe *i* est chronologiquement proche ou loin des frappes qui lui sont immédiatement spatialement contiguës dans l'*AvantTexte*, au moment où *i* est exécutée. Lorsque ces valeurs sont présentées dans un format visuel, il est d'ailleurs possible de repérer des motifs nouveaux, en un simple coup d'œil. Ces mesures permettent donc de répondre à l'objectif de cette thèse, qui est de favoriser la détection des motifs complexes dans les données d'écriture.

Ce chapitre présentera d'abord la méthode de calcul des mesures de proximité chronologique. Par la suite, un exemple permettant de démontrer la pertinence et le potentiel de ces mesures sera détaillé. Finalement, les avantages et inconvénients de la proximité chronologique seront présentés.

6.1 Calcul de la proximité chronologique

Deux valeurs de proximité sont calculées pour chaque nœud i représentant une frappe dans un graphe. Posons i le nœud pour lequel l'indice de proximité chronologique est calculé.

La proximité antérieure, dénotée ϕ_i^-, se calcule à partir des prédécesseurs de i. La proximité postérieure, dénotée ϕ_i^+, se calcule à partir des successeurs de i.

6.1.1 Proximité antérieure

Notons ϕ_i^- la proximité antérieure du nœud i.

Posons j le seul nœud prédécesseur de i tel que $j < i$. Autrement dit, j correspond au prédécesseur de i au moment où le nœud i est ajouté au graphe.

La proximité antérieure d'un nœud i est égale à la différence entre i et son prédécesseur j tel que $\phi_i^- = i - j$.

Lorsque i et j sont des frappes consécutives type *I*, la valeur de la proximité antérieure sera toujours de 1.

Lorsqu'une frappe i est exécutée au début du texte, son prédécesseur est $j = -1$. En prenant le calcul de proximité antérieure énoncé plus haut, on obtiendrait que $\phi_i^- = i - (-1) = i + 1$. En prenant par exemple la frappe $i = 1$, on aurait que $\phi_1^- = 1 - (-1) = 2$. Un tel résultat n'est pas cohérent. Pour éviter cette situation, lorsque $j = -1$ la valeur de proximité antérieure est égale à i.

Le calcul complet de la proximité antérieure est donc :

$$\phi_i^- = \begin{cases} i - j & \text{si } j \neq -1 \\ i & \text{sinon} \end{cases}$$

6.1.2 Proximité postérieure

Notons ϕ_i^+ la proximité postérieure du nœud i.

Posons k le seul nœud successeur de i tel que $k < i$. Autrement dit, k correspond au successeur de i au moment où le nœud i est ajouté au graphe.

La proximité postérieure d'un nœud i est égale à la différence entre i et son successeur k tel que $\phi_i^+ = i - k$.

Lorsqu'une frappe i est exécutée à la fin de l'*AvantTexte*, son successeur est $k = 0$. En prenant le calcul de proximité postérieure énoncé plus haut, on obtiendrait que $\phi_i^+ = i - (0) = i$. Par contre, ce résultat n'est pas nécessairement cohérent.

Prenons un court exemple, le fichier *log* 6.1, qui s'exécute en 3 grandes étapes.

1. Frappes 1 à 3 :

   le scripteur écrit *Tet* de manière à ce que le texte final à ce moment précis soit ***Tet*** .

2. Frappe 4:

   le scripteur change son curseur d'endroit pour se positionner devant le *t* minuscule et insère la lettre *x*. Le texte à ce moment-ci est ***Text*** .

3. Frappe 5:

   le scripteur repositionne son curseur à la fin du texte et insère la lettre *e*. Le texte à ce moment-ci est ***Texte*** .

Le graphe sans pertes correspondant au fichier log 6.1 est présenté dans la figure 6.1.

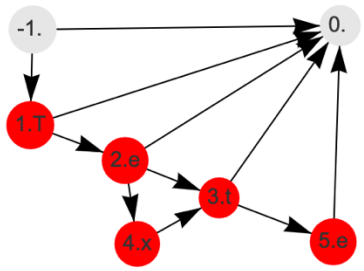

Figure 6.1 Graphe *sans pertes* du fichier *log* 6.1

En prenant la formule $\phi_i^+ = i - k$, on obtiendrait que $\phi_3^+ = 3 - (0) = 3$, $\phi_4^+ = 4 - (3) = 1$ et $\phi_5^+ = 5 - (0) = 5$. Les valeurs de proximité postérieures sont très différentes, alors qu'en réalité ces trois frappes sont exécutées presqu'au même endroit.

En pratique puisqu'il n'y a rien après la fin du texte, il serait plus cohérent que lorsque $k = 0$, la valeur de proximité antérieure soit égale à 0.

En appliquant cette règle, on obtient que $\phi_3^+ = 0$, $\phi_4^+ = 4 - (3) = 1$ et $\phi_5^+ = 0$. Les valeurs démontrent maintenant de façon beaucoup plus claire que les frappes ont été exécutées consécutivement, presqu'au même endroit dans le texte.

Le calcul complet de la proximité postérieure est donc :

$$\phi_i^+ = \begin{cases} i - k & \text{si } k \neq 0 \\ 0 & \text{sinon} \end{cases}$$

6.1.3 Proximité chronologique dans la *visualisation progressive*

Bien que les mesures de proximité chronologique aient d'abord été pensées pour être dérivées du graphe *sans pertes*, il est aussi possible de retrouver ces valeurs dans la *visualisation progressive*. Le calcul s'inspire de la procédure pour retrouver les *AvantTextes* dans cette visualisation. Conséquemment, il est beaucoup moins intuitif de retrouver les valeurs de proximité chronologique par la *visualisation progressive* que par le calcul initial simple dérivé de la structure du graphe *sans pertes*.

6.1.3.1 Proximité antérieure dans la *visualisation progressive*

Rappelons que lors de la construction de la *visualisation progressive*, le nœud correspondant à j, prédécesseur de i au moment où le nœud i est ajouté au graphe, est tel que si $Pos(i) > 1$, alors $coordonnéeX_{i-1}(j) = Pos(i) - 1$. Dans le cas où $Pos(i) = 1$, cela signifie que la frappe est exécutée au début du texte.

Il est possible retrouver j à partir uniquement de la version finale de la *visualisation progressive* puisque celle-ci a la propriété de ne perdre aucune information. La frappe j doit avoir été exécutée avant la frappe i. La frappe j doit être une insertion. La frappe j ne doit pas avoir été supprimée lorsque la frappe i est exécutée.

En résumé, j doit satisfaire l'ensemble des critères suivants :

- $j < i$;
- $coordonnéeX_\lambda(j) < coordonnéeX_\lambda(i)$;
- $Type(j) = I$;
- il n'existe pas de nœud u tel que $coordonnéeX_\lambda(j) = coordonnéeX_\lambda(u)$ et que $u < i$.

Si plusieurs nœuds répondent à ces critères, alors on considérera que j est le nœud possédant la plus grande valeur $coordonnéeX_\lambda$ parmi tous ceux trouvés.

Si aucun nœud ne répond à l'ensemble de ces critères, alors on considérera que $j = -1$ puisque cela signifie que la frappe est exécutée au tout début du texte.

