Institut National des Langues et Civilisations Orientales

Département Textes, Informatique, Multilinguisme

Détection de contenu utile depuis des sites d'actualité

MASTER

TRAITEMENT AUTOMATIQUE DES LANGUES

Parcours :

Traitement Automatique des Langues / Ingénierie Multilingue

par

Kévin DETURCK

Encadrement par l'école :

Damien Nouvel

Encadrement par l'entreprise :

Nicolas Heulot, Maxime Maillot

Année universitaire 2014/2015

RÉSUMÉ

Dans un contexte de veille sur des sites d'actualité, la détection automatique du contenu rédigé par les journalistes est nécessaire au bon déroulement de traitements linguistiques automatisés. Nous adressons ce sujet en partant du constat que sur un même site d'actualité c'est principalement le contenu textuel porteur d'information qui change d'une page à l'autre. Plus généralement, nous définissons le contenu utile comme le contenu textuel qui diffère entre deux pages internet du même site. Nous utilisons différentes heuristiques et une comparaison par distance d'édition entre arbres HTML de mêmes structures pour déterminer ce contenu utile. Nous montrons que notre approche améliore les résultats par rapport à un outil de référence sur un corpus composé de pages récentes issues de différents sites d'actualité.

Table des matières

CHAPITRE 1

AVANT-PROPOS

Ce travail de mémoire s'inscrit dans le cadre de la fin du cursus de Master "Traitement Automatique des Langues" (TAL) spécialité "Ingénierie" promotion 2015 à l'Institut National des Langues et Civilisations Orientales (INaLCO). Cette formation vise à former des techniciens de la langue, capables de composer avec les grandes thématiques linguistiques à l'aide d'outils d'analyse adaptés voire empruntés à l'informatique. La spécialité "Ingénierie" mène à la conception de tels outils par l'apprentissage de la programmation, du code informatique, tout comme de théories mathématiques, d'algorithmes, toujours dans le but de répondre aux problèmes posés par des faits de langue. Je me suis dirigé dans ce domaine parce que j'aime l'idée que les machines traitent des problématiques issues de l'humain. Par exemple, il est possible d'utiliser un ordinateur pour déterminer l'opinion que délivre une personne à travers un texte. Le fait de vouloir automatiser des traitements linguistiques nous met face à une complexité que nous ne soupçonnons même pas en parlant une langue au quotidien.

Comme Zola au XIXe siècle dans sa conception du Naturalisme, j'aime partir de l'observation des éléments les plus triviaux pour ensuite conceptualiser et comprendre. À cet égard, je m'intéresse avant tout à l'étude de données concrètes, ce que propose la linguistique. Je prends l'informatique comme l'outil qui me permet d'expérimenter et d'en extraire des observations. Ce sont ces observations qui doivent servir à une conceptualisation. Je favorise donc les approches de réflexion allant du bas vers le haut plutôt que le contraire qui a tendance à fondre le concret dans une théorie faite par avance.

Je sors maintenant du cadre pédagogique pour me confronter, dans la nature, à une catégorie de problèmes pour lesquels on m'a formé. Ce sont maintenant de véritables enjeux qui me sont proposés dans un cadre professionnel. Je dois user de réflexion et de méthode pour mettre en place une approche qui pourrait permettre de contribuer à la recherche dans le domaine du TAL.
Ce document est capital puisqu'il est l'occasion d'une prise de recul sur l'ensemble d'un cursus universitaire. Le bagage de connaissances acquis durant la formation me permet maintenant de faire preuve d'assez d'esprit critique pour choisir mon chemin dans un domaine aussi vaste que le TAL, puis de développer de l'autonomie face à de vrais problèmes rencontrés en contexte professionnel. C'est avant tout un travail de recherche avec une manière de procéder qui se doit d'être rigoureuse tout en se laissant parfois porter par ses intuitions. C'est l'occasion ici de proposer des idées innovantes pour répondre à des questions découlant de la problématique du mémoire.

La définition de ce mémoire est étroitement liée à la réalisation de mon stage de fin d'études, un stage de six mois réalisé à l'Institut de Recherche Technologique (IRT) SystemX dans le projet Intégration Multimédia Multilingue (IMM). C'est un projet ambitieux qui vise à structurer des connaissances à partir de ressources issues de l'internet. La collecte de pages internet comme la construction de la base de connaissances sur les entités nommées est automatisée. C'est une véritable chaine de traitement de documents destinée à faire de la veille sur l'internet et à créer de la connaissance à partir d'entités reconnues automatiquement. Cela nécessite l'intégration de multiples composants, aussi nombreux que les différentes facettes à regarder pour avoir la meilleure vue possible sur les ressources à traiter. Le cadre de l'IRT me permet de travailler sur des vraies problématiques de TAL dans un contexte de recherche tout à fait propice à la rédaction de ce mémoire.

Il s'agit donc de structurer de l'information à partir de contenus semi-structurés. Le travail que je réalise doit réellement servir le système en place et veut même l'améliorer. J'ai donc dû m'intégrer vraiment dans l'équipe en utilisant les outils déjà mis en place afin que mon travail s'inscrive au sein de l'existant. Je m'insère donc dans cette structure pour l'amélioration du nettoyage des pages internet collectées en vue de traitements linguistiques spécifiques. Ma tâche se situe donc en amont de la plus grande partie de la chaine, ce qui rend mon travail déterminant pour le résultat final des traitements. J'ai choisi ma problématique en lien direct avec la thématique de mon travail pour l'entreprise.

FIGURE 1.1 – Chaine de traitement sur le projet du stage

L'outil que je conçois se trouve dans cette chaine 1.1 en sortie d'un "Crawler" qui collecte sur l'internet des pages spécifiques à un contexte donné. La chaine pourrait en effet devoir répondre à la demande d'utilisateurs provenant d'univers très différents. Les sites exploités seront principalement consacrés à l'actualité. Ce projet est multilingue ; la langue des documents ne doit donc pas être un obstacle à leur analyse correcte.

La thématique de ce mémoire aura la particularité d'offrir une belle ouverture permettant de faire dialoguer des champs d'étude éclectiques. C'est aussi un défi de devoir s'accaparer des techniques spécifiques qui ne sont pas forcément faites au départ pour cohabiter, de réussir à les relier pour les rendre encore plus puissantes. J'ai constaté que les techniques qui semblent les plus intelligentes ne sont pas forcément les plus efficaces. Ce sont parfois les idées les plus simples qui apportent les meilleurs résultats.

Pour ce mémoire, je dois suivre une méthodologie scientifique propre. Il sera donc proposé un bilan sur ce qui a été fait pour notre sujet, telle une phase d'observation pour décider de ce qui peut être étudié dans le cadre de ce travail. Pour la suite, il sera question de l'approche développée dans ce mémoire qui nous mènera jusqu'à son évaluation. Ce travail de recherche se terminera sur des perspectives possibles qui pourront lui faire suite.

Le présent mémoire bénéficie d'un encadrement d'une part de l'INaLCO et d'autre part de l'IRT SystemX. Il est contraint dans sa forme par les consignes de l'école. Ce travail fait l'objet d'une évaluation avec une soutenance à l'école pour le porter. Le stage qui lui est associé donne aussi lieu à une note. Le résultat global compte pour l'obtention de mon Master. Ainsi, mon encadrement occupe une place particulièrement importante pour le bon déroulement général du stage et du mémoire. Je suis en communication régulière qui s'assure de la validité de mes travaux et du présent document selon les attentes de l'équipe pédagogique. Au quotidien, c'est l'équipe encadrante de l'entreprise qui me guide pour mener à bien mes travaux.

CHAPITRE 2

INTRODUCTION

Contexte

L'internet propose aujourd'hui un flux de données si volumineux et complexe que pouvoir le gérer est crucial. La publicité le submerge et les velléités d'interactivité avec l'utilisateur empêchent parfois l'accès à l'essentiel. Lorsque nous lisons une page internet, c'est pour y récupérer l'information qui nous intéresse. C'est la raison d'être des contenus produits et celle pour laquelle nous les lisons. Les éditeurs veulent quant à eux atteindre leur lecteur pour financer le contenu gratuit qu'ils proposent en y apposant des contenus externes comme de la publicité.

Des métiers comme ceux de la veille nécessitent d'analyser une grande quantité de documents sur internet pour faire de la prospection. Une société, pour faire de l'intelligence économique, demande à pouvoir étudier rapidement tout ce qui se dit dans son domaine à travers l'internet. Cela nécessite d'aller plus vite dans le traitement de masses d'information. Cette demande indique combien il est crucial de pouvoir automatiser la gestion de contenu depuis l'internet. L'informatique propose aujourd'hui des capacités de calcul et de mémoire appropriées aux enjeux que sont complexité et quantité. Les machines déterminent un résultat plus rapidement qu'une équipe d'humains quand il est question de traiter des données. La qualité revient à l'humain par sa configuration de la machine. Notre sujet a trait au domaine "Web Scrapping" qui touche non seulement le traitement du texte mais aussi plus largement le traitement des données en général. Le texte est une donnée à part entière.

Les sites journalistiques doivent produire toujours plus vite pour rester sur le marché. Les éditeurs de contenu, comme les journaux Le Figaro pour *lefigaro.fr* ou Libération pour *liberation.fr*, profitent de systèmes automatisant l'écriture de code HTML pour générer une base de page. C'est aussi plus confortable et apporte une cohérence assurée à travers un même site. Ces outils sont les "Content Management Systems" (CMS) qui, comme Wordpress [Wor, 2015] ou Drupal [Dru, 2015], sont fortement mis à profit aujourd'hui sur les plus grands sites. Cela permet ainsi de définir un cadre structurel que nous pouvons retrouver d'une page à l'autre d'un même site et pourquoi pas d'un site à l'autre. Ces outils automatisant l'édition tendent à uniformiser l'internet, ce qui permettrait de mieux l'appréhender pour y trouver une information. Toutefois, ce sont les développeurs qui choisissent la configuration des outils utilisés par les journalistes. La production des pages n'est donc pas totalement standardisée.

La ressource de notre travail est la page internet. Si un lecteur utilise un navigateur internet pour visualiser une page, c'est qu'il s'agit d'un outil pour interpréter le code avec lequel elle est écrite. Le langage HTML, pour "Hypertext Markup Language", est le langage de l'internet pour écrire une page. Une page internet est codée selon une norme qui lui permet d'être interprétée par un navigateur. Cette norme a la particularité d'être semi-structurée c'est-à-dire qu'elle définit une syntaxe qui ne nécessite pas une structuration régulière et complète. Cette structure se présente sous la forme d'une arborescence hiérarchisée, c'est l'arbre HTML. Chaque élément de cette structure est identifié par une balise. Les éléments peuvent être décrits par des attributs dans la balise ouvrante. Les ouverture et fermeture des balises ainsi que leur imbrication définissent une frontière pour le contenu des éléments permettant de dessiner la hiérarchie. L'éditeur se trouve relativement libre dans la composition pouvant par là même faire fi des conventions.
DOM pour "Document Object Model", est la représentation d'un arbre HTML obtenue par un certain calcul. C'est ce que les navigateurs utilisent pour fournir le rendu visuel d'une page. Le W3C a standardisé le DOM pour que les différents navigateurs unifient leur fonctionnement et ainsi éviter d'avoir des réactions différentes entre navigateurs pour une même page. Dans notre approche, nous n'utilisons pas le rendu de la page HTML par un quelconque navigateur. Nous restons attachés au code source de la page, c'est-à-dire le code HTML sans interprétation pour produire un arbre DOM. Nous parlerons donc d'arbre HTML.

D'importants obstacles s'opposent au traitement de notre sujet. C'est ce que nous allons voir.

Problématique

Il est ici question de définir un programme qui va automatiser la détection de la donnée recherchée par un lecteur sur une page d'actualité issue d'internet. Un problème est de juger ce qui est utile sur une page d'actualité. Nous sommes placés dans le cas d'un schéma communicationnel classique. Alors, un article de journal, y compris en ligne, est publié pour diffuser un message porteur d'information. L'émetteur est le journaliste qui crée le message et le récepteur est le lecteur de la page d'actualité. Le site internet, son éditeur et même le réseau tout entier constituent le canal de communication.

Les éléments de publicité peuvent alors se représenter comme le bruit inhérent au passage du message dans le site internet. C'est un bruit volontaire de la part de l'éditeur du site internet mais qui ne concerne pas l'émetteur de l'information dans notre schéma, c'est-à-dire le journaliste. Nous pouvons remarquer à cet égard la tendance à une publicité ciblée sur internet afin qu'elle ne soit plus du bruit pour le lecteur.
De la même façon, le menu ou les éléments de navigation sont propres à l'édition du site mais ne concernent pas la diffusion de l'information à proprement parler. Ce ne sont pas des objets appartenant au message. Ce sont autant d'éléments qui peuvent perturber la communication entre un journaliste et son lecteur.
Les commentaires sont un cas intéressant parce que ce sont des messages comme l'article. Ils contiennent eux aussi une information. Nous choisissons de les éluder, de façon objective, parce qu'ils gravitent autour du texte de l'article. Dans l'organisation

d'une page d'actualité, les commentaires ne seraient pas présents s'il n'y avait pas un texte d'article. C'est ici un critère d'indépendance qui intervient. Le corps de l'article peut exister seul, pas les autres contenus.

Nous sommes dans le cadre d'une chaine de traitement destinée à l'analyse de l'information essentielle d'une page d'actualité. Le contenu utile est donc ce qui appartient au message du journaliste et rien d'autre.
Notre objectif est de retrouver le texte exact contenu dans le corps de l'article.

Comment un système peut-il reconnaitre automatiquement le contenu utile d'une page internet ?

Dans la structure HTML d'une page internet, le problème est d'identifier l'élément minimal porteur du contenu utile, soit le nœud le plus bas dans l'arbre HTML qui contient tout le contenu utile. L'accès au contenu utile d'une page internet n'est pas évident pour une machine pour les deux raisons principales suivantes :

— Le caractère semi-structuré de l'HTML est source d'une variabilité qui peut bruiter la réception du contenu. Rien n'oblige un éditeur à toujours mettre le contenu d'un article dans une même balise à travers les pages d'un site.
— Le contenu utile est du texte en langage naturel, fondamentalement ambigu et donc non trivial pour une machine.

En HTML, les balises ne permettent pas de distinguer de façon assez précise les constituants d'une page internet. La balise "div" et la balise "p" peuvent parfois être employée indifféremment justement parce qu'elles ne proposent pas une sémantique précise au regard d'une fonction dans le rendu de la page, du point de vue d'un lecteur.

Nous travaillons sur les sites de journaux en ligne avec un contenu réparti en différentes natures et thématiques pouvant donner lieu à autant de "templates" différents. Par exemple, sur *lefigaro.fr* les articles traitant de sport ne sont pas construits comme les tribunes politiques. Nous ne devons donc pas sous-estimer cette variabilité qui est un enjeu fort pour notre problème. Il est possible à cet égard de voir ce dernier de deux façons différentes :

— Repérer le contenu qui n'est pas singulier à la page internet analysée, c'est-à-dire propre à l'article de la page, de manière à le retirer comme un procédé typique de nettoyage.
— Cibler directement le contenu utile pour l'extraction.

Nous n'avons pas de vue sur les données à traiter. Nous devons penser notre approche pour qu'elle soit suffisamment générale dans l'optique de la variété et même de l'évolutivité des pages internet. Notre système pourrait donc rencontrer des cas auxquels nous n'aurions même pas pensé au moment de sa conception. A l'inverse, la généralisation ne permet pas de traiter certains cas particuliers. Nous voyons ici qu'il s'agit de trouver un juste équilibre entre spécificité et généricité. L'adéquation entre ces paramètres est fondamentale dans la manière de penser notre approche. Le choix peut dépendre du contexte d'utilisation. Il est possible de ne vouloir traiter que certains cas bien identifiés et non la totalité d'un corpus. Sinon englober le plus de cas possibles rend le système plus général mais cela complexifie par la même son paramétrage.

En particulier, voici les propriétés que notre système doit vérifier :

— spécifique au genre des sites d'actualité
— adaptable à la variété potentielle des cas rencontrés
— indépendant de l'intervention humaine.
— robuste à la taille des structures.

Notre volonté d'identifier le contenu utile revient à se demander comment identifier une donnée particulière dans un champ de données semi-structurées. Nous voulons pouvoir traiter une vaste quantité de pages pour que notre système soit vraiment rentable. À chaque nouvelle page, il doit comprendre la ressource de manière à pouvoir récupérer les indices qui lui serviront à tracer son chemin dans un arbre donné. Quand nous parlons de "compréhension", c'est pouvoir aborder la sémantique des différentes parties de la page :

— le titre
— le corps de l'article
— les images
— les commentaires
— la publicité
— les liens de partage/liens vers les autres contenus (menus)

Cette problématique demande de produire automatiquement un choix puisqu'il s'agit d'opérer une sélection à travers les différents chemins possibles dans l'arbre HTML. Lorsque nous donnons au système une page pour analyse, il doit être capable de composer avec une structure arborescente, induisant la possibilité de suivre ou pas une branche dans l'arbre. Cela met en jeu la génération automatique d'un choix face à une situation qui n'est pas déterministe. À ce propos, nous pensons au concept d'"arbre de décision" qui décrit les conditions à remplir pour que le système puisse faire un choix devant plusieurs possibilités. Faire un choix nécessite des connaissances a priori. Pour l'arbre de décision, c'est la description des cas. La difficulté est la conception d'un système pouvant prendre en compte toutes les situations possibles. À cet égard, la solution la plus utilisée actuellement est l'apprentissage supervisé qui classe des données concrètes et annotées par l'humain afin que le système reconnaisse les situations existantes dans d'autres données. Un corpus d'entrainement représente ici la connaissance a priori du système. Cela nécessite un coût humain pour l'annotation des ressources et une quantité de données assez importante pour que le système obtienne de bons résultats.

Les questions posées, nous vous présentons brièvement l'approche que nous avons choisie afin d'y répondre.

Approche

[Gibson et al., 2005] mesure à 50% le volume de l'internet qui n'est pas véritablement porteur d'information. C'est le "template que [Gibson et al., 2005] pointe ici en le définissant comme les éléments de mise en forme, de navigation et de promotion. Ils ont utilisé des algorithmes d'extraction de"template" pour obtenir une proportion sur un corpus extrait. Ce corpus a été constitué en utilisant un crawler à grande échelle. 50 millions de sites et deux milliards de pages.

L'information essentielle d'une page est finalement bruitée par énormément d'éléments qui ne sont que du contexte. Nous pouvons rencontrer des liens vers les réseaux sociaux pour partager le contenu visité, des propositions d'abonnement, des archives autour du contenu utile, la biographie du journaliste ; les possibilités sont grandes. Ces éléments sont de nature à fourvoyer notre système dans son repérage du contenu utile. La définition de règles pour traiter systématiquement les documents est rendue difficile.
Nous pensons que la structure d'une page internet est pertinente pour identifier le contenu utile. Notre point de vue est d'analyser le contenant pour appréhender le contenu. C'est un point de vue empirique qui est lié à l'expérience de l'utilisateur humain sur l'internet. Nous supposons que la structure d'une page internet obéit à une cohérence sémantique qui doit permettre de la comprendre pour accéder au contenu utile. La lecture d'une nouvelle page par un humain suit un ordre logique assuré par la cohérence des blocs. Nous cherchons ici à capter une répétition dans la structuration qui met en œuvre cette cohérence.

Une nouvelle conception de l'internet est justement proposée pour que la machine puisse interpréter le contenu dans une page internet. C'est le Web sémantique qui requiert d'ajouter un nouveau balisage pour expliquer en langage formel ce que le contenu présente au lecteur. Des vocabulaires différents, nommés "ontologies", sont proposés selon la nature des objets à décrire. Par exemple FOAF [FOA, 2015] pour les personnes, Dublin Core [Dub, 2015] pour les documents et SKOS [SKO, 2015] pour les concepts. Elle met en avant le sens attribué aux éléments HTML pour que la machine accède plus facilement l'information proposée dans la page.
La norme HTML 5 va dans la même direction visant à sémantiser le contenu d'internet. Des nouvelles balises plus sémantiques sont ainsi proposées servant à décrire des éléments de structure particuliers dans la page suivant leur fonction respective. Nous pouvons citer la balise "article" qui désigne le contenu principal sur une page, sans les contenus connexes.
Rien n'oblige chaque site à utiliser ces normes.

Pour extraire le contenu utile d'une page internet, les méthodes répandues actuellement cherchent à se rendre le plus générique possible, prenant la page internet dépourvue de tout contexte pour juger de ce qui sera informatif ou pas. Par exemple :

— [Pasternack and Roth, 2009] cherche à classifier le texte.
— [Weninger et al., 2010] est hybride dans la mesure où il utilise à la fois de l'apprentissage et des heuristiques sur le texte comme sur les nœuds de la structure arborescente.
— [Sun et al., 2011] s'appuient sur la densité textuelles et des heuristiques empiriques pour déterminer ce qui est éventuellement pertinent.
— [Kohlschütter et al., 2010] procède par apprentissage sur les éléments de structure et de rendu visuel.

Pour identifier le contenu utile, nous avons choisi ce qui semble le plus intuitif : ce qui entoure la page internet, ce qui est dans son milieu le plus proche constitue une indication à exploiter pour déterminer ce qui est nouveau, en utilisant la structure comme repère. C'est ainsi que nous décidons de procéder par comparaison plutôt que par une simple lecture du document. Nous utilisons le bruit pour la redondance d'éléments d'une page internet à une autre.

Seuls la structure ou le contenu changent d'une page à l'autre (Figure 2.1). Si nous faisons abstraction de la structure, cela nous donne accès au contenu. Nous avons donc besoin de la page seule pour appliquer notre approche comme suit :

— Notre approche nécessite des ressources à structures semblables pour identifier le "template". Nous choisissons de regrouper les pages par source et par catégorie de page à l'intérieur de cette même source.
— Il ne nous reste ainsi plus qu'à mettre en place notre algorithme de comparaison sur ces paires de page de même structure pour ne faire ressortir que le contenu.

Pour situer notre approche, nous allons d'abord vous présenter un état de l'art sur l'extraction de contenu utile à partir de l'internet. Nous décrirons ensuite notre approche. Puis, nous l'évaluerons et présenterons ses résultats. Pour finir, nous prendrons du recul pour discuter les performances obtenues et déterminer des perspectives.

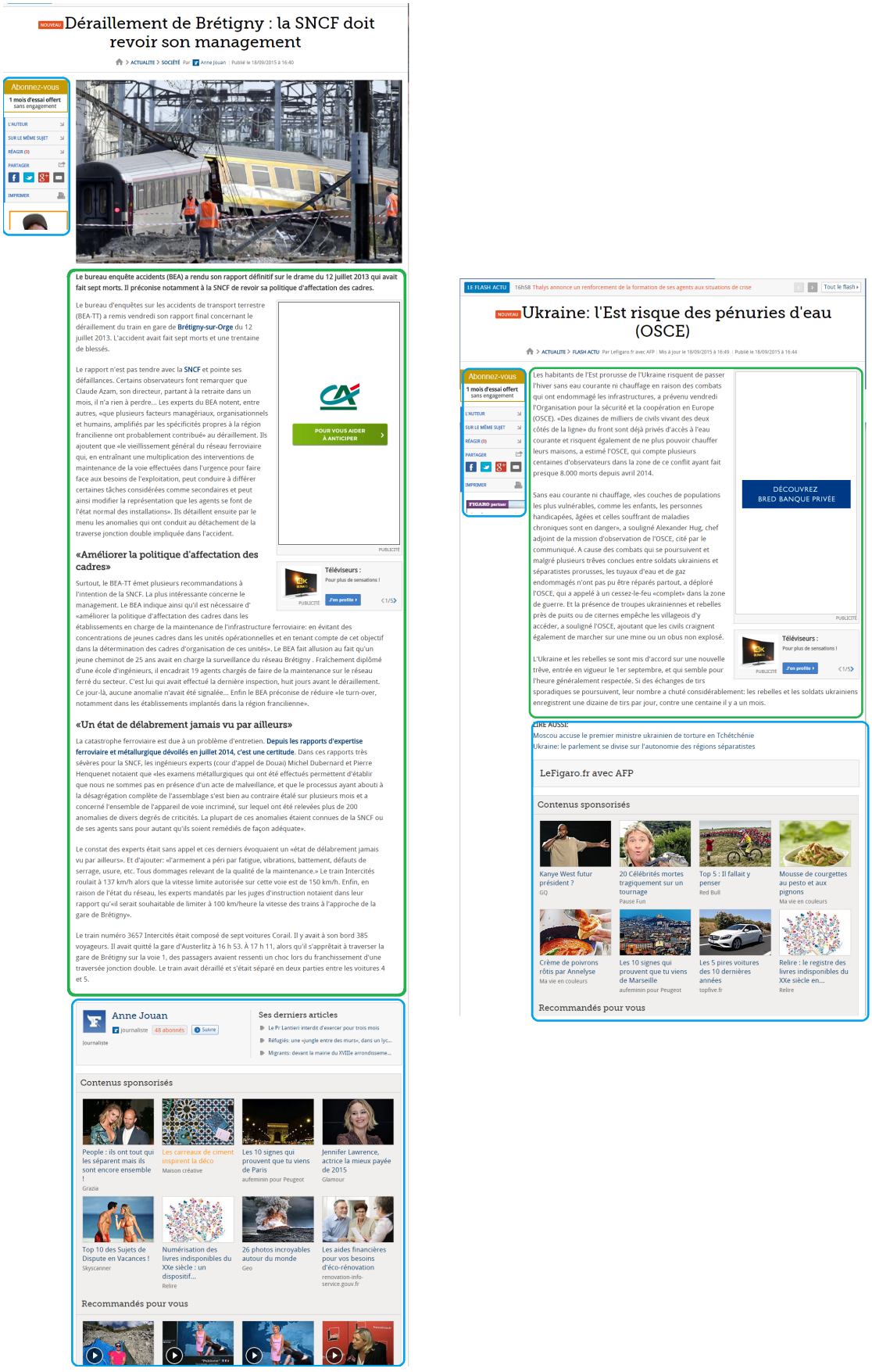


FIGURE 2.1 – Comparaison entre deux pages du site *lefigaro.fr*. Le contenu utile est encadré en vert, et le template en bleu

CHAPITRE 3

ÉTAT DE L'ART

3.1 Notion d'information pour définir le contenu utile

Nous parlons de contenu *utile* pour montrer qu'il ne s'agit pas d'extraire l'ensemble du contenu d'une page internet sans filtre. Ce qui sous-tend notre réflexion est la volonté de repérer le contenu porteur d'information. Nous devons être en mesure de décrire ce qui est informatif pour configurer un système destiné à le repérer. L'"information" est un terme employé dans énormément de domaines différents avec des significations souvent très différentes. Il est alors essentiel d'en poser une définition pouvant s'appliquer aux données textuelles.

C'est dans le domaine des télécommunications, à la fin du XIXème siècle, qu'apparait le concept d'"information" pour améliorer la transmission des messages en contexte de guerre. Lors de la seconde guerre mondiale, Shannon a développé une théorie qui définit les règles de transmission d'un message. Voulant travailler sur la communication de l'information, Shannon dut la caractériser. C'est ainsi que sa théorie de la communication [Shannon and Weaver, 1949] est communément appelée "Théorie de l'information". Il y a trois éléments : le message, l'émetteur et le récepteur. Le message est la donnée transmise, l'émetteur est celui qui envoie le message, le récepteur est celui qui reçoit le message. Le message étant codé par l'émetteur, le récepteur doit connaitre le code pour bien lire le message lorsqu'il le reçoit. La notion de transmission est donc essentielle.
Selon Ackoff [Ackoff, 2010], théoricien des systèmes d'information, ce schéma communicationnel répond à un besoin d'information. Ce besoin nait d'un manque de connaissance pour répondre à un problème. Pour [Shannon and Weaver, 1949], l'émetteur transmet avant tout un message. L'interprétation, soit le sens donné au message, va juger de la pertinence de ce message au regard du besoin exprimé.

Ainsi, nous considérons comme informatif pour un lecteur un message dont le contenu diffère de ce qu'il a déjà pu recevoir.

Shannon observe que lors de sa transmission le message subit des dégâts, créant une désorganisation. C'est la notion de bruit qui est introduite ici. C'est tout ce qui va parasiter le message dans son canal de transmission. Ceci va créer des éléments dans le message qui ne sont pas nécessaires à la transmission de l'information et qui pourront même nuire à sa compréhension. Cette observation implique que le message se désorganisera avec le temps. Ceci conduit Shannon à traiter l'information comme une entité physique à part entière.

Nous pouvons lier la notion d'instabilité de structure à celle du contenu pour un message. Un message dont le contenu change de façon irrégulière a plus de chance d'être inédit et donc d'être informatif ; c'est le cas d'un message aléatoire. Dans ce cas, la variation est liée à la volonté de l'émetteur, contrairement à la variabilité afférente au bruit. C'est dans la méconnaissance de son contenu que le message est informatif. L'entropie est une mesure de la quantité d'information.

Dans le cadre de l'information, l'entropie est en fait une néguentropie ; sa valeur baisse lorsque la désorganisation augmente. De cette façon, pour qu'un message garde son caractère informatif, il doit maximiser sa valeur d'entropie.

La formule de l'entropie H s'applique à une variable aléatoire X dont on connait la distribution (le logarithme en base 2 est utilisé par commodité. l'unité est le *bit*).

$$H(X) = -\sum_{x \in X} P(X = x) \times \log P(X = x)$$

Toujours selon Shannon [Shannon and Weaver, 1949], l'information contenue dans le message est la juste quantité de données nécessaire pour le reproduire. L'information, ce que nous appelons dans notre travail le *contenu utile*, est le message moins le bruit ayant provoqué sa désorganisation. Ainsi, n'importe quelle donnée est la base de l'information. L'information n'est pas une donnée connue de son récepteur. Dans le cas du traitement de données, l'information est par essence porteuse de sens. Le lien est donc étroit entre "information" et "connaissance". Selon Ackoff [Ackoff, 2010], la donnée utile est justement l'information. La donnée prend sa valeur de pertinence pour un individu particulier dans un contexte particulier. La notion de subjectivité est de ce fait très importante lorsqu'on parle d' "utilité".

Les propriétés de l'information que nous venons de définir permettent la construction des techniques présentées dans la suite pour la localiser.

3.2 Segmentation de pages internet

Il s'agit ici de découper une page internet pour n'en conserver que son contenu utile. Dans la constitution d'une page internet, l'auteur tend à regrouper par parties les éléments de sa page. Cette répartition obéit à la fonctionnalité comme au contenu même de chacun des éléments de la structure. Nous pouvons parler de la structure de l'arbre DOM comme de celle du rendu visuel en sortie d'un navigateur internet. Nous distinguons à cet effet la position visuelle en deux dimensions de la position linéaire, l'offset, dans le code HTML de la page internet.
Le but du créateur de contenu est de faciliter l'accès au contenu pour l'utilisateur et donc de l'organiser comme il se doit. La référence pour un algorithme de segmentation de page internet serait donc l'humain qui, naturellement, va repérer par simple lecture la constitution de la page.