Lorsque i et j sont identifiés, la proximité antérieure se calcule de la même façon que décrit dans la Section 6.1.1.

6.1.3.2 Proximité postérieure dans la *visualisation progressive*

Rappelons que lors de la construction de la *visualisation progressive*, le nœud correspondant à k, successeur de i au moment où le nœud i est ajouté au graphe, dans le cas où $Pos(i) < |AvantTexte(i-1)|$, est tel que :

- si $Type(i) = I$, $coordonnéeX_{i-1}(k) = Pos(i)$;
- si $Type(i) = S$, $coordonnéeX_{i-1}(k) = Pos(i) + 1$.

Dans le cas où $Pos(i) = |AvantTexte(i-1)|$, cela signifie que la frappe est exécutée à la fin du texte.

Tout comme dans le cas de la proximité antérieure, il est possible retrouver k à partir uniquement de la version finale de la *visualisation progressive*. La frappe k doit avoir été exécutée avant la frappe i. La frappe k doit être une insertion. La frappe k ne doit pas avoir été supprimée lorsque la frappe i est exécutée.

En résumé, k doit satisfaire l'ensemble des critères suivants :

- $k < i$;
- $coordonnéeX_\lambda(k) > coordonnéeX_\lambda(i)$;
- $Type(k) = I$;
- il n'existe pas de nœud v tel que $coordonnéeX_\lambda(k) = coordonnéeX_\lambda(v)$ et que $v < i$.

Si plusieurs nœuds répondent à ces critères, alors on considérera que k est le nœud possédant la plus petite valeur $coordonnéeX_\lambda$ parmi tous ceux trouvés.

Si aucun nœud ne répond à l'ensemble de ces critères, alors on considérera que $j = 0$ puisque cela signifie que la frappe est exécutée dans le pré-contextuel, c'est-à-dire à la toute fin du texte.

Lorsque que i et k sont identifiés, la proximité postérieure se calcule de la même façon que décrit dans la Section 6.1.2.

6.2 Analyse des tendances de la proximité chronologique

Afin de présenter l'analyse des tendances de la proximité chronologique, voici un exemple, le fichier *log* 6.2. L'écriture de cet exemple se fait en 7 grandes étapes.

1. Frappes 1 à 24 :

   le scripteur écrit d'abord *Il chantait dans le pré.* de manière à ce que le texte final à ce moment précis soit ***Il chantait dans le pré.*** .

2. Frappes 25 à 32 :

   le scripteur change son curseur d'endroit pour supprimer la lettre *l* du mot *Il* et insère *sabelle*, sans repositionner son curseur. Le texte à ce moment-ci est ***Isabelle chantait dans le pré.*** .

3. Frappes 33 à 42 :

   le scripteur change à nouveau son curseur d'endroit pour se positionner au début du texte et insère *La petite* de manière à ce que le texte final à ce moment précis soit ***La petite Isabelle chantait dans le pré.*** .

4. Frappes 43 à 51 :

   le scripteur change son curseur d'endroit pour se positionner devant le mot *dans* et insère *gaiement* . Le texte à ce moment-ci est ***La petite Isabelle chantait gaiement dans le pré.*** .

5. Frappes 52 à 57 :

   le scripteur change son curseur d'endroit pour se positionner devant le mot *Isabelle* et insère *fille*. Le texte à ce moment-ci est ***La petite fille Isabelle chantait gaiement dans le pré.*** .

6. Frappes 58 à 84 :

   le scripteur change à nouveau son curseur d'endroit pour supprimer *Isabelle chantait* et insère *fredonnait* de manière à ce que le texte devienne ***La petite fille fredonnait gaiement dans le pré.*** .

7. Frappes 85 à 100 :

   le scripteur change son curseur d'endroit pour se positionner à la fin du texte, supprime *e pre.* et insère *a prairie.* sans repositionner son curseur. C'est ainsi que le texte prend sa forme finale soit ***La petite fille fredonnait gaiement dans la prairie.*** .

Les mesures de proximité antérieure et postérieure de chaque nœud du fichier *log* 6.2 ont été calculées. Dans le but d'utiliser ces valeurs à leur plein potentiel, elles sont présentées sur un diagramme à courbes dans la figure 6.2. L'axe des abscisses représente la chronologie des frappes tandis que l'axe des ordonnées représente la valeur de proximité de chaque frappe.

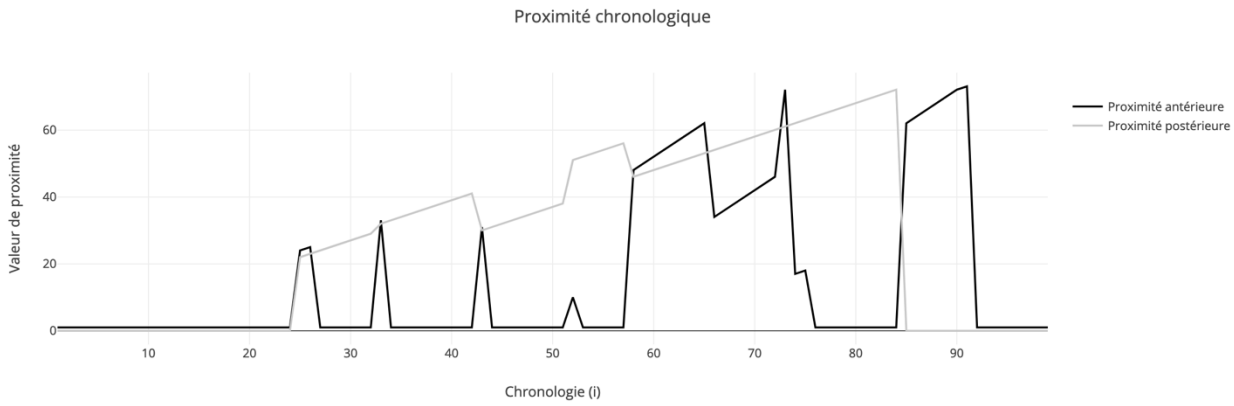

Figure 6.2 Courbes de proximité chronologique du fichier *log* 6.2

La courbe de proximité antérieure varie à six endroits. Les quatre premiers blocs sont plutôt simples tandis que le cinquième, composé des frappes 58 à 75, possède une forme intrigante.

A priori, il pourrait sembler que cette portion du processus d'écriture comporte plus d'un repositionnement du curseur. Par contre, la courbe de proximité postérieure croît à un rythme régulier entre les frappes 58 à 75. Ceci confirme que le curseur n'a pas été repositionné entre l'exécution de ces frappes.

La figure 6.3 reprend les courbes de proximité chronologique du fichier *log* 6.2 en y superposant de l'information supplémentaire. Les lignes verticales vertes indiquent quand le curseur a été repositionné. Les rectangles verts non opaques indiquent les épisodes de suppression. Le texte inséré est également présenté sous l'axe des abscisses.

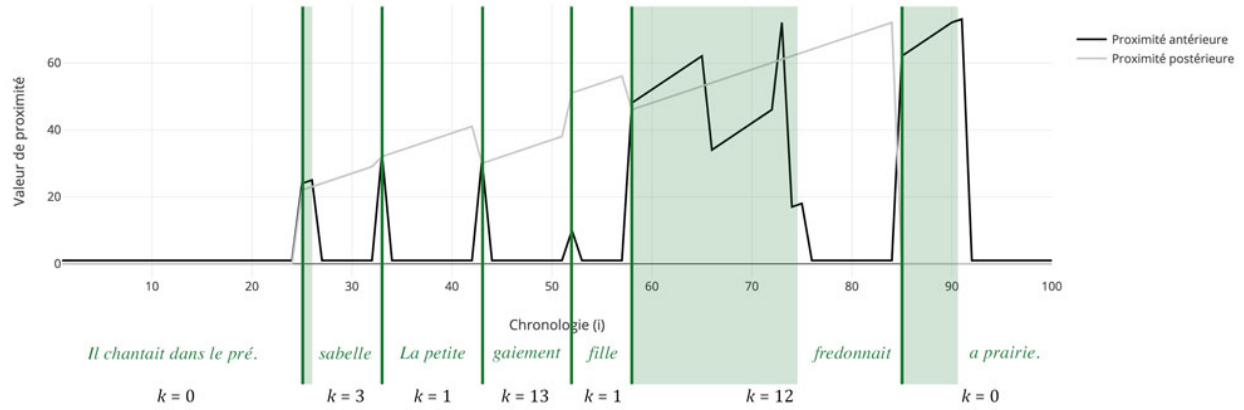

Figure 6.3 Courbes de proximité chronologique du fichier *log* 6.2 avec information superposée

On peut confirmer que les frappes 58 à 75 ont été exécutées sans que le curseur soit repositionné. Cette anomalie de la courbe de proximité antérieure peut signifier qu'un épisode de suppression débute avec la frappe 58 et que le texte supprimé a été produit dans plus d'une phase d'écriture. Puisque le curseur n'est pas repositionné, les caractères insérés lors de ces phases d'écriture sont spatialement consécutifs.

Afin de corroborer cette explication, reprenons la *visualisation progressive*. La figure 6.4 présente la *visualisation progressive* du fichier *log* 6.2. Dans cette figure, les frappes 58 à 75 sont encerclées en vert.

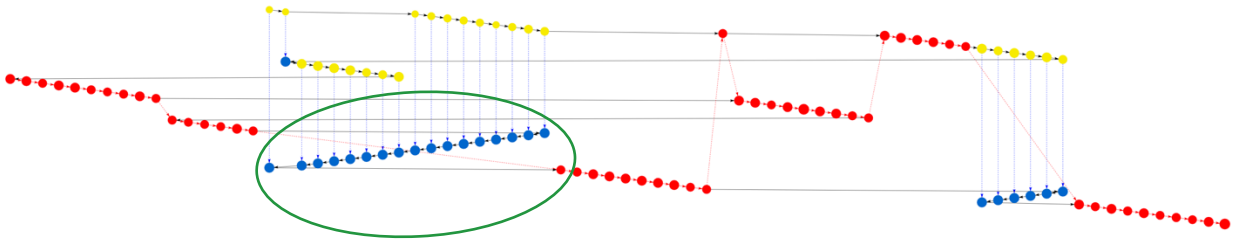

Figure 6.4 *Visualisation progressive* du fichier *log* 6.2

Les caractères supprimés par les frappes 58 à 74 ont bien été insérés dans deux phases d'écriture différentes. Les frappes 58 à 66 suppriment les insertions 3 à 11 tandis que les frappes 67 à 73 suppriment les insertions 26 à 32. Finalement, la frappe 74 supprime l'insertion 1. Notons que les frappes 1 et 3 à 11 ont été insérées lors de la même phase d'écriture.

On remarque aussi que cet épisode de suppression survient au milieu du texte en cours d'écriture. Plus spécifiquement, cet épisode s'arrête avant la frappe 57. Par la suite, l'insertion 75 est donc immédiatement inséré après la frappe 57 dans l'*AvantTexte*.

Dans le graphe *sans pertes*, si une frappe i est exécutée spatialement après une autre frappe j, alors le nœud j est nécessairement un prédécesseur du nœud i. Ceci explique pourquoi les valeurs de proximité antérieure des nœuds 74 et 75 sont plus faibles que les valeurs des nœuds 58 à 73.

Afin de présenter les différences de la proximité chronologique par rapport aux visualisations précédentes, reprenons maintenant le *SIG*. La figure 6.5 illustre le fichier *log* 6.2 grâce au *SIG*. Si nécessaire, le lecteur peut se référer au Chapitre 3, qui présente en détails comment interpréter le *SIG* ainsi que toutes les autres visualisations du processus d'écriture. Soulignons que dans la figure 6.5, l'axe des abscisses représente la chronologie de chaque frappe et non pas sa temporalité comme dans la version originale du *SIG*. Ce choix a été posé pour que le lecteur puisse comparer plus facilement le *SIG* aux mesures de proximité chronologique.

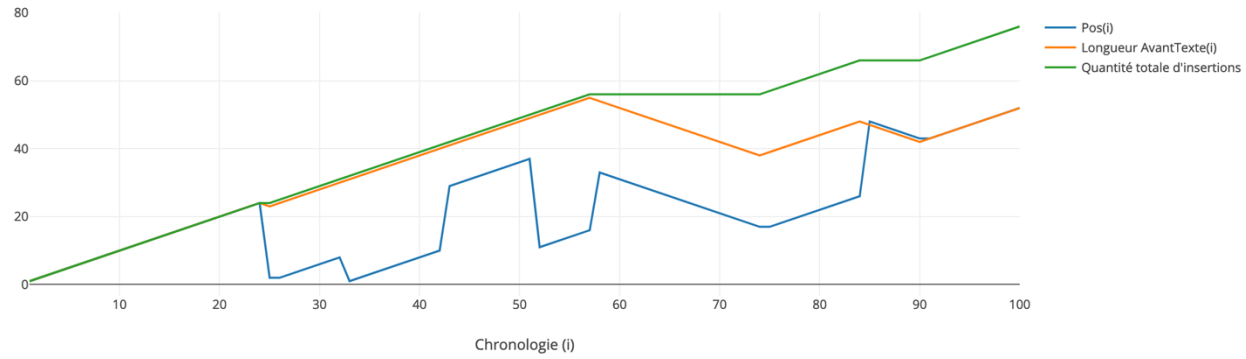

Figure 6.5 *SIG* du fichier *log* 6.2

Dans cette visualisation, on observe que la frappe 58 consiste en un repositionnement du curseur. Il est également possible de déduire que les frappes 58 à 74 sont de type S et que la frappe 75 est de type *I*. Autrement, aucune tendance particulière ne se dégage des frappes 58 à 75. Par rapport au *SIG*, les valeurs de proximité chronologique apportent donc une nouvelle information.

Reprenons maintenant la *représentation par les graphes* à la figure 6.6.