[Chen et al., 2003] initièrent ce type d'algorithme pour des raisons pratiques car au début des années 2000 les différents outils portables permettant de naviguer sur l'internet, comme les PDA pour "Personnal Digital Assistant", les ordinateurs portables ou les smartphones, ne permettaient pas d'afficher de trop grandes pages HTML. Le principe était de découper la page pour n'en garder que l'essentiel et ainsi

pouvoir l'afficher sur des systèmes légers. L'algorithme est itératif et descendant avec comme entrée la page entière. Il consiste à classifier sémantiquement les éléments HTML lors du parcours de la page pour que chaque groupe corresponde à un lot d'information.
Toujours dans l'optique d'améliorer la navigation internet sur des appareils mobiles, [Baluja, 2006] propose une *segmentation intelligente* de page internet, consistant à représenter le problème comme celui d'un arbre de décision pour de la classification. L'arbre va représenter les différentes possibilités de découpage. La prise de décision consiste pour chaque nœud à choisir une branche où couper sur la page. On donne en paramètre le nombre de segments que l'on souhaite, correspondant au nombre de segments horizontaux et verticaux qui seront tracés sur le rendu visuel de la page internet.
[Fernandes et al., 2011] propose de changer l'unité d'analyse en entrée de l'algorithme. L'hypothèse de base de la méthode est que toutes les pages d'un même site partagent la même structure. Les auteurs vont alors prendre en considération le site entier pour segmenter plutôt qu'une page seule. Chaque modèle créé est donc lié à un seul et même site. La difficulté de cette approche est de devoir générer un nouveau modèle pour chaque nouveau site rencontré. Ce coût est tout de même compensé par une meilleure précision de l'algorithme. Ces approches prennent donc pour parti de n'utiliser que la structure DOM d'une page internet.

[Cai et al., 2003] proposent le principe novateur de se calquer sur le rendu visuel de la page internet pour la découper, comme peut le faire un utilisateur humain devant son écran. Non content des résultats proposés par le Traitement Automatique des Langues, il préfère ne se baser que sur la structure visuelle de la page en essayant de l'inférer. Ceci fonctionne par une simulation de navigateur sur la page pour obtenir son rendu tel qu'un utilisateur de référence l'obtiendrait. Il s'agit de créer une nouvelle structure pour le contenu de la page fondée sur son rendu visuel. La donnée d'entrée est la page entière. À chaque récursion, un nouveau segment est considéré. Chaque segment est alors représenté par des blocs qui ne correspondent pas nécessairement aux nœuds DOM d'origine. Les auteurs utilisent en plus les séparateurs visuels pondérés selon leur visibilité et les relations entre blocs.

[Chakrabarti et al., 2008] proposent quant à eux de voir le problème de segmentation de pages internet comme celui de la minimisation de chemin sur un graphe pondéré. Comme [Cai et al., 2003], ils prennent aussi pour parti de prendre en compte le rendu visuel de chaque nœud tout en considérant quand même la structure DOM pour les relations entre nœuds calculées comme des distances dans l'arbre. Leurs propriétés visuelles intrinsèques sont aussi prises en paramètres. Ces trois éléments sont entrés dans une fonction qui doit être minimisée pour pouvoir décider que deux nœuds doivent se trouver dans le même segment. Chaque application de fonction retournera une valeur numérique qui est le coût d'une certaine segmentation. Ainsi, plus une segmentation s'avère coûteuse moins elle est considérée comme étant de qualité. Cette approche s'adapte aux pages très structurés à la base par la prise en compte des relations entre nœuds DOM comme à ceux qui ont une structure plus volatile mais dont on peut inférer un découpage par leur aspect visuel.

[Kohlschütter and Nejdl, 2008] proposent l'utilisation du texte pour segmenter avec un apport un peu plus linguistique qui utilise la densité de texte comme critère de choix. Leur approche ne prend pas du tout en compte la structure de la page ni

tout ce qui a trait à l'aspect visuel. Les blocs d'un arbre DOM vont être fusionnés selon leur densité de texte. Cette mesure de densité s'appuie sur une quantité de "tokens" qui sont les segments de texte séparés par des espaces. Elle est obtenue par la formule suivante : $D(b) = \frac{N_{tokens}}{N_{lignes}}$, avec D la densité, b le bloc, N_{tokens} le nombre de tokens dans le bloc et N_{lignes} le nombre de lignes.

Ainsi, le contenu textuel est dimensionné par sa hauteur en nombre de lignes et quantifié avec le token pour unité. Cette méthode n'a pas besoin de procéder à des analyses linguistiques plus poussées comme le lexique ou la syntaxe. Seules deux statistiques sont nécessaires par bloc. L'algorithme est glouton car il fait en sorte de constituer le plus grand segment avec des blocs de même densité textuelle.

Nous venons de présenter des méthodes qui distinguent différentes parties structurelles sur une page alors que les prochaines repèrent la répétition d'une seule structure pour localiser le contenu utile.

3.3 Détection de motifs structurels

Détecter une répétition de motifs à travers des pages internet permet de remarquer un cadre standard employé par n'importe quel auteur de contenu. Sur un corpus constitué d'un ensemble de pages qui sont certifiées par des humains comme possédant la même topologie (aux arbres DOM similaires), il sera alors possible de détecter un invariant. Cet invariant sert de balisage pour accéder à notre contenu utile.

[Kushmerick, 2000] présente une approche dont le principe de base est de repérer les délimiteurs gauche et droite du contenu utile. Ces délimiteurs sont en fait des chaines de caractères cadrant le contenu visé. À chaque cadre correspond un contenu utile. Un corpus de référence contient des exemples de page et leurs délimiteurs associés. Les frontières gauche et droite doivent être indiquées proprement pour que la reconstitution s'effectue correctement. La méthode par induction utilise des contraintes se calquant sur le corpus de référence pour définir et valider les frontières candidates. À chaque jeu de pages de même topologie est alors attribué un vecteur de tuples. Chaque tuple contient un délimiteur gauche et un délimiteur droit pour un certain contenu cible. Dans le but de s'adapter eux ressources plus complexes, l'auteur distingue différentes variantes selon les biais utilisés pour l'apprentissage. Ces biais utilisant des éléments HTML spécifiques aux cas particuliers. L'approche nécessite une quantité particulièrement importante de données pour l'apprentissage.

[Muslea et al., 2001] tente de diminuer la quantité de données annotées à utiliser pour l'apprentissage. Il définit des règles d'extraction par la constitution d'automates. Cette méthode fait appel à une nouvelle représentation des documents, nommée "embedded catalog", qui vise à mettre en évidence l'information d'une large gamme de pages. C'est une structure dans laquelle toutes les feuilles sont des unités d'information. À l'intérieur, chaque nœud est une liste d'éléments de type tuple contenant une feuille ou une nouvelle liste de même nature que précédemment. Sur cette nouvelle structure l'auteur utilise des repères pour atteindre le contenu ciblé. Ces repères sont en fait des motifs séquentiels de tokens bruts ou de tokens classés selon leur ontologie.
La limite de ces deux dernières méthodes tient dans le fait que chacun des motifs ainsi définis est propre à une certaine topologie de page. Ce n'est pas généralisable.

Cela nécessite donc un coût important pour s'adapter à la variété des cas rencontrés. La maintenance est aussi essentielle puisque les structures de pages peuvent changer. Nous pouvons même dire qu'elles se complexifient avec de nombreux contenus additionnels ajoutés à l'information de base. Le contenu même d'une page étant de plus en plus divers et même sans rapport interne évident. Les méthodes par motif vont alors se focaliser sur le bruit plutôt que chercher à atteindre directement le contenu recherché.

[Crescenzi et al., 2001] veut diminuer l'importance de l'intervention humaine pour ce type de méthodes. Il choisit alors de comparer directement deux pages internet en regardant ce qui change et ce qui ne change pas pour inférer la présence de contenu utile. Nul besoin d'étiqueter des données en amont. Il utilise l'algorithme "match" qui permet de comparer deux arbres HTML entre eux. La structuration des arbres doit être assez exigeante pour que l'algorithme fonctionne ; il utilise la norme XHTML qui est du HTML plus restrictif quant à l'ouverture et la fermeture des balises. Il utilise aussi de la linguistique avec un analyseur lexical pour la tokenisation des ressources textuelles. L'algorithme nécessite toujours deux documents pour fonctionner. Chaque token peut être une étiquette HTML ou n'importe quelle chaine de caractères. La base de l'algorithme est une expression régulière. Le principe est de trouver l'expression régulière commune aux deux pages entrées et tokenisées par la résolution des différences. L'initialisation est faite par la description d'une seule page par expression régulière. Le traitement consiste ensuite à modifier cette expression régulière pour qu'elle coïncide avec l'autre page. Ainsi, les seuls éléments qui vont être mis en avant seront ceux qui diffèrent. Ce sont les éléments qui diffèrent d'une page à l'autre qui sont considérés comme de l'information.

[Bar-Yossef and Rajagopalan, 2002] introduisent le terme de "Pagelet" pour désigner une région de la page internet qui possède une certaine autonomie. Cette notion permet d'isoler sur une même page les unités d'information. Pour détecter le type "pagelet" les auteurs prennent en entrée une seule page internet et ils utilisent l'autonomie fonctionnelle ou thématique du "pagelet". Leur critère de choix est la présence de lien hypertexte. Si un nœud HTML contient plus de K lien(s) hypertexte(s) (avec K un paramètre au choix), alors l'algorithme considère qu'il s'agit d'une unité à part. Les auteurs traitent aussi le cas du "template" qu'ils définissent comme une collection de pages possédant une même topologie et sous la même autorité. Alors, peu importe l'information d'une page à une autre, elles peuvent toutes être similaires donnant une même impression à l'utilisateur ou à la machine, ce qui rend d'autant plus difficile la recherche d'information. Pour détecter un template, les auteurs reconsidèrent la notion comme étant un ensemble de "pagelets" avec les deux caractéristiques suivantes : le contenu HTML de chaque "pagelet" est identique et chaque page appartenant au même "template" doit former un ensemble connecté par un même chemin dans la page vers la déclaration d'autorité.

[Yi et al., 2003] proposent une méthode fondée sur la structure arborescente d'une page internet et utilisant un calcul d'information. L'approche part de pages appartenant à un même site pour pouvoir définir une structure rendant compte à la fois de leur contenu et de leur disposition. L'utilisation des attributs (id, class, style) des balises de l'arbre DOM n'est pas suffisante. Les répétitions d'organisation sont repérées d'une page à l'autre. Le poids d'une certaine organisation est mesuré par un calcul d'entropie. L'importance de chaque nœud est considérée selon la diversité de

son contenu et de la variété de sa présentation. [Vieira et al., 2006] mettent en avant le coût important de la méthode précédente à cause des multiples passes à travers les pages d'une même classe qui sont nécessaires pour repérer le bruit. La nouvelle méthode alors proposée par les auteurs de cette critique consiste à calculer en un seul tour le motif optimal permettant d'associer un arbre à un autre. Cela revient en fait à calculer une distance d'édition entre arbres, problème analogue à la distance de Levenstein appliquée aux chaines de caractères. Lors des opérations d'édition, il convient de trouver une structure de sous arbre qui se répète ce qui constituerait une trace de bruit.

[Kao et al., 2005] proposent un système entier nommé WISDOM pour "Web intrapage informative structure mining based on the document object mode". Son but est de viser l'information pertinente parmi les blocs HTML d'une page internet juste à partir de la structure DOM. Trois phases :

— Calcul d'information dans l'arbre DOM.
— Prendre les K (valeur arbitraire) blocs les plus riches en information.
— Agrandir le lot de segments d'information en récupérant ceux qui partagent les mêmes caractéristiques.

Les auteurs utilisent le contenu textuel en extrayant les mots reflétant le contexte sémantique à partir d'une liste de mots considérés comme étant "vides". Pour mesurer l'information, c'est un classique calcul d'entropie qui est ensuite effectué sur les termes récoltés. Les auteurs s'appuient en même temps sur les propriétés des nœuds HTML pour déterminer ceux qui ne sont que de la structure pure.

[Chen et al., 2006] s'intéressent à l'indexation dans les moteurs de recherche en voulant faire en sorte que le "template" ne soit plus pris en compte dans l'opération. Leur choix est d'effectuer des mesures de similarités entre blocs. Ils prennent chaque page internet comme unité d'entrée parce qu'ils considèrent inefficace la détection de "template" au niveau du site entier. C'est le bloc qui est mis en avant. Ils s'appliquent à segmenter la page en blocs distincts. Cette segmentation est réalisée en prenant en compte la présentation des éléments sur la page. Tous les blocs possédant une disposition similaire sont mis ensemble. Si du contenu ressemblant est observé dans ces groupes de blocs ils sont supprimés car considérés comme du bruit étant donné leur redondance. Pour mesurer la similarité des contenus, ils utilisent la distribution des mots du texte. Les paramètres sont alors la fréquence des termes (TF-Term Frequency) et leurs positions dans la portion de texte.

[Chakrabarti et al., 2007] prennent pour parti de classifier les nœuds. L'algorithme utilisé attribue une valeur mesurant à quel point un certain nœud est de nature à constituer un "template". C'est ce score qui est optimisé par la fonction de lissage. L'apprentissage du classifieur est réalisé en préprocessant chaque page de façon à supprimer les mots considérés comme étant vides et par l'annotation de chaque élément selon sa disposition dans la page. L'approche se positionne au niveau du site entier. Le classifieur se veut être la représentation la plus générale de ce qu'est un "template". Les traits ainsi utilisés considèrent le contenu comme la disposition dans la page.

3.4 Extraction de contenu en langage naturel

Pour cette classe d'algorithmes, le problème consiste à extraire d'une page internet son texte essentiel. Ce sont donc des méthodes qui se focalisent cette fois sur le contenu linguistique des pages internet. C'est sur des descripteurs linguistiques et de textométrie qu'on va s'appuyer pour faire de l'apprentissage.

[Gibson et al., 2007] réduit ce problème à la linéarisation des segments de texte provenant des éléments HTML. La sortie est une séquence textuelle uniforme qui correspond au contenu utile. Les auteurs utilisent une labellisation des segments dans la séquence. Cette labellisation est bien meilleure lorsque les nœuds importants ne sont pas consécutifs mais séparés par du bruit. Ce bruit est un marqueur pour repérer le contenu utile. Chaque séquence de texte est donnée à un étiqueteur statistique. L'étiqueteur est avant toute chose paramétré en utilisant des informations statistiques et le résultat d'analyses linguistiques sur le corpus dont les pages sont issues. Leur étiqueteur est en fait une variante de "Conditional Random Field" (CRF), des algorithmes couramment utilisés pour la tâche d'étiquetage. Notons l'algorithme Victor par [Marek, 2007] qui a obtenu les meilleurs scores d'évaluation en utilisant ce procédé de linéarisation avec des traits de structure et de contenu textuel comme paramètres donnés à un CRF.