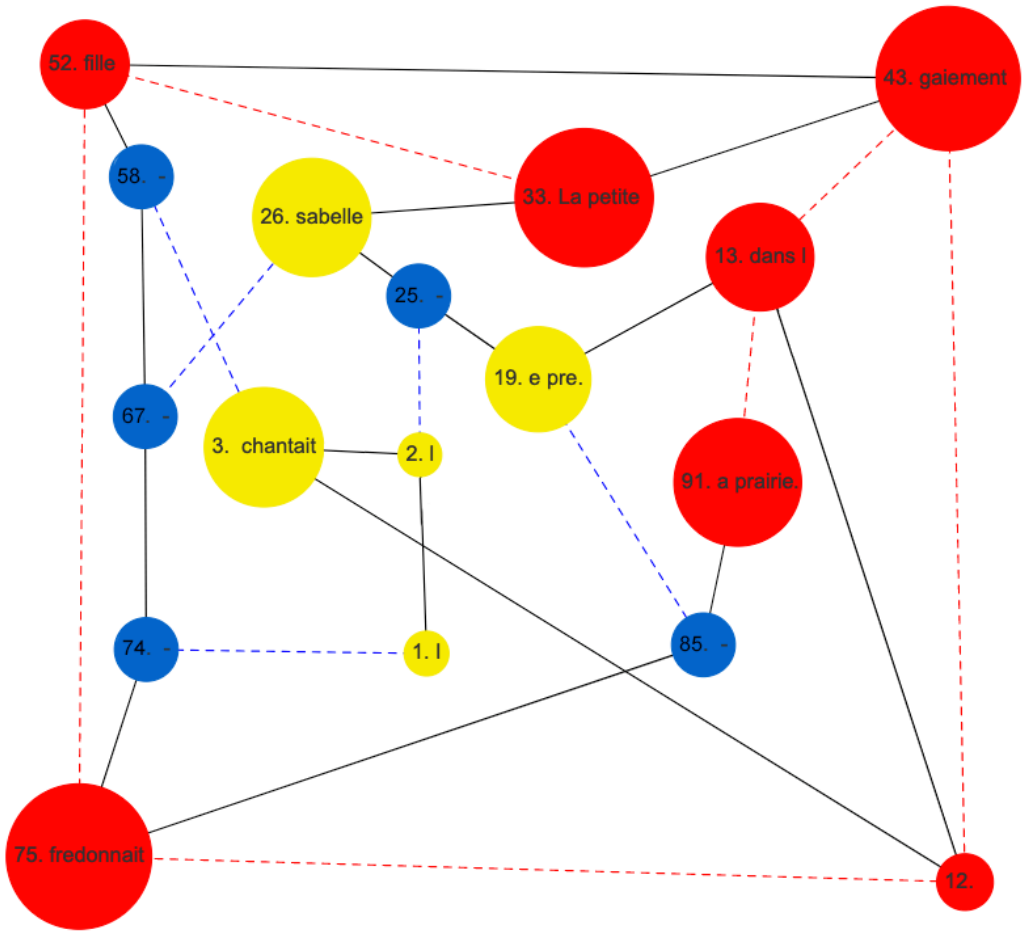

Figure 6.6 *Représentation par les graphes* du fichier *log* 6.2

Dans cette visualisation, on observe que plusieurs opérations de révision ont été effectuées. Toutes les suppressions ont été effectuées dans une phase d'écriture différente que les insertions qu'elles suppriment, ce qui est indiqué par le trait pointillé entre chaque paire de nœuds jaune et bleu. Rien n'indique à quel endroit dans le texte ces insertions ont été effectuées ni où elles étaient lorsqu'elles ont été supprimées. Bien que les frappes 58 à 74 suppriment du texte qui a été rédigé lors de deux phases d'écriture différentes, cette visualisation ne montre pas si le curseur a été repositionné ou non. Il en va de même pour la frappe 75 qui consiste en une insertion. En se fiant uniquement à la

représentation par les graphes, il n'est pas possible d'identifier s'il y a eu, ou non, un repositionnement du curseur entre les frappes 74 et 75.

Dans le contexte où rien n'indique que les frappes 58 à 75 sont dignes d'intérêt, seule une analyse approfondie d'un processus d'écriture, en utilisant plus d'une visualisation, permettrait de mettre en lumière cet ensemble de motifs complexes. Puisque les mesures de proximité chronologique ont permis d'identifier clairement et rapidement cette anomalie, elles consistent en un outil complémentaire de l'étude du processus d'écriture.

Le but de cet exemple était de démontrer que l'information fournie par les mesures de proximité chronologique est nouvelle. Ces mesures, lorsque représentées sous la forme de diagramme à courbes, permettent notamment de déduire en un coup d'œil des tendances qui sont autrement plutôt difficiles à trouver. Cet avantage rend les mesures de proximité chronologique pertinentes et complémentaires aux outils existants d'analyse du processus d'écriture.

6.3 Conclusion

Les mesures de proximité chronologique sont issues du modèle *sans pertes*. Ces nouvelles mesures permettent d'avoir accès à une information qui, sans être traitée de la sorte, est autrement difficile à obtenir sans une analyse approfondie d'un processus d'écriture.

Grâce aux mesures de proximité chronologique, il a été démontré que très rapidement, il est possible d'extraire de l'information pertinente avec le graphe *sans pertes*. La même information serait bien entendue calculable à partir du fichier *log* directement. Par contre, il est beaucoup plus facile de faire ces calculs avec le graphe *sans pertes*. L'avantage du graphe *sans pertes* est donc de donner un accès direct à l'information. Rappelons que l'objectif de la thèse est de modéliser le fichier *log* de manière à extraire de l'information et à faciliter le repérage de motifs complexes dans les données. Le graphe *sans pertes* ainsi que les mesures de proximité chronologique qui découlent de celui-ci répondent donc à cet objectif.

Dans ce chapitre, le potentiel des mesures de proximité chronologique a été présenté. Par contre, leur exploitation pourrait être poussée de plusieurs manières.

Puisqu'il s'agit de mesures et que celles-ci sont calculées pour chaque frappe d'un même fichier *log*, il serait intéressant d'appliquer des statistiques diverses comme le rang centile par exemple. Celui-ci, étant une mesure de dispersion, permettrait l'analyse sommaire des valeurs d'un fichier *log*. Ceci pourrait de plus faciliter la comparaison rapide entre les valeurs de plusieurs fichiers *log*.

Finalement, il serait aussi intéressant d'utiliser les mesures de proximité chronologique pour modéliser le processus d'écriture d'une toute nouvelle manière. Ceci pourrait être fait en créant un outil de visualisation qui exploite ces mesures.

Le prochain et dernier chapitre consiste en la conclusion de cette thèse.

CHAPITRE 7 CONCLUSION ET RECOMMANDATIONS

Le processus d'écriture est complexe. Bien qu'il soit possible de l'enregistrer dans un fichier pour avoir accès à une liste détaillée des frappes exécutées et de leurs caractéristiques, ces données peuvent difficilement être interprétées lorsqu'elles ne sont pas traitées. Il existe certes plusieurs outils et techniques qui visent à aider le chercheur dans son analyse en agrégeant ou en transformant ces données. Par contre, ces outils et techniques peuvent être améliorés. Deux opportunités d'amélioration ont été trouvées dans la revue de littérature. D'abord aucune visualisation ne représente l'ensemble des variables du processus d'écriture. Ensuite, il est certainement possible de trouver de nouveaux moyens pour extraire de l'information et trouver des motifs complexes dans les données.

Le travail de recherche effectué dans cette thèse visait à modéliser le fichier *log* pour en extraire de l'information. Ces informations permettent de faciliter le travail d'analyse du processus d'écriture et prennent souvent la forme de motifs complexes.

Pour répondre à cet objectif, deux nouveaux modèles du processus d'écriture ont été présentés :

- la *visualisation progressive*;
- le modèle *sans pertes*.

Un outil d'analyse calculé à partir du modèle *sans pertes*, les mesures de proximité chronologique, a également été présenté.

Cette thèse se conclut en résumant les principaux résultats. Leurs limites seront par la suite abordées. Finalement, les futurs développements qui seraient complémentaires aux travaux effectués dans cette thèse et qui permettraient de continuer à faire avancer l'analyse du processus d'écriture seront détaillés.

7.1 Synthèse des travaux

Dans un premier temps, la *visualisation progressive* a permis de synthétiser les principaux avantages des différentes visualisations existantes et de les combiner en une seule. Inspirée des *représentations linéaires*, des *SIG* et de la *représentation par les graphes*, elle est la seule

visualisation représentant l'ensemble des variables du processus d'écriture. Cette caractéristique fait en sorte que les chercheurs n'ont plus à recourir à plus d'une visualisation pour avoir accès à l'ensemble des variables et contribue certainement à simplifier le processus d'analyse, ce qui favorise la découverte de tendances complexes dans les données et atteint donc en ce sens un des objectifs de la thèse.

Le modèle *sans pertes* permet de structurer les données issues du fichier *log* de manière à faciliter la création d'outils d'analyse et de mesures. En effet il est possible d'utiliser des propriétés des graphes pour identifier des tendances dans les données. Ce modèle produit non seulement un graphe unique à chaque fichier *log* mais il contient aussi toutes les informations relatives au processus d'écriture et aux variations des *AvantTextes* dans le temps. Le graphe *sans pertes* a été créé initialement en tant que modèle, d'où il tire son nom, et non en tant que visualisation. Ses propriétés quantitatives intéressantes en font au départ davantage un modèle théorique. Cependant, toutes les propriétés de représentation et de présentation des données utilisées pour la *visualisation progressive*, et d'autres propriétés supplémentaires qui sont présentées dans le Chapitre 5, peuvent être appliquées sur le modèle *sans pertes*.

Finalement, la structure du modèle *sans pertes* a été exploitée pour créer les mesures de proximité chronologique. Pour chaque frappe d'un fichier *log*, deux mesures sont calculées : la proximité antérieure et la proximité postérieure. Ces mesures indiquent si une frappe *i* est chronologiquement proche ou loin des frappes qui l'entourent lorsqu'elle est exécutée. La proximité antérieure permet de situer la frappe *i* par rapport à la frappe *j* qui la précède dans l'*AvantTexte*. La proximité postérieure permet de situer la frappe *i* par rapport à la frappe *k* qui la suit dans l'*AvantTexte*.

Lorsqu'elles sont représentées sous la forme de diagramme à courbes, les mesures de proximité chronologique permettent de déduire en un coup d'œil des motifs complexes qui sont autrement plutôt difficiles à trouver. En ce sens, l'information fournie par ces mesures est donc nouvelle.

Les mesures de proximité chronologique peuvent être recalculées à partir de la *visualisation progressive*. Ce calcul est similaire à la procédure pour retrouver les *AvantTextes* dans cette visualisation. Plusieurs étapes sont nécessaires pour obtenir les valeurs de proximité chronologique de cette manière. Il est donc plus compliqué de retrouver ces valeurs par la *visualisation progressive* que par le graphe *sans pertes*.

Évidemment, les mesures de proximité chronologique pourraient aussi être calculées directement par le fichier *log*. Dans ce cas, il faudrait retraiter le fichier au complet pour d'abord retracer les *AvantTextes.* Par la suite, il faudrait retracer les caractères qui précèdent et suivent chaque frappe *i* lorsque cette dernière est exécutée. Ce calcul n'est pas intuitif. De plus, le traitement de l'information nécessaire pour calculer les mesures reviendrait, à peu de choses près, à modéliser l'information du fichier *log*.

Il a été démontré que la *visualisation progressive* et le modèle *sans pertes* ont la propriété de ne perdre aucune information. Ces deux modèles encodent et présentent l'information de manière différente. Dans les deux cas, ceci permet des analyses qui sont elles aussi différentes. En fonction de ce qu'on veut retrouver, certains modèles vont être plus faciles à utiliser que d'autres. Par exemple, la *visualisation progressive* permet de mieux trouver insertions supprimées et de les contextualiser par rapport aux frappes qui les suppriment. Cette visualisation possède tous les avantages des trois types de visualisations du processus utilisés. Elle est donc pertinente pour réaliser l'ensemble des types d'analyse qui auraient autrement été réalisées en utilisant la *représentation linéaire*, le *SIG* ou la *représentation par les graphes*.

Le modèle *sans pertes* tire sa pertinence du fait qu'il modélise le fichier *log* de manière à rendre l'information facilement accessible. Les mesures de proximité chronologique en sont un exemple. De par sa structure de graphe, il permet aussi de visualiser l'information. Ceci est certainement un atout pour aider l'analyse et la compréhension du processus d'écriture.

Il existe déjà plusieurs outils qui modélisent le fichier *log* et bon nombre de chercheurs utilisent une combinaison de ceux-ci pour réaliser leurs analyses. Les outils développés dans cette thèse visent à être complémentaires et à faciliter la découverte de tendances dans les données d'écriture.

7.2 Limites

Le but de cette section est de déterminer les limites de cette recherche. Les principaux axes de ces limites sont l'exploitation de la notion de temps, le fichier *log* et ses opérations d'écriture réelles et finalement les données utilisées.

7.2.1 Notion de temps dans le graphe

Le temps est un facteur non négligeable de l'analyse du processus d'écriture. Plusieurs chercheurs ont d'ailleurs orienté une portion de leur recherche sur l'étude des pauses et la considèrent comme une opération au même titre que l'insertion et la suppression dans leur approche (Wengelin A. , 2006; Wengelin, et al., 2009; Leijten & Van Waes, 2013; Baaijen, Galbraith, & Glopper, 2012).

Au-delà des pauses, le temps est tout de même un facteur central du processus d'écriture. Le temps est représenté de façon claire dans la *visualisation progressive*. Par contre dans le modèle *sans pertes* et dans les mesures qui découlent de ce modèle, seule la chronologie est présente. Même si chaque nœud représentant une frappe contient l'attribut de temps dans le graphe *sans pertes*, cette valeur n'est pas exploitée dans cette forme.

7.2.2 Fichier *log* et opérations d'écriture réelles

En pratique, un scripteur n'utilise pas seulement les frappes insertion et suppression de façon individuelle sur son clavier. Les logiciels de traitement de texte modernes permettent entre autres de supprimer un ensemble de caractères en un geste ou encore de copier et coller des portions de textes qui proviennent autant du texte en cours de rédaction, que de sources externes.