[Gottron, 2008] pointe le fait que le contenu utile se situe dans une zone avec une majorité de texte et une minorité de code HTML. Cette approche a la particularité d'aller vers un niveau d'analyse encore plus profond que celui de la page puisqu'il se focalise sur le caractère. Chaque caractère est étiqueté comme du contenu ou comme du code selon son appartenance soit à un élément HTML soit à du texte visible. L'auteur a aussi travaillé au niveau du token en prenant cette unité pour découper chaque page, le tag HTML étant considéré comme un token. L'algorithme s'applique à créer une séquence de caractères ou de tokens. Il y a une séquence d'initialisation de la séquence par des chiffres : 1.0 pour du code et 0.0 pour du contenu. C'est pour l'auteur une référence au "Content Code Vector" (CCV) pour la séquence de codes qui sert à détecter la frontière entre une région de code et une région de contenu. Un seuil est fixé comme paramètre au CCV qui va ainsi déterminer la zone de contenu sur la page.

L'outil "NCleaner" [Evert, 2008] met en avant l'utilisation de modèles N-Gramme pour détecter le contenu essentiel. La première phase consiste en un prétraitement pour récupérer les paragraphes d'une page. Ces blocs de texte sont ensuite transmis à deux modèles N-Gramme de caractères distincts. L'un est entrainé pour détecter le texte source de bruit, l'autre pour détecter le texte essentiel. Ces deux modèles sont utilisés de façon comparative. La règle est simple : la classe attribuée à un bloc est celle qui est spécifique au modèle qui donne la plus grande probabilité pour ce même bloc. L'auteur utilise deux types de configuration pour la modélisation. Un premier type "lexical" entrainé sur du texte brut mais dont la limite tient dans son entrainement avec la seule langue anglaise et donc applicable seulement sur du contenu dans cette même langue. Le second type est "non-lexical" car le texte est normalisé tel que chaque caractère non numérique est codé "a" tandis que chaque caractère numérique est codé "1". C'est en quelque sorte un assistant du modèle précédent.

[Pasternack and Roth, 2009] utilise un classifieur pour attribuer un score à chaque token. Le principe est de maximiser une séquence de scores. Un prétraite-

ment est appliqué sur chaque page en racinisant les tokens et en normalisant les tokens numériques. Les auteurs attribuent les scores avec un classifieur Bayésien naïf configuré selon deux paramètres : des trigrammes de tokens et l'étiquette de l'élément HTML courant. Sur une séquence de tokens, un trigramme de tokens contient le token à la position courante et deux tokens consécutifs après la position courante. L'élément HTML courant est celui correspondant à la dernière balise ouvrante rencontrée à la position courante. Pour définir une région de texte essentiel, on prend la sortie du classifieur qui est un vecteur de scores et on le découpe de façon à obtenir une sous-séquence qui maximise la somme des scores. On généralise ensuite les résultats pour obtenir une prédiction au niveau des pages internet.

[Kohlschütter et al., 2010] s'intéressent à des propriétés de bas niveau sur le texte. Ils utilisent la technique de linéarisation d'un arbre DOM en constituant une suite de portions de texte. Ils visent les niveaux les plus bas de l'arbre en les annotant avec des caractéristiques linguistiques, de structure et de textométrie. Leurs expérimentations utilisent à la fois les arbres de décision et les "Support Vector Machine" (SVM) pour la tâche d'apprentissage. Le corpus de référence utilisé est annoté manuellement par des humains. C'est le même corpus qui sera utilisé pour l'évaluation. Le classifieur ainsi entrainé est ensuite utilisé pour former une prédiction sur chaque bloc de texte pour une page internet jamais vue par le système.

[Weninger et al., 2010] passe par une structure supplémentaire nommée "Text-to-Tag Ratio" (TTR) pour analyser l'arbre DOM en entrée. Chaque page internet subit un prétraitement consistant à retirer toutes les étiquettes HTML et et les lignes vides. La structure TTR est constituée d'un tableau qui attribue pour chaque ligne du document HTML prétraité la valeur du ratio entre son texte pur et ce qui constitue une étiquette HTML. C'est à partir de la structure TTR que les auteurs appliquent plusieurs méthodes de classification comme la maximisation d'espérance (Expectation Maximization-EM) ou K-Means. Ils utilisent aussi une heuristique qui pose pour seuil de détection l'écart-type issu du lissage de la structure TTR.

[Sun et al., 2011] utilisent la caractéristique de la densité textuelle et des heuristiques empiriques. Après expérimentation en n'utilisant que la densité de texte, ils en sont venus à utiliser une nouvelle métrique nommée "Composite text Density" (CTD). Cette dernière métrique se distingue d'une densité classique de texte dans sa pénalisation des éléments HTML contenant trop de liens hypertextes pointant vers des sources de "template" déjà identifiées. D'abord, la somme des CTD est propagée au niveau des parents pour chaque élément. C'est la densité du nœud "body" est considérée comme seuil pour définir une zone de contenu. Ensuite, l'algorithme passe dans l'arbre en descendant pour identifier un nœud qui possèdent une valeur de densité supérieure au seuil. Il recherche enfin l'élément possédant la valeur supérieure la plus grande. Tous les fils de l'élément sélectionné seront alors considérés comme porteurs de contenu.

En conclusion, nous retenons que les méthodes n'utilisant qu'une page seule nécessite des paramètres plus importants que celles qui procèdent par comparaison. Le "template" semble être un indice efficace pour se repérer dans la structure DOM d'une page internet. Nous choisissons alors d'utiliser nous aussi la comparaison des pages et le "template" en utilisant des heuristiques empiriques pour mieux cibler le contenu utile. Nous vous les présentons dans la prochaine section.

CHAPITRE 4

APPROCHE

Les méthodes qui s'appuient sur de la détection de "template" nous semblent particulièrement intéressantes pour l'absence d'apprentissage supervisé. Nous choisissons alors de reprendre l'idée de départ que la différence est source d'information. Nous reprenons aussi la nécessité d'utiliser des sources possédant le même "template" pour être certains de n'identifier au final que le contenu utile. Nous pouvons rappeler à cet égard [Fernandes et al., 2011] qui vise les similarités de structure à l'intérieur d'un même site pour y trouver l'information voulue ou encore [Crescenzi et al., 2001] qui, déjà en 2001, faisait le choix de comparer les pages internet entre elles pour identifier les changements avec une heuristique affirmant que l'information réside dans ce qui diffère. Ces méthodes avaient toutefois la limite d'être spécifiques selon les structures rencontrées. Nous voulons que cette heuristique sur la différence devienne plus robuste.

Nous mettons ainsi en œuvre une méthode de classification non supervisée des pages internet pour effectuer le regroupement par structure. Sur les regroupements ainsi créés, nous utilisons la mesure de distance d'édition entre arbres pour associer les nœuds correspondants et distinguer les zones inédites. C'est par là même une façon de quantifier l'information à travers la structure arborescente d'une page internet. Nous décrivons des heuristiques (hypothèses déterminées empiriquement et qui n'ont pas vocation à être absolues) concernant les pages internet afin de cibler finement le contenu utile dans la zone repérée comme significative.

Nos travaux obéissent à une démarche scientifique itérative qui peut se décrire avec les étapes suivantes :

— Une modélisation du problème.
— Nos heuristiques de départ et un corpus de développement pour les vérifier.
— Constitution de paires sur des pages classées par structures similaires.
— Nouvelles heuristiques supposant plus de propriétés sur les pages pour plus de précision.

Nous avons posé les grandes lignes de nos travaux. Passons maintenant à leur description.

4.1 Modélisation

Notre approche fonctionne en deux grandes étapes :

— Le regroupement en paires de pages comparables structurellement.
— Entre les pages appariées, la comparaison des nœuds textuels équivalents par leur position respective dans l'arbre HTML.

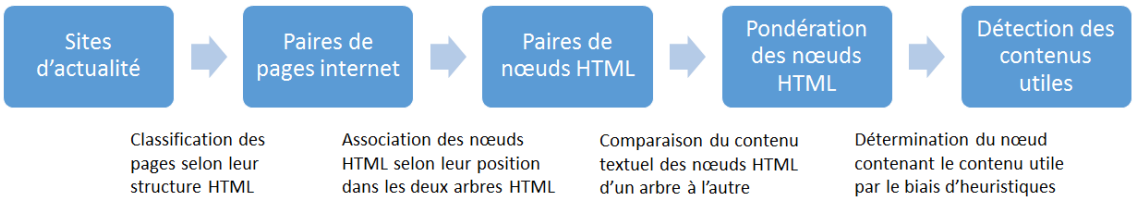


FIGURE 4.1 – Schéma résumant notre approche.

Nous avons choisi le genre des sites d'actualité. Se focaliser sur un type de corpus permet de déterminer plus aisément la structure générale d'un document. Sur une page d'actualité, nous trouvons en général un article avec des contenus additionnels comme ses métadonnées. Ce que nous définissons comme le contenu utile est le corps de l'article parce que nous considérons que c'est l'élément qui donne sa raison d'être à la page d'actualité. Nous avançons que le contenu utile a une taille de texte particulièrement importante et possède un caractère spécifique à l'intérieur de sa page.
Nous mesurons la spécificité d'un contenu en le comparant pour des pages équivalentes du point de vue de leurs structures. Ceci pour comparer des nœuds textuels de mêmes natures.
Ainsi, nous fixons l'élément à récupérer pour chaque page analysée comme étant la balise contenant la plus grande quantité de texte différent.
La différence mesurée est issue d'une comparaison entre pages internet similaires. La comparaison fonctionne en associant les arbres HTML respectifs aux deux pages comparées. C'est la supposition d'associer des nœuds correspondants dans les deux arbres qui nous fait utiliser des pages similaires en termes de cadre. Trois caractéristiques importantes donc :

— La taille de la zone de texte qui est en fait sa quantité de texte exprimée en nombre de token.
— La différence de texte qui se mesure par la quantité de texte différent en comparant deux pages similaire en structure.
— L'heuristique qui suppose qu'une page d'actualité sur internet présente un titre suivi d'un corps d'article.

Le point sensible pour notre approche est de pourvoir détecter la zone qui ne contient que le contenu qui nous intéresse. L'erreur serait de sélectionner des zones de texte différentes mais qui n'appartiennent pas au corps de l'article. Les commentaires, les métadonnées, les contenus connexes voire les publicités peuvent avoir une place plus importante que le corps de l'article. Nous avons alors posé une heuristique sur la structure générale des sites d'actualité affirmant que le titre précède toujours le corps de l'article.

Pour débuter nos expériences, nous avions besoin immédiatement d'un corpus-jouet constitué de paires de pages regroupées d'après leur structure. Pour cela, nous avons manuellement sélectionné les ressources sur lesquelles nous voulions travailler. Nous avons fait en sorte d'avoir des regroupements par site puisque nous partons du principe que les pages d'une même section sur un site sont construites sur la base un même "template". Il suffit pour ce point de prendre en compte l'URL de la source. Puis, à l'intérieur de ces sites, nous avons constitué des paires de pages comparables. Pour constituer ces paires de façon manuelle, nous avons dû repérer pour chacun des sites sélectionnés les pages qui semblaient obéir à un même "template" dans ce site, répondant donc à des critères de structure.

4.2 Corpus de développement

Nous avons constitué un ensemble de pages assez éclectique et d'une dimension suffisante pour permettre des premières observations déterminantes.
Voici la description de ce premier corpus :

— Catégorie "Blogs" : une paire de pages depuis *CocktailMolotov.org*, une paire de pages depuis *francoismonti.com*
— Catégorie "Actualités" : cinq paires de pages depuis *Atlantico.fr*, cinq paires de pages depuis *LeFigaro.fr*, cinq paires de pages depuis *LeMonde.fr*
— Une paire de pages depuis le site *TED.com*
— Catégorie "Commerce" : une paire de pages depuis le site *Amazon.fr*, une paire de pages depuis le site *Fnac.com*
— Une paire de pages depuis le site *LegiFrance.gouv.fr*
— Une paire de pages constituée de deux articles issus de *Wikipedia.fr*

Statistiques sur le corpus :

— Taille totale en nombre de paires de pages : 17.
— Nombre de mots moyen : 2000
— Nombre de liens moyen : 200
— Nombre de nœuds moyen : 1000
— Profondeur moyenne : 15
— Largeur moyenne : 70

La ressource de notre approche est un ensemble de paires de pages comparables, propriété nécessaire pour faire une association entre des nœuds qui se correspondent d'un arbre à l'autre. Nous voulons dès lors faire en sorte que le système parvienne à constituer automatiquement les paires de pages comparables.

4.3 Regroupement de pages internet similaires

Pour déterminer si deux pages internet sont comparables, nous utilisons une mesure de distance entre leurs arbres HTML respectifs. La comparaison d'arbres est un problème bien connu en informatique et auquel répondent les algorithmes de distance d'édition entre arbres. Ces algorithmes utilisent le principe de la distance de Levenshtein ou distance d'édition qui s'applique d'ordinaire aux chaines de caractères [Levenshtein, 1966]. Ce principe vise à quantifier les dissimilarités entre deux

entrées pour dire à quel point elles sont éloignées l'une de l'autre. Le résultat est égal au nombre minimal de caractères à supprimer, insérer ou modifier pour pouvoir passer d'une chaine à l'autre. Le résultat vaut zéro lorsque les deux chaînes entrées sont identiques. Plus les chaînes sont différentes l'une de l'autre et plus le résultat est grand.
Pour plus de précisions, voir l'annexe (algorithme-1). La distance de Levenshtein vérifie l'inégalité triangulaire : $Levenshtein(a, c) \leq Levenshtein(a, b) + Levenshtein(b, c)$, avec a, b, c trois chaines de caractères et $Levenshtein$ la distance de Levenshtein. Elle est symétrique, c'est-à-dire qu'elle donnera le même résultat si ses deux membres sont permutés. $Levenshtein(A, B) = Levenshtein(B, A)$, avec A et B deux chaines de caractères.
Nous voulons appliquer ce principe aux arbres.

Nous avons utilisé l'implémentation Java de l'algorithme proposé par la bibliothèque RTED [Pawlik and Augsten, 2011] pour "Robust Tree Edit Distance".
Le principe reste le même que pour la distance de Levenshtein appliquée à des chaines de caractères. Les opérations d'édition utilisées sont encore l'insertion, la suppression et la substitution. L'insertion installe un nœud entre un autre nœud déjà présent et ses fils éventuels. La suppression défait un nœud de son parent et relie ses fils à sa place. La substitution modifie l'étiquette d'un nœud.
En sortie de l'algorithme RTED, nous pouvons obtenir un rapport d'édition pour les deux arbres donnés. Ce rapport d'édition décrit la séquence des opérations d'édition réalisées par l'algorithme RTED pour faire l'association entre les deux arbres donnés. Chaque opération apparait comme un couple de nœuds identifiés par une valeur numérique. "0" désigne l'absence de nœud. Nous supposons que dans ce rapport d'édition, les nœuds associés sont comparables d'un arbre à l'autre. C'est ce rapport d'édition que nous utilisons pour identifier les nœuds qui apportent de la différence. Nous choisissons pour cela les couples comportant des nœuds différents.
Pour obtenir ces associations entre nœuds, l'algorithme RTED optimise et décompose le problème d'édition entre arbres à des fins de robustesse.