Les différents modèles présentés dans cette thèse sont donc une simplification du processus authentique de rédaction.

7.2.3 Données utilisées

Les données utilisées dans le cadre de cette recherche ont été créées manuellement dans le but d'obtenir des exemples courts, représentatifs et ayant un certain niveau de complexité. Bien que ceci ait été utile pour présenter les différents modèles, l'utilisation de données réelles aurait été complémentaire.

7.3 Futures recherches

Cette thèse se conclut sur des possibilités de développement qui constituent des opportunités de recherche intéressantes pour l'étude du processus d'écriture. Plusieurs des limites soulignées ci-haut consistent également en des pistes de développement et les opportunités qu'elles représentent seront mentionnées.

7.3.1 Mise en contexte authentique d'écriture

La première avenue de développement est d'appliquer les visualisations et outils d'analyses développés dans cette thèse sur des fichiers *log* issus d'un corpus réel ayant déjà été analysé par des linguistes. Ceci permettrait de faire le rapprochement entre les structures mathématiques trouvées et les structures identifiées par les chercheurs en linguistique.

La mise en contexte authentique pour l'étude d'écriture fait aussi référence aux variables traitées. Comme il a été souligné dans les limites, la notion de temps est inhérente au processus d'écriture. Plusieurs options seraient possibles pour inclure le temps dans les mesures découlant du modèle *sans pertes*.

Pour reprendre une suggestion de Leblay et Caporossi (2015) et tel que mentionnée dans la Section 5.2.1.2, les pauses pourraient être intégrées dans le graphe *sans pertes* sous la forme d'un nœud.

7.3.2 Visualisation des données d'écriture

À moins d'être présentées dans un rapport ou en tant qu'image, les visualisations des données d'écriture en tant qu'outils sont nécessairement imbriquées dans des logiciels qui les génèrent.

Dans ce contexte, une bonne visualisation des données qui permet d'optimiser la compréhension de son utilisateur doit nécessairement passer par l'ergonomie globale du logiciel. Dans la mesure où les présentations et résultats d'analyses seraient présentés dans un logiciel, plusieurs autres propriétés de présentation et représentation des données facilitant la lecture et l'exploitation pourraient être offertes.

7.3.3 Analyse de l'évolution du contenu du texte

L'aspect au cœur de l'étude du processus d'écriture est bien évidemment le texte lui-même. La discipline de l'*Analyse du Langage Naturel* s'intéresse depuis les années 50 à traiter, trier et catégoriser les différents éléments du texte en utilisant des méthodes issues de la logique, du raisonnement probabiliste et plus récemment, des réseaux de neurones (Thanaki, 2017).

L'opportunité de développement la plus intéressante est de pouvoir incorporer une analyse du contenu du texte et pouvoir savoir avec exactitude si, par exemple, une suppression immédiate d'un seul caractère est une erreur orthographique ou grammaticale. Une telle implémentation permettrait

de contextualiser les motifs complexes trouvés et de quantifier avec plus de précision les stratégies d’écriture des scripteurs ou leur niveau de maîtrise d’une langue.

RÉFÉRENCES

Agarwal, U., & Singh, U. (2009). *Graph Theory.* Dew Delhi: Laxmi Publications.

Aggarwal, C. C., & Wang, H. (2010). Graph data management and mining: a survey of algorithms and applications. Dans C. C. Aggarwal, & H. Wang (Éd.), *Managing and mining graph data* (pp. 13-68). New York: Springer.

Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). *Network flows: Algorithms and applications.* Upper Saddle River: Prentice Hall.

Aigner, W., Miksch, S., Schumann, H., & Tominski, C. (2011). Visualization of Time-Oriented Data. *Human-Computer Interaction Series*. London: Springer.

Alamargot, D., & Lebrave, J.-L. (2009). The study of professional writing: A joint contribution from cognitive psychology and genetic criticism. *European Psychologist*.

Alamargot, D., Caporossi, G., Chesnet, D., & Ros, C. (2011). What makes a skilled writer? Working memory and audience awareness during text composition. *Learning and Individual Differences*(21), 505-516.

Almond, R., Deane, P., Quinlan, T., Wagner, M., & Sydorenko, T. (2012). *A preliminary analysis of keystroke log data from a timed writing task.* Princeton: ETS Research Report Series.

Baaijen, V. M., Galbraith, D., & Glopper, K. d. (2012). Keystroke analysis: reflections on procedures and measures. *Written Communication, 29*(3), 246-277.

Balakrishnan, R., & Ranganathan, K. (2012). *A textbook of graph theory* (éd. 2nd Edition). New York: Springer.

Bartolini, I., Ciaccia, P., Ntoutsi, I., Patella, M., & Theodoridis, Y. (2004). A unified and flexible framework for comparing simple and complex patterns. *European Conference on Principles of Data Mining and Knowledge Discovery* (pp. 496-499). Berlin: Springer.

Bartram, D. (1980). Comprehending spatial information: The relative efficiency of different methods of presenting information about bus routes. *Journal of Applied Psychology*, 103.

Bartram, L. (1997). Perceptual and interpretative properties of motion for information visualization. *Proceedings of the 1997 workshop on new paradigms in information visualization and manipulation*, 3-7.

Bécotte-Boutin, H.-S., Caporossi, G., & Hertz, A. (2015). The progressive visualization, a new tool for analyzing the writing process. *Cahiers du GERAD, 131*.

Bécotte-Boutin, H.-S., Caporossi, G., Hertz, A., & Leblay, C. (2016). Analyse automatique des données scripturales prétraitées par des outils de visualisation. *Congrès Mondial de Linguistique Française - CMLF 2016. 27.* Tours: SHS Web of Conferences.

Bécotte-Boutin, H.-S., Caporossi, G., Leblay, C., & Hertz, A. (2019). Writing and rewriting: Keystroke logging's colored numerical visualization. Dans K. Sullivan, & E. Lindgren (Éd.), *Observing writing: logging handwriting and computer keystrokes* (pp. 96-124). Leyde: Brill Academic Publishers.

Berthold, M. (2011). Bisociative Knowledge Discovery. *Advances in Intelligent Data Analysis X* (pp. 1-7). Porto: Springer.

Bethel, E., Prabhat, P., Byna, S., Rubel, O., Wu, K., & Wehner, M. (2013). Why high performance visual data analytics is both relevant and difficult. Burlingame, California, USA: Visualization and Data Analysis.

Blanchard, F. (2005). Visualisation et classification de données multidimensionnelles application aux images multicomposantes. Reims: Université de Reims Champagne Ardenne.

Bondy, A. J., & Murty, U. (1982). *Graph theory with applications.* New York: Elsevier Science Publishing Co.

Borkin, M. A., Bylinskii, Z., Kim, N. W., Bainbridge, C. M., Yeh, C. S., Borkin, D., & Oliva, A. (2016). Beyond memorability: Visualization recognition and recall. *IEEE transactions on visualization and computer graphics, 22*(1), 519-528.

Boyd, D., & Crawford, K. (2012). Critical questions for big data. *Information, Communication & Society, 15*(5), pp. 662-679.

Breetvelt, I., Van den Bergh, H., & Rijlaarsdam, G. (1994). Relations between Writing Processes and Text Quality : When and How. *Cognition and Instruction, 12*(2), 103-123.

Caporossi, G. (2015, juin 21). *GenoGraphiX.* Consulté le septembre 2016, sur https://www.gerad.ca/Gilles.Caporossi/ggx/GGX/GenoGraphiX.html

Caporossi, G., & Leblay, C. (2011). Online Writing Data Representation : A Graph Theory Approach. *International Symposium on Intelligent Data Analysis* (pp. 80-89). Berlin: Springer.

Caporossi, G., & Leblay, C. (2014). Outils de visualisation de données enregistrées. Dans C. Leblay, & G. Caporossi (Éd.), *Temps de l'écriture: enregistrements et représentations* (pp. 147-166). Louvain-la-Neuve: Academia-L'Harmattan.

Card, S., Mackinlay, J., & Shneiderman, B. (1999). *Readings in Information Visualization: Using Vision To Think.* San Francisco: Morgan Kaufmann.

Chun, R. (2017). Redundant Encoding in Data Visualizations: Assessing Perceptual Accuracy and Speed. *Visual Communication Quarterly, 24*(3), 135-148.

Cislaru, G. (2015). The process-product interface. Dans G. Cislaru (Éd.), *Writing at the crossroads.* Amsterdam: John Benjamins Publishing Company.

Cox, M., Ortmeier-Hooper, C., & Tirabassi, K. (2009). Teaching Writing for the "Real World": Community and Workplace Writing. *The English Journal, 98*(5), 72-80.

Daelemans, W., Berck, P., & Gillis, S. (1997). Data mining as a Method for Linguistic Analysis: Dutch Diminutives. *Dutch Diminutives, Folia Linguistica, XXXI/I -2*, pp. 57-75.

Deane, P. (2014). *Using writing process and product features to assess writing quality and explore how those features relate to other literacy tasks.* Princeton: ETS Research Report Series.

Deane, P., & Zhang, M. (2015). *Exploring the feasibility of using writing process features to assess text production skills.* Princeton: ETS Research Report Series.

Dehmer, M., & Basak, S. C. (2012). *Statistical and machine learning approaches for network analysis.* Somerset: John Wiley & Sons.

Doquet, C. (2003). *Étude Génétique de l'Écriture sur Traitement de Texte d'Élèves de Cours Moyen 2, Année 1995-1996.* Paris: Université Sorbonne nouvelle.

Doquet, C. (2014). Pour une approche linguistique de l'écriture enregistrée. Dans C. Leblay, & G. Caporossi (Éd.), *Temps de l'écriture: enregistrements et représentations* (pp. 21-42). Louvain-la-Neuve: Academia-L'Harmattan.

Doquet, C., & Leblay, C. (2014). Temporalité de l'écriture et génétique textuelle : Vers un autre métalangage. *4e Congrès Mondial de Linguistique Française. 8.* Berlin: SHS Web of Conférences.

Dzemyda, G., Kurasova, O., & Zilinskas, J. (2013). Multidimensional Data Visualization: Methods and Applications. *Springer Optimization and Its Applications 75*. Vilnius: Springer.

Foucambert, D. (1995, mars). Les logiciels et leur usage. *Actes de lecture*, 1-5.

Foucambert, D., & Foucambert, J. (2014). Gestes d'écriture et caractéristiques linguistiques des textes achevés. Dans C. Leblay, & G. Caporossi (Éd.), *Temps de l'écriture: enregistrements et représentations* (pp. 43-70). Louvain-la-Neuve: Academia-L'Harmattan.

Fournier, J.-C. (2010). *Théorie des graphes et applications.* Paris: Lavoisier.

Grabowski, J. (2008). The internal structure of university students' keyboard skills. *Journal of Writing Research*, 27-52.

Gross, J., & Yellen, J. (1999). *Graph theory and its applications.* Boca Raton: CRC Press.

Gross, J., Yellen, J., & Zhang, P. (2015). *Handbook of Graph Theory* (éd. 2nd Edition). Boca Raton: Chapman and Hall/CRC.

Harary, F. (1969). *Graph Theory.* Reading: Addison-Wesley Publishing Company.

Heer, J., & Bostock, M. (2010). Crowdsourcing graphical perception: using mechanical turk to assess visualization design. *SIGCHI conference on human factors in computing systems* (pp. 203-212). Atlanta: ACM.

Henley, E. J., & Williams, R. (1973). *Graph theory in modern engineering.* Houston: Academic Press.

Inputlog. (2009, 09 25). *Knowledgebase, Fact Sheet.* Consulté le 07 31, 2018, sur http://www.inputlog.net/docs/graphical%20representation.xls

ITEM. (2014, mai 7). *Enjeux de recherche*. Consulté le juin 14, 2014, sur ITEM: http://www.item.ens.fr/index.php?identifier=l-item

Kirk, A. (2012). *Data visualization: a successful design process.* Birmingham: Packt Publishing.

Klein, J. (2014). *Interdisciplining Digital Humanities: Boundary Work in an Emerging Field.* Ann Arbor: University of Michigan Press.

Kollberg, P. (1996). *Rules for the S-notation: a computer-based method for representing revisions.* Stockholm, Sweden: IPLab, Royal Institute of Technology (KTH).

Kollberg, P. (1997). S-notation as a tool for analysing the episodic nature of revisions. Barcelona: European Writing Conferences.

Latif, M. M. (2008). A State-of-the-Art Review of the Real-Time Computer-Aided Study of the Writing Process. *International Journal of English Studies, 8*(1), 29-50.

Leblay, C. (2009). La question du «déjà écrit» dans le processus d'écriture observé en temps réel. Une contribution de la génétique à la didactique. *Modèles linguistiques, 59*, 153-176.

Leblay, C. (2011, novembre 12). *Le Temps de l'Écriture. Genèse, durée, représentations.* Consulté le 12 15, 2013, sur University of Jyväskylä: http://urn.fi/URN:ISBN:978-951-39-4519-0

Leblay, C. (2016). Génétique textuelle et écritures mono- et plurilingues. *TTR : traduction, terminologie, rédaction, 29*(1), 33-59.

Leblay, C., & Caporossi, G. (2014). Introduction aux données temporelles de l'écriture. Dans C. Leblay, & G. Caporossi (Éd.), *Temps de l'écriture: enregistrements et représentations* (pp. 5-15). Louvain-la-Neuve: Academia.

Leblay, C., & Caporossi, G. (2015). A graph theory approach to online writing data visualization. Dans *Writing at the crossroads : The process-product interface* (pp. 171-181). Amsterdam: John Benjamins Publishing Company.

Lebrave, J.-L. (2001). Comment écriront-ils? *Diogène, 196*(4), 163-171.