Le calcul de distance d'édition entre arbres est particulièrement coûteux parce qu'il envisage les différentes configurations possibles pour l'association. C'est contraignant pour la gestion mémoire et déterminant pour la robustesse de notre algorithme. L'algorithme utilisé pour notre approche est optimisé de façon à atteindre une complexité en O(n^2) avec n le nombre de nœuds le plus grand entre les deux arbres donnés. Le principe de cette optimisation est de découper chaque problème en sous-problèmes. Le point important est de toujours choisir la meilleure décomposition. Pour cela, l'algorithme définit une stratégie du chemin optimal qui guidera la manière de décomposer. Le but de la décomposition est de produire des sous-forêts pertinentes et des sous-arbres pertinents à partir d'un arbre sélectionné d'après le chemin énoncé dans la stratégie définie auparavant. Ces éléments sont ensuite utilisés dans l'algorithme général pour alléger le calcul de distance.
Nous voulons appliquer le calcul sur toutes les pages d'un site pour les situer les situer entre elles.

À partir d'une matrice de distance calculée pour chacun des sites de notre jeu de données, le but est à présent de regrouper les pages de façon automatisée. Nous formons des paires en alliant simplement les pages qui sont les plus proches l'une de l'autre. Cette méthode s'avère efficace mais ne permet pas de repérer les pages qui

seraient à part dans l'ensemble donné. Le risque est de créer des couples avec toutes les pages quelle que soit leur distance les unes aux autres. Cela mènerait alors à comparer des éléments non comparables, ce qui est critique pour notre algorithme. Dans un contexte naturel, notre système serait branché à une collecte qui peut renvoyer des pages aux structures très différentes sans que nous puissions le contrôler. Nous avons donc besoin d'une méthode de classification plus complexe dans l'optique de repérer les "outliers", ces pages avec lesquelles nous ne pouvons pas associer d'éléments comparables. Dans le cas des sites d'actualité, les "outliers" sont des pages ne présentant pas un article mais plutôt une sélection ; nous pouvons donc les nommer "menu".

Nous avons besoin d'augmenter le nombre de pages par site pour mettre à l'épreuve les différentes techniques de clustering que nous allons sélectionner. Nous sommes ainsi passés de 34 pages précédemment à un corpus de 50 pages par site étudié. La constitution de ce nouveau corpus est plus fine puisque nous souhaitons créer des classes de pages pour un site donné. Ces classes de pages sont en rapport avec la sémantique des pages à l'intérieur d'un certain site. Ces classes de référence réalisées manuellement seront donc celles que devra reconstituer chaque technique de classification essayée. Nous nous sommes alors arrêtés à deux sites pour ce corpus de référence : *Atlantico.fr* et *LeFigaro.fr*.
Les classes et leur taille ainsi constituées par site sont les suivantes :

— *Atlantico.fr* : Articles (14 pages), les pages présentant une compilation d'articles (10 pages), les dépêches (10 pages), les menus (10 pages).
— *LeFigaro.fr* : les dépêches (10pages), les articles Figaro Madame (10 pages), les recettes (8 pages), les tribunes Figaro Vox (10 pages).

Notre classification ne se fonde que sur la structure des pages. En effet, ce que nous voulons mettre en avant ce sont les rapprochements dans l'organisation des pages pour déceler les traces de "template".

Sur la Figure4.2, nous pouvons constater que les classes manuelles sont assez homogènes puisque chaque zone blanche regroupe les pages d'une même classe annotée manuellement. Les classes à trouver sont donc mises en évidence par le critère de distance, ce qui est positif pour la classification automatique. La catégorie "Vox" sera la plus difficile tandis que celle des recettes est la plus simple.

Notre algorithme va donc comparer les arbres HTML des pages à partir de leur seule structure. Il ne s'agit à aucun moment de prendre en compte le contenu textuel. Nous choisissons d'utiliser un calcul de distance d'édition entre arbres pour comparer les pages entre elles. Depuis le nouveau jeu de données auparavant constitué :

— Pour chaque site de notre jeu de données, récupérer toutes les pages indépendamment de leur classe attribuée manuellement.
— Générer la matrice des distances sur un site en utilisant les valeurs de distances d'édition entre les arbres de chacune des pages.

Nous avons entrepris d'expérimenter l'apprentissage non-supervisé pour classer automatiquement des documents. Nous choisissons le type "non-supervisé" parce qu'il ne nécessite aucune donnée d'apprentissage ; l'algorithme est adaptable. Cela permet de traiter des données différentes sans une modification complexe du para-

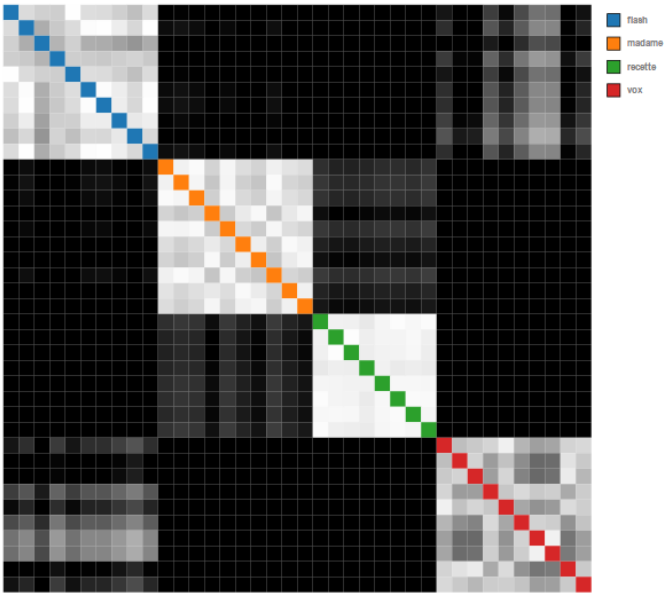


FIGURE 4.2 – Visualisation des distances entre les pages *lefigaro.fr* classées manuellement. On utilise le blanc pour une distance de 0 et le noir pour une distance de 1. Les couleurs sur la diagonale correspondent aux classes de chaque page. Le graphique se lit en ligne et colonne pour voir la distance d'une page à une autre. Les zones claires montrent les clusters décris par la distance d'édition, on remarque que ces clusters correspondent globalement aux classes.

métrage de l'algorithme. Le seul corpus que nous devons constituer est la référence pour l'évaluation. Pour choisir notre méthode de classification, nous avons utilisé la librairie Scikit-Learn [Pedregosa et al., 2011] en Python dont la simplicité et la documentation permettent d'essayer rapidement des familles différentes d'algorithmes de classification. Parmi les techniques de classification non supervisées proposées, nous avons expérimenté dans un premier temps celles prenant en entrée une matrice de distance et non des positions dans un espace multidimensionnel :

— La classification dite "spectrale" qui réalise une projection de la matrice dans un espace et applique une classification dans cet espace. Ce type de clustering fonctionne très bien sur les images pour repérer des zones de pixels similaires mais n'est pas optimisée pour notre cas parce que nos matrices ne sont pas aussi éparses.
— La classification hiérarchisée découpe ou regroupe les classes de manière à en former une hiérarchie représentée sous la forme d'un arbre. Cette famille d'algorithmes nécessite un paramétrage trop important.
— L'algorithme DBSCAN pour "Density-Based Spatial Clustering of Applications with Noise" qui prend en considération les zones à forte densité et les voisins de ces zones pour définir des classes. Les paramètres de voisinage ne sont pas évidents à définir dans notre cas.

Notre choix s'est porté sur l'algorithme K-Means parce qu'il nécessite le moins de paramètres à propos des classes à constituer. Le problème est qu'il n'est pas directement applicable sur une matrice de distances parce qu'il utilise une représentation

des clusters dans un espace avec un certain nombre de dimensions. Nous l'avons alors utilisé en association avec un algorithme de projection de matrice de distance nommé MDS pour "multidimensionnal scaling". Cet algorithme permet de représenter nos documents dans l'espace selon les distances mesurées entre eux. Plonger les données dans un espace multidimensionnel apporte une meilleure visibilité afin de mieux pointer vers les "outliers". Nous avons jugé qu'un espace à deux dimensions était suffisant pour classer correctement nos données.
Une fois la matrice de distances projetée dans l'espace, il est possible d'appliquer le K-Means. Le principe de l'algorithme K-Means est de définir des points virtuels appelés "centroïdes" qui représentent les centres des clusters. Ces points sont initialement positionnés au hasard. Ils sont ensuite déplacés pour être au centre d'un certain ensemble de pages. Finalement, lorsque l'optimisation n'est plus possible, l'algorithme conserve les centroïdes en place qui permettent d'identifier les clusters ainsi constitués.
Il est important de noter que la combinaison du MDS avec K-Means induit une disposition aléatoire à chaque nouvelle application. La méthode n'est donc pas déterministe. Les éléments comparés ne seront pas forcément les mêmes à chaque lancement et les résultats pourront ainsi différer. Nous devons lui fournir à chaque application le nombre de clusters "k" désiré.

Pour trouver le nombre de classes optimal, nous calculons le coefficient de silhouette pour chaque K-Means lancé avec un certain paramètre k. C'est une valeur qui permet d'évaluer la pertinence des k clusters obtenus pour une certaine application du K-Means. Elle permet de dire à quel point un élément est lié au cluster dans lequel il a été classé en utilisant la dissimilarité qui est un indice sur la distance. Pour cela le calcul prend deux critères en considération :

— La dissimilarité moyenne entre un élément i et les autres éléments se trouvant dans la même classe. Ce terme $a(i)$ quantifie la force du lien d'un élément avec son cluster.
— La plus petite dissimilarité d'un certain élément i par rapport aux éléments présents dans les autres clusters que celui dans lequel il se trouve. C'est le terme $b(i)$ qui donne la proximité potentielle d'un élément à l'extérieur.

Soit $s(i)$ le coefficient de silhouette pour un élément i.

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}$$

Nous pouvons aussi écrire la formule avec des conditions comme un système d'équations, c'est ce que nous avons utilisé pour l'implémenter.

$$s(i) = \begin{cases} 1 - \frac{a(i)}{b(i)}, \textbf{ si } a(i) < b(i) \\ 0, \text{ si } a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1, \textbf{ si } a(i) > b(i) \end{cases}$$

Cette valeur doit alors être maximisée pour trouver la valeur de k optimale qui désigne la meilleure classification.

Après avoir conduit nos expérimentations en méthode "script" avec Python, nous avons dû revenir à Java qui est le langage de base sur notre chaine de traitement.

Nous avons alors recherché des équivalents Java aux composants Python que nous avons utilisés. Nous avons sélectionné la bibliothèque MLlib de Spark [MLl, 2015] qui permet de mettre en place un K-Means dans des conditions similaires à nos expérimentations. Pour réaliser la projection de nos matrices de distance dans un espace euclidien, nous n'avons pas trouvé directement le même algorithme. Nous avons utilisé une implémentation Java du MDS proposée par la librairie MDSJ (Multidimensional Scaling for Java) [Pich, 2009]. Cette implémentation fonctionne en deux étapes :

— Un "classical scaling" [Torgerson, 1952] pour fixer des positions initiales en deux dimensions à partir de la matrice de distances.
— Un ajustement du positionnement 2D via une optimisation de la fonction de stress définie par Kruskal [Kruskal, 1964] afin que les distances 2D entre points correspondent presque exactement aux distances de la matrice d'origine.

Suite à la classification des pages d'un même site, nous partons de ces clusters pour constituer des paires de pages comparables. Le but est toujours de pouvoir créer automatiquement l'entrée utilisable par notre algorithme à base de distance d'édition entre arbres. Pour créer les paires, nous procédons comme suit :

— Nous prenons les coordonnées du centroïde de chaque cluster.
— Nous sélectionnons l'échantillon qui se trouve le plus proche du centroïde.
— Nous utilisons cet échantillon comme pivot pour constituer des paires avec tous les autres éléments de la classe. Si l'élément est seul dans sa classe, nous ne le traitons pas et nous le déclarons "outlier".

4.4 Détection de contenu utile

Cette partie est la finalité de notre approche. Nous cherchons le contenu utile avec exactitude. La difficulté fondamentale pour cela est de trouver la juste sélection qui nous permettra d'avoir l'élément minimal qui porte tout le contenu utile. Pour qu'un élément soit minimal, il doit être le plus bas possible dans l'arbre. Nous prenons ce critère pour ne pas prendre trop de contenu. Cependant, nous gardons pour objectif de ne perdre aucun segment du texte intéressant, ce qui signifie ne pas descendre trop bas dans la structure pour ne pas voir le texte être dispersé à travers les branches.

4.4.1 Premières expériences

Nouvelle structure
Nous souhaitons mettre en œuvre notre principe de départ qui consiste à identifier le contenu utile comme le texte qui diffère entre deux pages de mêmes structures. Puisque nous voulons calculer la différence textuelle, nous ciblons les nœuds textes. Ainsi, ce n'est pas une différence en termes de structure HTML que nous voulons mesurer dans le cas présent mais une différence sur le contenu textuel uniquement.
Nous souhaitons garder notre approche utilisant une distance d'édition entre arbres mais elle ne prend absolument pas en compte le contenu des nœuds. Nous modifions donc préalablement les arbres HTML originaux pour une nouvelle structure qui va représenter le contenu textuel sous une forme arborescente dans les arbres originaux. Nous choisissons le token comme segment de représentation parce qu'il n'impose pas

de traitement linguistique particulier tout en étant suffisamment proche du mot pour être significatif. Nous pensons que des traitements linguistiques comme la lemmatisation n'apporteraient pas vraiment d'amélioration pertinente pour notre cas.

Nous procédons alors aux étapes suivantes à partir d'un arbre HTML classique :

— Découpage de chaque nœud textuel de l'arbre d'origine en segments de textes séparés par des espaces et de la ponctuation.
— Chaque token créé devient un nœud dont le parent est le même que le nœud texte dont il est issu.

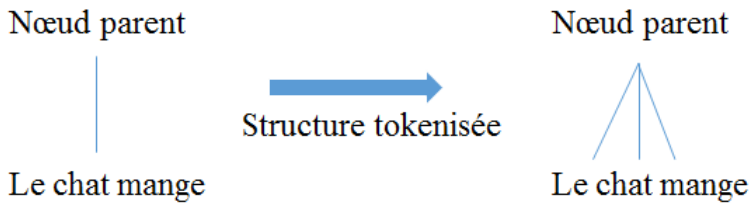


FIGURE 4.3 – Illustration de la nouvelle structure arborescente tokenisée.

Les nœuds tokens se retrouvent être des feuilles dans notre nouvelle structure arborescente. Cette nouvelle structure a pour originalité de prendre le texte comme partie intégrante de la structure de l'arbre dont il est issu. Nous pouvons alors utiliser l'algorithme de distance d'édition entre arbres sur ces nouvelles structures incluant le texte. Dans notre modèle, nous distinguons les nouveaux nœuds spéciaux "Token" des nœuds classiques faisant partie de la structure originale de l'arbre.

Pour quantifier le contenu utile.
Dans le but de trouver l'élément essentiel dans l'arbre, nous voulons quantifier le contenu utile à travers les nœuds. Nous attribuons à chaque nœud un poids qui correspond à sa quantité de texte différent. Les nœuds Token n'ont pas d'attribut de poids contrairement aux nœuds classiques dans la mesure où ce qui nous intéresse est de pondérer les éléments originaux de l'arbre qui contiennent les "tokens" qui diffèrent. Notre algorithme permet d'avoir une trace des opérations réalisées lors de la comparaison entre deux arbres. Il propose un rapport d'édition qui est constitué d'une séquence de couples de nœuds représentant une opération d'édition. Nous supposons que ces associations sont faites sur des nœuds comparables, c'est-à-dire à des positions équivalentes d'un arbre à l'autre. Pour attribuer un poids à un nœud dans l'arbre nous repérons simplement dans le relevé des opérations d'édition le nombre de fois qu'il apparait avec un nœud différent. Chaque token qui diffère augmente le poids de son père d'une unité (algorithme-2). Les poids sont ainsi calculés sur les nœuds textuels directs, sans prendre en compte la hiérarchie et ses inclusions. Nous voulons maintenant faire en sorte que la pondération s'établisse proportionnellement à l'échelle de tout un arbre. Cela revient à rétropropager les poids des nœuds à travers tout l'arbre.
Le parcours de l'arbre se fait du bas vers le haut pour la rétropropagation (algorithme-3). L'élément racine de l'arbre portera bien le poids le plus élevé puisque c'est l'élément le plus englobant.