Lebrave, J.-L., & Grésillon, A. (2009, 03 23). *Linguistique et génétique des textes : un décalogue.* Consulté le 12 14, 2013, sur Item: http://item.ens.fr/index.php?id=434571

Leijten, M., & Van Waes, L. (2006). Inputlog: New perspectives on the logging of on-line writing. *Studies in writing*(18), 73-94.

Leijten, M., & Van Waes, L. (2013). Keystroke Logging in Writing Research: Using Inputlog to Analyze and Visualize Writing Processes. *Written Communication, 30*(3), 358-392.

Lindgren, E., & Sullivan, K. P. (2002). The LS Graph : A Methodology for Visualizing Writing Revision. *Language Learning, 52*(3), 565-595.

Lindgren, E., Sullivan, K. P., Lindgren, U., & Spelman Miller, K. (2007). GIS for Writing: Applying Geographical Information Systems Techniques to Data Mine Writings' Cognitive Processes. Dans G. Rijlaarsdam, D. Galbraith, M. Torrance, & L. van Waes, *Writing and Cognition* (pp. 83-96). Amsterdam: Elsevier.

Lynch, C. (2008). How do your data grow? *Nature, 455*(4), 28-29.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Hung Byers, A. (2011). *Big data: the next frontier for innovation, competition, and productivity.* New York: McKinsey Global Institute.

Mellet, S. (2003, Décembre 15). *Corpus et recherches linguistiques.* Consulté le Juillet 26, 2018, sur http://corpus.revues.org/7

Midgette, E., Haria, P., & MacArthur, C. (2008). The effects of content and audience awareness goals for revision on the persuasive essays of fifth- and eighth-grade students. *Reading and Writing, 21*, 131-151.

Minelli, M., Chambers, M., & Dhiraj, A. (2013). *Big data, big analytics : Emerging business intelligence and analytic trends for today's businesses.* Hoboken: Wiley.

Moller, T., Hamann, B., & Russell, R. (2009). Preface. Dans T. Moller, B. Hamann, R. Russell, T. Moller, & R. Russell (Éd.), *Mathematical foundation of scientific visualization, computer graphics, and massive data exploration* (pp. V-VII). Berlin: Springer.

Nyhan, J., & Flinn, A. (2016). *Computation and the Humanities. Towards an Oral History of Digital Humanities.* (J. Nyhan, & A. Flinn, Éd.) Springer International Publishing.

Ohlhorst, F. J. (2012). *Big Data Analytics : Turning Big Data into Big Money.* Hoboken: John Wiley & Sons.

Perrin, D. (2003). Progression Analysis (PA): Investigating Writing Strategies at the Workplace. *Journal of Pragmatics, 35*, 907-921.

Perrin, D. (2019). Progression analysis: Working with large data corpora in field research on writing. Dans K. Sullivan, & E. Lindgren (Éd.), *Observing writing: logging handwriting and computer keystrokes* (pp. 143-162). Leyde: Brill Academic Publishers.

Plane, S., Alamargot, D., & Lebrave, J.-L. (2010). Temporalité de l'écriture et rôle du texte produit dans l'activité rédactionnelle. *Langages, 177*(1), pp. 11-32.

Saha Ray, S. (2013). *Graph theory with algorithms and its applications in applied science and technology.* Rourkela, India: Springer India.

Southavilay, V., Yacef, K., Reimann, P., & Calvo, R. A. (2013). Analysis of Collaborative Writing Processes Using Revision Maps and Probabilistic Topic Models. *Proceedings of the Third International Conference on Learning Analytics and Knowledge*, 38-47.

Stromqvist, S., Holmqvist, K., Johansson, V., & Karlsson, H. (2006). What keystroke-logging can reveal about writing. Dans K. Sullivan, & E. Lindgren (Éd.), *Computer keystroke logging and writing: methods and applications* (Vol. 18, pp. 45-72). Oxford: Elsevier.

Sullivan, K. P., & Lindgren, E. (2014). La révision en production écrite enregistrée. Dans C. Leblay, & G. Caporossi (Éd.), *Temps de l'écriture: enregistrements et représentations* (pp. 71-92). Louvain-la-Neuve: Academia.

Thanaki, J. (2017). *Python natural language processing: explore NLP with machine learning and deep learning techniques.* Birmingham: Packt Publishing.

Tory, M., & Moller, T. (2004). Human factors in visualization research. *IEEE Transactions on visualization and computer graphics, 10*(1), 72-84.

Tufte, E. R. (2001). *The visual display of quantitative information* (éd. 2e). Cheshire: Graphics Press.

Unwin, A., Chen, C.-h., & Hardle, W. K. (2008). Introduction. Dans A. Unwin, C.-h. Chen, W. K. Hardle, C.-h. Chen, W. K. Hardle, & A. Unwin (Éd.), *Handbook of Data Visualization* (pp. 3-14). Berlin: Springer.

Van Gelderen, A., & Oostdam, R. (2004). Revision of form and meaning in learning to write comprehensible text. Dans *Revision Cognitive and Instructional Processes* (pp. 103-123). Dordrecht: Springer Netherlands.

Van Horenbeeck, E., Pauwaert, T., Van Waes, L., & Leijten, M. (2012). *S-notation: S-notation markup rules.* Antwerp: University of Antwerp.

Van Waes, L., & Leijten, M. (2015). Fluency in Writing: A Multidimensional Perspective on Writing Fluency Applied to L1 and L2. *Computers and Composition*, 79-95.

Van Waes, L., & Schellens, P. J. (2003). Writing Profiles : The Effect of the Writing Mode on Pausing and Revision Patterns of Experienced Writers. *Journal of Pragmatics, 35*, 829-853.

Van Waes, L., Leijten, M., Wengelin, A., & Lindgren, E. (2012). Logging Tools to Study Digital Writing Processes. Dans V. Berninger (Éd.), *Past, present, and future contributions of cognitive writing research to cognitive psychology* (pp. 507-533). New York: Psychology Press.

Van Waes, L., Van Weijen, D., & Leijten, M. (2014). Learning to write in an online writing center: The effect of learning styles on the writing process. *Computers and Education, 73*, 60-71.

Ware, C. (2004). Chapter 5 - Visual attention and information that pops out. Dans C. Ware, *Information Visualization: perception for design* (pp. 145-186). Amsterdam: Elsevier.

Wengelin, A. (2006). Examining Pauses in Writing: Theory, Methods and Empirical Data. Dans K. Sullivan, & E. Lindgren (Éd.), *Computer keystroke logging and writing: methods and applications* (Vol. 18, pp. 107-130). Oxford: Elsevier.

Wengelin, A. (2014). Temps et pauses dans l'écriture au clavier. Dans C. Leblay, G. Caporossi, C. Leblay, & G. Caporossi (Éd.), *Temps de l'écriture: enregistrements et représentations* (pp. 97-124). Louvain-la-Neuve: Academia.

Wengelin, A., Torrance, M., Holmwvist, K., Simpson, S., Galbraith, D., Johansson, V., & Johansson, R. (2009). Combined eyetracking and keystroke-logging methods for studying cognitive processes in text production. *Behavior Research Methods, 41*(2), 337-351.

Yau, N. (2011). *Visualize this: the flowing data guide to design, visualization and statistics.* Indianapolis: Wiley Publishing.