Première heuristique
Notre heuristique initiale est de considérer que le contenu utile se trouve, dans l'arbre donné, sur un chemin qui part de l'élément avec la plus grande quantité de texte et qui suit ce même critère jusqu'à une feuille. Pour trouver l'élément avec la plus grande quantité de texte, nous mesurons la quantité de texte d'un nœud en nous basant sur le nombre de fils Token qu'il contient directement. L'ordre de parcours dans l'arbre n'est pas important puisque chaque nœud est candidat. Nous devons donc le parcourir entièrement (algorithme-4).

Pour repérer l'élément essentiel.
Nous supposons que le chemin ainsi calculé inclut l'élément minimal qui porte tout le contenu utile. Pour trouver dans un arbre HTML l'élément essentiel, nous pensons que la différence de poids entre les nœuds est significative. Nous définissons alors le différentiel de poids comme la différence la plus petite entre les poids d'un nœud parent et sa descendance immédiate. Nous observons que le différentiel de poids augmente depuis la racine jusqu'à un point culminant. En effet, plus nous sommes haut dans l'arbre plus les éléments avec peu voire pas de tokens sont nombreux. C'est le cas des liens et de divers éléments liés au cadre de la page internet. Nous pensons que l'élément dont le différentiel de poids est maximum est en fait celui qui vient agréger toutes les parties textuelles (algorithme-5). Nous considérons alors que la valeur du différentiel doit être attribuée au père puisqu'elle mesure en fait sa quantité d'information en plus par rapport à son fils.

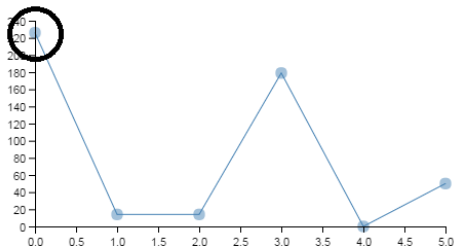
Nouveau départ
Le problème rencontré avec cette première heuristique repose sur le choix de partir de l'élément possédant le plus grand nombre de tokens pour tracer un chemin dans l'arbre. Le chemin sélectionné ne contient pas forcément l'élément qui possède le poids maximal dans l'arbre. En effet, l'élément possédant le plus de tokens peut se trouver à l'identique dans les deux arbres ; ce n'est donc pas du tout celui qui contient l'information essentielle. Nous avons alors un problème de mauvaise localisation portée par un point de départ mal choisi.
Nous avons à cet égard développé une nouvelle heuristique qui part cette fois-ci de l'élément possédant le plus grand différentiel de poids dans l'arbre pour ensuite tracer l'évolution des valeurs de différentiel jusqu'à la racine. Nous visons toujours le maximum de différentiel le long du tracé.
Nous nous rendons compte que nous ciblons parfois trop haut dans l'arbre, emportant des informations annexes au contenu utile.

Heurisitique sur la hauteur dans l'arbre
Nous ajoutons une nouvelle heuristique disant que la zone de contenu utile est basse dans l'arbre. Nous utilisons alors un critère de hauteur dans l'arbre qui est pour un nœud le nombre de niveaux dans l'arbre entre ce nœud et la racine. La valeur ainsi calculée est la suivante :

$$Diff_{pondéré}(\text{nœud}) = Diff(\text{nœud}) \times hauteur(\text{nœud})$$

Nous pondérons maintenant le différentiel par la hauteur, ce qui tend à favoriser ce qui se trouve bas dans l'arbre.
Pour cette heuristique, nous ne partons plus de l'élément avec la plus grande quantité de texte mais nous préférons l'élément qui maximise le différentiel pondéré par la hauteur. Comme pour l'heuristique précédente, nous sélectionnons l'élément qui

affiche un maximum de différentiel à partir du point de départ.
Le problème qui intervient ici concerne le cas des commentaires. En effet, ce sont des quantités de texte différent possiblement plus grandes que le contenu, possédant alors la plus grande valeur de différentiel. De plus, la hauteur des commentaires peut être équivalente à celle du contenu utile.



Le parlement rwandais examinera à l'été les demandes par pétition de près de deux millions de Rwandais pour une réforme de la Constitution qui autoriserait un nouveau mandat présidentiel de Paul Kagame, l'homme fort du pays depuis la fin du génocide de 1994.

Attendue depuis des mois dans la perspective de la présidentielle de 2017, une telle annonce fait écho à un sujet brûlant sur le continent africain, au Burundi voisin, en République démocratique du Congo, au Congo ou encore au Burkina Faso: la modification des textes constitutionnels pour permettre à des chefs d'Etat, parfois au pouvoir depuis des décennies, de se maintenir à la tête de leur pays. "Nous avons reçu autour de deux millions de demandes de la population" pour la révision de l'article 101 de la Constitution portant sur la limitation du nombre de mandats présidentiels, a déclaré Donatilla Mukabalisa, présidente de la chambre des députés.

Les deux chambres composant le parlement -toutes deux dominées par le parti au pouvoir FPR (Front patriotique rwandais)- se prononceront sur les suites à donner à ces pétitions en séance plénière entre le 5 juin et le 4 août, a précisé la présidente de la chambre des députés. Ces demandes, qui représentent près de 17% de la population rwandaise, affluent depuis plusieurs mois au parlement sous la forme de pétitions spontanées, selon Mme Mukabalisa. Elles émanent d'individus, de groupes d'individus ou d'associations qui envoient des lettres ou se présentent en personne, a-t-elle expliqué.

FIGURE 4.4 – Evolution du différentiel (à gauche). L'élément entouré en noir représente le noeud contenant le texte affiché (à droite).

Heuristique sur le code HTML de la page analysée Nous pensons à utiliser la position des nœuds dans le code HTML pour distinguer le texte principal des autres éléments à forte proportion de texte. C'est le critère d'offset : la position d'un élément dans une chaine de caractères. Une grande différence d'offset peut mettre en évidence un point d'agrégation de nombreux éléments. Nous voulons alors suivre la valeur d'offset de chaque élément pendant le parcours de l'arbre. Nous seuillons la sélection au maximum de différence entre les valeurs d'offset. Ce critère n'est pas aberrant mais il ne suffit pas pour déterminer la zone de texte différent qui correspond exactement au corps de l'article.

4.4.2 Heuristique retenue

La proportion de texte
Le corps de l'article est une zone dont la quantité de texte est particulièrement importante. La proportion de texte à travers l'arbre peut alors nous y mener. Pour quantifier le texte, nous utilisons comme unité le nœud token. À chaque élément, nous attribuons le rapport entre son nombre de tokens et le nombre de tokens total sur la page. Nous rétropropageons les valeurs dans l'arbre pour avoir des proportions englobantes. Cela permet, pour chaque élément de la structure, de savoir s'il contient une quantité importante du texte de la page ou non. La racine portera donc assurément une valeur de 1, puisqu'elle inclut tout le texte. Notre méthode est alors d'élaguer les zones qui ne possèdent pas de tokens ou pas assez (par exemple moins de 2% du texte initial) pour nous diriger vers celles qui en possèdent un maximum. Nous parcourons l'arbre de manière à produire un classement des nœuds selon leur proportion de texte.

Sur un positionnement linéaire des éléments
Une forte quantité de texte n'est pas forcément synonyme de pertinence. Nous voulons maintenant faire un nouveau classement qui utilise notre heuristique initiale concernant la structuration standard d'une page d'actualité. Ce nouveau classement est réalisé suivant l'offset. La position dans le code HTML suit la position verticale dans le rendu visuel. C'est un meilleur critère que la hauteur dans l'arbre pour situer un élément comme le verrait un lecteur humain dans son navigateur internet. Nous considérons le premier élément du classement comme étant le titre et le deuxième élément comme étant notre contenu utile. Nous ajoutons que ce deuxième élément doit posséder une proportion de texte plus importante que le premier ; cela permet d'éviter les éventuels éléments placés entre le titre et le corps. Pour ne pas monter trop haut dans la structure, c'est-à-dire récupérer un élément trop englobant qui contiendrait le texte principal avec d'autres contenus, nous nous assurons qu'un élément retenu dans la liste ne doit pas être déjà contenu par un élément visité.

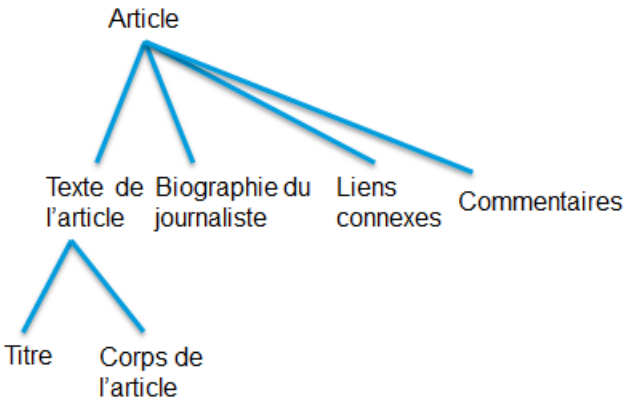


FIGURE 4.5 – Le corps de l'article dans son contexte bruité.

Trouver des nœuds comparables pour la différence textuelle.
Nous comptions sur le rapport d'édition pour comparer les nœuds Token mais il sélectionne parfois des nœuds englobants, ce qui nous empêche de quantifier précisément la différence textuelle. Pour que la différence textuelle soit significative, nous devons la calculer pour deux nœuds comparables. Nous avons alors choisi de mettre en place une méthode pour faire une sélection plus sûre. Deux nœuds comparables doivent être identifiables de la même façon entre les deux arbres. Pour identifier un nœud dans un arbre, nous utilisons le chemin qui le sépare de la racine. XPath est une syntaxe qui permet d'exécuter des requêtes à base de chemin dans un arbre. Nous l'utilisons pour comparer un nœud candidat dans un premier arbre au nœud de même chemin dans l'autre arbre. Pour s'assurer de notre sélection, nous vérifions que les deux nœuds possèdent les mêmes attributs. Ce qui nous intéresse maintenant est de mesurer si le texte d'un élément candidat est utile selon notre définition de départ. Nous jugeons que le texte d'un nœud candidat est d'autant plus utile qu'il diffère de celui appartenant à un nœud comparable.

Afin de comparer les textes contenus par les nœuds appariés, nous avons expérimenté une mesure de distance ensembliste basée sur le caractère. Elle mesure l'intersection de caractères entre deux chaines (algorithme-6). Le caractère n'est pas assez complexe pour décrire la teneur réelle d'un texte. Deux textes occidentaux vont

avoir la même proportion de caractères quelles que soient leurs teneurs propres. Nous sommes passés à l'échelle du mot avec la similarité cosinus appliquée aux deux chaines de caractères à comparer. Elle consiste à mesurer l'écart entre deux textes représentés sous une forme vectorielle. Les vecteurs sont constitués par les mots de la chaine de caractères qu'ils représentent. Il s'agit ensuite de mesurer l'angle entre ces deux vecteurs et calculer son cosinus. Le résultat détermine si les textes sont proches ou distants l'un de l'autre. Nous conservons bien sûr le texte s'il n'est pas identique puisque nous considérons qu'un texte issu de "template" sera strictement le même quelle que soit la page.

Soit θ l'angle entre les documents A et B.

$$\cos\theta = \frac{\vec{A} \cdot \vec{B}}{||\vec{A}||||\vec{B}||}$$

La valeur de $\cos\theta$ est comprise dans l'intervalle [-1. . . 1]. La valeur -1 indique que les vecteurs s'opposent. "0" signifie que les vecteurs sont indifférents l'un de l'autre pusiqu'orthogonaux. "1" désigne une superposition ce qui signifie qu'ils sont équivalents. Ainsi, plus le résultat tend vers 1, plus les documents sont proches tandis que vers -1 les documents se distinguent l'un de l'autre. Dans notre cas, une valeur négative n'a pas de sens puisque nous comparons la présence de mots. Ainsi, nos résultats seront entre 0 pour des pages internet qui n'ont pas du tout les mêmes mots et 1 lorsqu'elles possèdent exactement les mêmes.
La similarité cosinus donne des résultats cohérents. C'est une mesure très présente en TAL, par exemple pour la détection d'opinion. Nous n'avons pas utilisé la distance de Levenshtein, plus traditionnelle pour comparer des textes, parce qu'elle est plus coûteuse à calculer.

Filtrage d'élément
Le texte qui change d'une page à l'autre n'est pas forcément le contenu de l'article. Nous pensons notamment aux commentaires qui peuvent être beaucoup plus grands. Nous avons alors fait en sorte de filtrer la zone de commentaire en éludant les balises identifiées comme pointant typiquement sur ce type de texte (à l'aide d'un attribut de CSS "comment" par exemple). Nous prenons garde à supprimer le cas particulier de l'élément "gallery" ; ce sont les galeries photos qui contiennent beaucoup de texte avec les légendes. Nous observons aussi que notre élément sélectionné contienne un nom de classe et des propriétés CSS pour tenter de s'assurer d'un élément déterminant sur la page.

Nous avons fait en sorte de localiser le plus finement possible l'élément porteur de contenu à l'aide d'heuristiques basées sur la structure des arbres HTML et sur la qualité textuelle. Nous sommes satisfaits parce qu'elles utilisent majoritairement des généralités sur les pages que nous analysons, ce qui confère de la robustesse à notre approche. Cependant, nous craignons que l'association XPath soit rapidement limitée si les paires ne contiennent pas des pages assez semblables.
Nous allons évaluer notre approche le plus objectivement possible pour mettre en évidence les difficultés effectives.

CHAPITRE 5

ÉVALUATION

L'objectif de cette partie est de trouver la bonne façon pour juger la pertinence de notre approche. Nous prendrons un point de comparaison qui est le système Boilerpipe. Nous l'avons choisi parce que c'est le système qui est utilisé sur la chaine de traitement que nous voulons améliorer.
Nous devons simuler un fonctionnement dans des conditions propres à une collecte de pages internet ciblant des sites d'actualité. Nous constituons un corpus d'évaluation avec des données assez proches de celles que notre système pourrait rencontrer dans un tel contexte. Nous allons donc d'abord décrire la constitution de ce corpus. Nous décrirons ensuite notre méthode d'évaluation. Nous présenterons les résultats pour les discuter. Cela sera l'occasion d'initier les perspectives possibles à notre approche.

5.1 Corpus d'évaluation

Nous devons trouver les données sur lesquelles évaluer la précision de notre méthode. Elles doivent être des pages internet telles que nous les trouverions en sortie d'un crawler classique. Nous devons poser le moins de préjugé possible sur les sources sélectionnées. Nous rappelons que notre approche s'appuie sur l'importance prise aujourd'hui par le "template" dans la manière de produire du contenu sur internet. Nous avons besoin de constituer un corpus d'évaluation contenant des pages de l'internet actuel. Puisque notre algorithme vise principalement à traiter les sites d'actualité, nous ne choisissons donc que des sites de ce type. La langue n'est pas une barrière pour notre approche, donc la sélection des sites ne porte pas forcément sur une seule langue. Les données se doivent d'être assez nombreuses et variées pour observer la robustesse de notre système. Nous avons ainsi collecté un corpus multilingue de sites d'actualité.
La récupération des pages internet s'est faite de façon manuelle. Cela permet d'avoir une sélection plus fine des pages de manière à ce qu'elles se situent bien dans le cadre de notre évaluation. À l'intérieur de chaque site, nous avons sélectionné des pages en essayant de varier les genres.

Voici la liste des sites détaillée :

— *lefigaro.fr* : 50 pages, français.
— *lemonde.fr* : 50 pages, français
— *20minutes.fr* : 50 pages, français.
— *atlantico.fr* : 50 pages, français.
— *liberation.fr* : 50 pages, français.

— *metronews.fr* : 50 pages, français.
— *lepoint.fr* : 50 pages, français.
— *bbc.com* : 50 pages, anglais.
— *spiegel.de* : 50 pages, allemand.
— *rainews.it* : 50 pages, italien.

Nous avons pu observer les caractéristiques générales suivantes :

— Taille totale : 50 Mo
— Poids moyen d'une page : 100 Ko
— Nombre de liens moyen : 150
— Nombre de mots moyen : 1500
— Hauteur moyenne de l'arbre : 20
— Largeur moyenne de l'arbre : 50 Nombre de nœuds moyen : 1000

Ce corpus sert de référence pour notre évaluation Pour cela, nous avons dû procéder à l'annotation générale des documents pour indiquer le contenu attendu en sortie du système.

5.2 Résultats

Dans les deux tableaux qui suivent, nous comparons T, le texte trouvé par l'approche évaluée à T^*, le texte de référence. Nous utilisons les paramètres suivants :

— $Sites$, les sites internet utilisés pour l'évaluation.
— $exactitude$, le pourcentage des textes qui recoupent la référence en termes de distance cosinus à au moins 0,9. Ce seuil a été défini de façon empirique en regardant si le texte obtenu est acceptable.
— $cos\theta$, la moyenne des valeurs de similarité cosinus entre T et T^*. Elle permet de seuiller la validité d'un résultat.
— L'égalité stricte soit la proportion de cas où T est égal à T^*.
— L'inclusion stricte de T^* dans T, ce qui signifie que nous avons sélectionné plus haut que le corps de l'article dans l'arbre.
— L'inclusion stricte de T dans T^* ce qui signifie que nous avons extrait à l'intérieur du contenu utile.

Sites	exactitude	$cos\theta$	$T = T^*$	$T \subset T^*$	$T^* \subset T$
20minutes.fr	94%	0.96	46	2	0
atlantico.fr	68%	0.76	34	0	0
bbc.com	92%	0.95	45	3	0
spiegel.de	92%	0.94	45	1	2
lefigaro.fr	80%	0.89	32	8	3
lemonde.fr	88%	0.89	44	0	0
lepoint.fr	92%	0.97	45	3	1
liberation.fr	98%	0.98	49	0	0
metronews.fr	90%	0.94	40	5	0
rainews.it	94%	0.94	37	4	7
total (500pages)	88%	0.92	417	26	13

FIGURE 5.1 – Résultats de notre approche sur notre corpus d'évaluation.

La différence $|T \subset T^*| - |T = T^*|$ montre que dans la plupart des cas nous parvenons à trouver le texte visé mais nous coupons trop haut. Le défaut majoritaire de notre approche est donc de prendre trop d'éléments par rapport au contenu utile. Les cas où nous coupons trop bas correspondent à la valeur $|T^* \subset T|$; ils sont sensiblement moins importants. Cela s'explique parce que le filtrage par liste de balises permet d'éviter les divisions de texte comme les paragraphes. Cela sauve le critère de différence textuelle pour ne pas descendre trop bas, si une division de texte a plus de poids que la division englobant le corps de l'article. Ces deux observations corroborent notre volonté de ne pas perdre du texte essentiel.

Sites	exactitude	$cos\theta$
20minutes.fr	94%	0.95
atlantico.fr	38%	0.68
bbc.com	94%	0.95
spiegel.de	74%	0.86
lefigaro.fr	84%	0.93
lemonde.fr	98%	0.98
lepoint.fr	72%	0.81
liberation.fr	92%	0.95
metronews.fr	88%	0.93
rainews.it	90%	0.94
total (500pages)	82%	0.89

FIGURE 5.2 – Résultats de Boilerpipe sur notre corpus d'évaluation.

La comparaison avec Boilerpipe montre que notre système est globalement meilleur pour viser le texte de référence. Cela montre que nos heuristiques de départ sur la quantité de texte et la mesure de différence sont correctes pour élaguer l'arbre tout en se déplaçant dans la bonne zone.
Le point central qui différencie les deux méthodes est que la nôtre capte les éléments dans l'arbre HTML tandis que Boilerpipe utilise une analyse des éléments de texte. Boilerpipe ne fonctionne pas en divisions dans la page ce qui l'empêche de récupérer exactement le corps de l'article. Nous observons des résultats critiques sur le site *atlantico.fr* pour les deux approches. Ceci s'explique par la présence plus importante de contenus connexes au corps de l'article pour ce cas précis. C'est aussi que le "template" à l'intérieur de ce site est beaucoup plus variable.
Les résultats sur bbc.com avantagent Boilerpipe. Nous pouvons l'expliquer parce que le "template" de ce site internet divise de façon notable le texte du corps de l'article, le rendant plus diffus pour notre approche. Comme Boilerpipe ne prend pas les éléments de structure pour repérer le contenu utile, il est avantagé.

Notre critère d'arrêt en descendant sur les feuilles de l'arbre HTML est un point sensible. Il est encore difficile de déterminer ce qui fait partie du corps de l'article. Des éléments sans rapport direct avec l'article peuvent aussi être pris en compte lors de la segmentation. Hormis les commentaires, nous avons pu observer le cas de publicité ou de présentation d'autres articles qui peuvent eux aussi une quantité importante de texte différent. Cela influe alors sur l'orientation dans l'arbre suivant la quantité de texte et sur la valeur de différentiel. Ce sont ces éléments supplémentaires que l'on retrouve dans les cas de notre évaluation où le contenu utile est présent mais pas repéré de façon minimale.

Nous pensons que la quantité de pages peut influencer nos résultats. Intuitivement, plus les pages sont nombreuses dans le corpus analysé, plus il sera possible de trouver des pages comparables. Or, c'est essentiel pour s'assurer d'une comparaison de textes pertinente, c'est-à-dire entre nœuds de même XPath d'une page à l'autre. Nous pointons ici le cas des "outliers" que nous ne pouvons pas traiter. Nous parlons dans ce cas de "méthode endogène" pour désigner un système qui dépend du corpus pour obtenir des résultats corrects.
Nous diminuons le nombre de pages en supposant que les résultats vont ainsi diminuer.

Sites	exactitude	$cos\theta$	$T = T^*$	$T \subset T^*$	$T^* \subset T$
20minutes.fr	100%	1.0	5	0	0
atlantico.fr	60%	0.67	3	0	0
bbc.com	100%	0.99	4	1	0
spiegel.de	100%	1.0	5	0	0
lefigaro.fr	80%	0.9	2	2	1
lemonde.fr	100%	1.0	5	0	0
lepoint.fr	60%	0.76	3	1	0
liberation.fr	100%	1.0	5	0	0
metronews.fr	100%	1.0	5	0	0
rainews.it	100%	0.99	3	0	2
total (50pages)	90%	0.93	40	4	3

Nous observons une tendance générale identique à travers les sites, ce qui est positif pour la cohérence de notre méthode.
Nous augmentons les résultats sur une majorité de sites. Nous pensons qu'ils présentent un "template" majoritairement similaire. En diminuant leur nombre de pages, nous favorisons la chance de supprimer les "outliers". À l'inverse, pour les quelques sites dont les résultats diminuent, c'est que leur "template" est plus propice à une variation de leur structure, ce qui baisse la proportion de pages comparables dans le corpus.
Le nombre de pages influence effectivement nos résultats mais pas forcément de façon négative. C'est selon la constitution interne des sites analysés. Ces résultats constituent une bonne mesure sur la façon dont sont constitués les sites internet du corpus.

Dans la section suivante, nous allons discuter les résultats obtenus pour déterminer les failles et apports de notre approche.

5.3 Discussion

Nous tenons à nuancer les résultats de notre évaluation. Même si nous avons fait des efforts pour varier les types de page à l'intérieur d'un même site, notre corpus d'évaluation n'est certainement pas représentatif de tout l'internet mais il se veut être un échantillon caractéristique de pages issues de sites journalistiques. Notre jeu de données a été construit manuellement. Nous avons en vue de récupérer les pages de façon automatisée par un crawler, tel que cela se passerait en fonctionnement véritable.

5.3.1 Avantages

Indépendance
Notre approche n'a pas besoin d'intervention humaine pour fonctionner. La méthode de classification n'est pas supervisée, elle n'a pas besoin que nous lui indiquions les classes à trouver. Cela nous évite d'avoir à connaitre les données en amont, ce qui est particulièrement profitable dans un contexte de crawl depuis l'internet. La localisation de la zone de contenu utile n'est pas fondée sur de l'apprentissage. C'est un avantage conséquent de ne pas avoir à constituer de données annotées pour enrichir le système. Cet avantage est nuancé par les différentes propriétés sur les pages internet que nous devons spécifier.
L'avantage pour notre approche de ne comporter aucun traitement linguistique la rend indépendante de En effet, nous ne prenons en compte que la forme des pages pour localiser le contenu utile. Puisque nous avons spécifié le corpus sur lequel nous travaillons, nous avons déterminé que l'allure standard d'une page d'actualité n'est pas liée à la langue de l'article. C'est un atout notable parce qu'il n'est pas aisé d'identifier la langue des pages internet collectées. Notre approche peut ainsi s'intégrer dans des projets traitant de nature variée.

Spécificité
Elle prend le parti d'être spécifique pour une certaine catégorie de sites que sont les sites d'actualité. C'est une qualité de pouvoir se déterminer ainsi puisqu'en se focalisant sur un sous-ensemble des pages internet, nous pouvons développer des règles plus spécifiques qui permettent de gagner en exactitude avec la même généricité. L'évaluation est plus aisée dans l'optique de constituer un corpus de tous les cas naturels. La définition du contenu utile est plus simple sur un type de corpus bien spécifié. En limitant les types de page que nous pouvons rencontrer, nous réduisons la complexité des règles à définir pour être assez générique. Cela limite les cas possibles de structure que nous pouvons rencontrer. Or, la structure est déterminante pour les heuristiques que nous avons conçues. Cela nécessite que le crawler en amont de notre système puisse identifier le genre des sites qu'il visite.

5.3.2 Inconvénients

Condition d'arrêt
Le critère d'arrêt de notre algorithme reste un point difficile à définir. Le fait de devoir descendre "ni trop haut ni trop bas" dans les feuilles de l'arbre HTML est particulièrement difficile à contrôler. Les heuristiques mises en place se concentrent sur des quantités de texte. Nous pensons qu'une analyse linguistique des textes candidats serait utile pour trouver le contenu utile. Nous pourrions aussi faire un apprentissage sur les offsets et valeurs de différentiels pour la zone de contenu utile. L'utilisation des attributs CSS pour un apprentissage pourrait aussi nous aider à mieux localiser le corps de l'article.
De façon générale, sur deux pages dont la structure est comparable, la majorité des variations intervient précisément proche de la zone où se trouve l'élément minimal qui porte notre contenu utile. Cela correspond à la partie basse de la structure. C'est pourquoi il est difficile de trouver cet élément une fois l'arbre élagué grossièrement. De plus, le seuil de pourcentage de texte pour définir une zone de texte importante est fixée de façon empirique. Nous pourrions travailler à fixer ce pourcentage par une maximisation des résultats.

Robustesse technique
Le calcul de la distance d'édition entre arbres HTML est particulièrement coûteux puisqu'il doit envisager les différentes associations de nœuds. C'est une complexité de l'ordre $n \times m$ avec n et m les nombres de nœuds respectifs des deux arbres HTML analysés. C'est une indication importante pour comprendre le temps de calcul élevé de notre système.
La construction d'une matrice de distances est coûteuse parce qu'elle inclut le calcul de la distance d'édition entre arbres pour toutes les associations de pages dans le corpus étudié. Cela constitue une complexité de l'ordre n^2 avec n le nombre de pages dans le corpus. C'est en même temps une étape que nous ne pouvons pas réduire parce que nous avons besoin des distances entre toutes les pages d'un même site pour que la classification puisse fonctionner correctement.

Limite heuristique
Le cadre structurel que nous avons défini dans notre heuristique, tel qu'une page présente un titre puis un corps d'article, n'est pas forcément valable à travers les différentes ressources. L'heuristique avançant que le titre est plus petit que le corps de l'article est aussi un point qui pourrait casser notre approche si nous rencontrons une situation particulière.
L'association en paires de pages suppose un squelette identique. Le point important de notre système est de toujours pouvoir comparer des éléments comparables. L'utilisation d'XPath impose que les structures soient absolument identiques ; le chemin peut être annulé au moindre changement dans la constitution des structures.
Ainsi, nous ne pouvons pas traiter les cas de pages "singletons", celles qu'il n'est pas possible de lier dans une paire. Nous avons besoin de comparer pour trouver. Nous devons surtout être en capacité de les identifier pour ne pas les comparer à des pages différentes, ce qui nous donnerait des mauvais résultats.
Le fait d'utiliser des listes de balises spécifiques est très peu robuste dans la mesure où elles ne s'appliquent qu'à certains cas. Elles permettent d'améliorer très largement les résultats sur des sites particuliers mais cela peut aussi les rendre moins bons sur d'autres.
La caractérisation linguistique du texte est une perspective pour pallier la limite heuristique.

Déroulement des travaux
Le fait d'avoir utilisé une librairie déjà codée pour calculer les distances d'édition entre arbres est pratique pour lancer rapidement des expérimentations. Intégrer un modèle extérieur n'est pas évident parce qu'il ne fait pas forcément ce que nous voulons ou pire, ce que nous pensons qu'il fait. Nous avons été bloqués par l'association entre les arbres. Notre principe d'intégrer du texte dans les nœuds de l'arbre n'était pas fait pour la librairie. Lors de l'association, l'algorithme utilisait parfois des nœuds englobant le texte. Ce dernier n'était alors pas pris en compte pour la pondération des arbres. C'est pour cette raison que nous avons par la suite utilisé XPath. Nous pourrions essayer de coder nous-mêmes l'association entre arbres de façon à bien prévoir l'aspect textuel en structure.
La construction des arbres en mémoire par la librairie RTED est très lourde voire trop lourde pour certaines machines que nous avons utilisées. Notre système peut ainsi ne pas fonctionner normalement sur des machines dont la mémoire n'est pas suffisante. Nous avons essayé de procéder récursivement par découpage des arbres pour réduire leur taille mais cela influait ensuite les comparaisons. L'algorithme n'est

pas, à la base, prévu pour prendre en compte du texte. Notre choix original d'inclure le texte en tant que nœuds dans l'arbre peut avoir alourdi le traitement. Cependant, nous avons pu constater que même en se limitant à la structure originale, certains arbres sont impossibles à créer par le programme. Cela nous empêche de traiter certains cas.

Instabilité
La classification peut apporter du bruit dès la constitution des paires de pages. Notre approche fonctionne en chaine, ce qui signifie que chaque composant qui apporte un résultat bruité nuit d'autant plus au résultat final. Le MDS d'abord, lors de la projection des matrices de distances, crée du bruit qui peut modifier quelque peu les positions de nos échantillons.
L'algorithme K-Means induit de l'aléatoire dans le système à cause de son étape d'initialisation, ce qui ne donne pas forcément les mêmes paires à chaque lancement.

Conclusion

Dans ce mémoire, nous avons traité la problématique d'identification d'un certain contenu sur une page d'actualité. C'est le contenu utile ; nous l'avons défini comme le corps de l'article. C'est un travail d'autant plus passionnant que le besoin exprimé est tout à fait concret.
Cela peut sembler trivial si nous omettons que le code HTML est semi-structuré. Nous ne pouvons pas simplement nous appuyer sur la structure d'une page pour repérer son contenu utile. Les développeurs restent libres dans la manière d'agencer leurs pages.
Nous avons observé que le contenu utile est l'élément qui change entre deux pages de mêmes nature et thématique. Nous avons ainsi choisi de nous fonder sur une comparaison de pages internet pour identifier le contenu utile.

C'est par l'essai que nous avons choisi d'améliorer ce qui a déjà été fait sur le sujet. Nous avons procédé par itérations successives pour essayer nos différentes intuitions. Beaucoup de nos choix se sont construits de façon empirique. Nous avons travaillé sur des corpus de développement étroitement liés à la succession de nos heuristiques. Nos jeux de données devaient être assez équilibrés en taille et variété pour être représentatifs d'un contexte normal d'utilisation pour notre approche.
Il ne s'agissait pas seulement de définir une règle qui nous donnerait sans faillir le chemin vers l'élément désiré.
C'était d'abord la simple quantité de texte qui n'était pas suffisante pour identifier le contenu utile. Ensuite, le repérage d'un différentiel de poids textuel pour localiser la zone de plus forte agrégation ne faisait pas forcément ressortir le contenu utile mais des éléments au texte potentiellement plus grand, comme les commentaires. Nous avons alors utilisé la sémantique des balises pour filtrer les cas critiques remarquables. Cette liste permettait aussi de ne pas descendre trop bas dans la structure en évitant les divisions de texte à l'intérieur de la zone englobant tout le contenu utile. Le problème était ensuite de situer exactement la zone de contenu utile, sans prendre du contenu supplémentaire. Nous avons utilisé un critère de position avec une heuristique posée sur l'organisation du code HTML pour une page d'actualité standard. Cela permettait de définir un critère pour de ne pas récupérer le contenu juste au-dessus du corps de l'article, tel le titre.

Avec une moyenne de qualité à 88%, les résultats ont montré que nous gardions bien le contenu utile entier, validant notre méthode d'arrêt lors de la descente en structure. Il était essentiel pour nous de ne pas perdre du contenu utile. Les cas où nous ne récupérons qu'un sous-texte du contenu utile sont minoritaires parce que nous filtrons les divisions textuelles.
Le problème récurrent qui demeure est celui de prendre trop de contenu autour du contenu utile. Il est difficile pour notre approche de savoir jusqu'à quel point agglomérer du texte.

D'un point de vue technique, la robustesse mémoire du programme est encore une faille que nous n'avons pas réussi à surmonter. Nous n'avons pas traité le cas du contenu dynamique qui se charge dans le navigateur. Dans ce cas, le texte n'est pas directement inclus dans le code HTML extrait. Cela nécessiterait de simuler le rendu d'un navigateur internet sur la page à contenu dynamique. Nous pourrions utiliser pour cela un outil comme Selenium [Sel, 2015].

D'un point de vue scientifique, le critère d'arrêt pour découper exactement l'élément minimal porteur du contenu visé reste encore compliqué à définir. Pour faire face au bruit apporté possiblement par des erreurs du clustering ainsi que pour être capable de retourner un résultat pertinent sur les cas de pages trop à l'écart du reste de leur cluster, nous pourrions ajouter un système de vote qui mettrait en avant la balise la plus retournée pour le cluster courant. Il serait sûrement efficace d'essayer de trouver le pourcentage seuil qui permettrait de maximiser nos résultats. D'autres paramètres empiriques pourraient être utilisés comme ceux qui ont trait au rendu graphique de la page.
Des analyses linguistiques pourraient peut-être nous aider pour composer avec nos chiffres.

Notre approche peut s'ouvrir à d'autres genres de sources. Ne plus se limiter aux sites d'actualité va logiquement complexifier la modélisation du problème et peut-être les solutions à envisager.
Nous pourrions par la suite chercher à distinguer avec plus de finesse les types de contenu à récupérer. Nous nous sommes focalisés sur le corps d'article dans une page d'actualité. Par exemple, la récupération des métadonnées est une fonctionnalité pertinente pour la chaine de traitement dans laquelle s'inscrit ce travail. Pouvoir dire que nous choisissons de récupérer le titre de l'article, son chapeau, voire exactement les commentaires est très important dans l'optique de structurer l'information. Cela nécessite des éléments de sémantique sur les contenus récupérés.

La détection de contenu utile peut servir à l'amélioration du confort de navigation pour l'internaute, à l'instar des bloqueurs de publicité. En effet, envisagé comme un outil "Assistant" intégré au navigateur (plugin), notre approche servirait de filtre sur une page internet. Par exemple, le site internet *lemonde.fr* propose un "mode zen" qui permet une visualisation de la page d'un article en n'affichant que son texte. Le Web 3.0 semble fait pour notre problématique, ce qui montre la pertinence de l'un comme de l'autre. Nous pourrions penser que la sémantisation de l'internet rend caduque notre problématique mais c'est au contraire un moyen de l'élever. Gageons que cela nous permettra d'aller encore plus loin dans la granularité de ce que nous pourrons détecter.

BIBLIOGRAPHIE

[Dru, 2015] (2015). Drupal cms. http://drupalfr.org/. Accessed: 2015-09-14. – Cité page 11.

[Dub, 2015] (2015). Dublin Core ontologie. http://dublincore.org/. Accessed: 2015-09-21. – Cité page 15.

[MLl, 2015] (2015). MLlib machine learning library. http://spark.apache.org/docs/latest/mllib-guide.html. Accessed: 2015-09-07. – Cité page 34.

[Sel, 2015] (2015). Selenium browser automation. http://www.seleniumhq.org/. Accessed: 2015-09-10. – Cité page 50.

[FOA, 2015] (2015). SKOS ontologie. http://www.foaf-project.org/. Accessed: 2015-09-21. – Cité page 15.

[SKO, 2015] (2015). SKOS ontologie. http://www.w3.org/2004/02/skos/. Accessed: 2015-09-21. – Cité page 15.

[Wor, 2015] (2015). WordPress cms. https://fr.wordpress.com/. Accessed: 2015-09-14. – Cité page 11.

[Ackoff, 2010] Ackoff, R. L. (2010). From data to wisdom. *Journal of applied systems analysis*, 16:3–9. – Cité pages 19 et 20.

[Baluja, 2006] Baluja, S. (2006). Browsing on small screens: recasting web-page segmentation into an efficient machine learning framework. In *Proceedings of the 15th international conference on World Wide Web*, pages 33–42. ACM. – Cité page 21.

[Bar-Yossef and Rajagopalan, 2002] Bar-Yossef, Z. and Rajagopalan, S. (2002). Template detection via data mining and its applications. In *Proceedings of the 11th international conference on World Wide Web*, pages 580–591. ACM. – Cité page 23.

[Cai et al., 2003] Cai, D., Yu, S., Wen, J.-R., and Ma, W.-Y. (2003). Vips: a vision-based page segmentation algorithm. Technical report, Microsoft technical report, MSR-TR-2003-79. – Cité page 21.

[Chakrabarti et al., 2007] Chakrabarti, D., Kumar, R., and Punera, K. (2007). Page-level template detection via isotonic smoothing. In *Proceedings of the 16th international conference on World Wide Web*, pages 61–70. ACM. – Cité page 24.

[Chakrabarti et al., 2008] Chakrabarti, D., Kumar, R., and Punera, K. (2008). A graph-theoretic approach to webpage segmentation. In *Proceedings of the 17th international conference on World Wide Web*, pages 377–386. ACM. – Cité page 21.

[Chen et al., 2006] Chen, L., Ye, S., and Li, X. (2006). Template detection for large scale search engines. *Proceedings of the 2006 ACM symposium on Applied computing - SAC '06*, page 1094. – Cité page 24.

[Chen et al., 2003] Chen, Y., Ma, W.-Y., and Zhang, H.-J. (2003). Detecting web page structure for adaptive viewing on small form factor devices. In *Proceedings of the 12th international conference on World Wide Web*, pages 225–233. ACM. – Cité page 20.

[Crescenzi et al., 2001] Crescenzi, V., Mecca, G., and Merialdo, P. (2001). Roadrunner: Towards automatic data extraction from large web sites. *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 109–118. – Cité pages 23 et 27.

[Evert, 2008] Evert, S. (2008). A lightweight and efficient tool for cleaning web pages. *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, pages 3489–3493. – Cité page 25.

[Fernandes et al., 2011] Fernandes, D., de Moura, E. S., da Silva, A. S., Ribeiro-Neto, B., and Braga, E. (2011). A site oriented method for segmenting web pages. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 215–224. ACM. – Cité pages 21 et 27.

[Gibson et al., 2005] Gibson, D., Punera, K., and Tomkins, A. (2005). The volume and evolution of web page templates. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 830–839. ACM. – Cité page 14.

[Gibson et al., 2007] Gibson, J., Wellner, B., and Lubar, S. (2007). Adaptive web-page content identification. *Proceedings of the 9th annual ACM international workshop on Web information and data management - WIDM '07*, page 105. – Cité page 25.

[Gottron, 2008] Gottron, T. (2008). Content code blurring: A new approach to Content Extraction. *Proceedings - International Workshop on Database and Expert Systems Applications, DEXA*, pages 29–33. – Cité page 25.

[Kao et al., 2005] Kao, H.-Y., Ho, J.-M., and Chen, M.-S. (2005). Wisdom: Web intrapage informative structure mining based on document object model. *Knowledge and Data Engineering, IEEE Transactions on*, 17(5):614–627. – Cité page 24.

[Kohlschütter et al., 2010] Kohlschütter, C., Fankhauser, P., and Nejdl, W. (2010). Boilerplate detection using shallow text features. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 441–450, New York, NY, USA. ACM. – Cité pages 15 et 26.

[Kohlschütter and Nejdl, 2008] Kohlschütter, C. and Nejdl, W. (2008). A densitometric approach to web page segmentation. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 1173–1182. ACM. – Cité page 21.

[Kruskal, 1964] Kruskal, J. (1964). Multidimensional sclaing by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27. – Cité page 34.

[Kushmerick, 2000] Kushmerick, N. (2000). Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1):15–68. – Cité page 22.

[Levenshtein, 1966] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. – Cité page 29.

[Marek, 2007] Marek, M. (2007). Web page cleaning with conditional random fields. *Building and Exploring Web ...*, 5:1–8. – Cité page 25.

[Muslea et al., 2001] Muslea, I., Minton, S., and Knoblock, C. a. (2001). Hierarchical Wrapper Induction for Semistructured Information Sources. *Autonomous Agents and Multi-Agent Systems*, 4(1-2):93–114. – Cité page 22.

[Pasternack and Roth, 2009] Pasternack, J. and Roth, D. (2009). Extracting article text from the web with maximum subsequence segmentation. *Proceedings of the 18th international conference on World wide web - WWW '09*, page 971. – Cité pages 15 et 25.

[Pawlik and Augsten, 2011] Pawlik, M. and Augsten, N. (2011). Rted: a robust algorithm for the tree edit distance. *Proceedings of the VLDB Endowment*, 5(4):334–345. – Cité page 30.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. – Cité page 32.

[Pich, 2009] Pich, C. (2009). *Applications of Multidimensional Scaling to Graph Drawing*. PhD thesis, Universität Konstanz, Konstanz. – Cité page 34.

[Shannon and Weaver, 1949] Shannon, C. E. and Weaver, W. (1949). The mathematical theory of information. – Cité pages 19 et 20.

[Sun et al., 2011] Sun, F., Song, D., and Liao, L. (2011). DOM Based Content Extraction via Text Density. *Technology*, l:245–254. – Cité pages 15 et 26.

[Torgerson, 1952] Torgerson, W. (1952). Multidimensional scaling: I. Theory and method. *Psychometrika*, 17(4):401–419. – Cité page 34.

[Vieira et al., 2006] Vieira, K., da Silva, A. S., Pinto, N., de Moura, E. S., Cavalcanti, J., and Freire, J. (2006). A fast and robust method for web page template detection and removal. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 258–267. ACM. – Cité page 23.

[Weninger et al., 2010] Weninger, T., Hsu, W. H., and Han, J. (2010). CETR: content extraction via tag ratios. *Proceedings of the 19th international conference on World wide web*, pages 971–980. – Cité pages 15 et 26.

[Yi et al., 2003] Yi, L., Liu, B., and Li, X. (2003). Eliminating noisy information in web pages for data mining. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 296–305. ACM. – Cité page 23.

ANNEXES

$n = longueur(chaine1)$
$m = longueur(chaine2)$
if $n = 0$ **then**
  return m and exit
else
  $m = 0$
  return n and exit
end
$matrix \leftarrow matrix[n+1][m+1]$ **for** $0 < i < n$ **do**
  $matrix[n][0] = i$
end
for $0 < j < m$ **do**
  $matrix[0][j] = j$
end
for $1 < i < n$ **do**
  for $1 < j < n$ **do**
    if $chaine1[i] = chaine2[j]$ **then**
      $co\hat{u}t = 0$
    else
      $co\hat{u}t = 1$
    end
    $matrix[i][j] =$
    $minimum(matrix[i-1][j]+1, matrix[i-1][j]+1, matrix[i-1][j-1]+co\hat{u}t)$
  end
end
return matrix[n][m]

Algorithm 1: Calcul de la distance de Levenshtein

input : Le rapport d'édition entre deux arbres, les deux arbres utilisés.
foreach *couple* c *de nœuds dans le rapport d'édition* r **do**
  if $c[0]$ *est un nœud Token* **then**
    $poids(père(\text{c[0]})) + +$
  end
  if $c[1]$ *est un nœud Token* **then**
    $poids(père(c[1])) + +$
  end
end

Algorithm 2: Pondération des nœuds dans les arbres selon leur différence textuelle.

input : Un arbre dont on veut rétro-propager les poids inscrits.
foreach *nœud* n *dans l'arbre* **do**
  if n *est un nœud original* **then**
    $poids(n) \leftarrow somme(poids(fils(n)))$
  end
end

Algorithm 3: Rétropropagation des poids d'un arbre.

input : L'arbre dont on veut trouver l'élément minimal porteur de la plus grande quantité de texte
output: L'élément minimal porteur de la plus grande quantité de texte.
$maxCourant \leftarrow 0$
$majToken \leftarrow nul$
foreach *nœud* n *dans l'arbre* **do**
  $cmpText \leftarrow 0$
  foreach *fils* f *du nœud* n **do**
    if f *est de type Token* **then**
      $cmpt_text + +$
    end
  end
  if $cmpt_text > max_courant$ **then**
    $majToken \leftarrow n$
  end
end
return matrix[n][m]

Algorithm 4: Trouver l'élément minimal porteur de la plus grande quantité de texte dans un arbre.

input : Arbre pondéré en quantité de texte
output: L'élément possédant le contenu textuel essentiel de l'arbre.
$maxDiff \leftarrow 0$
$majorNode \leftarrow nul$
foreach *nœud* n *dans l'arbre* **do**
  if *n est un nœud structurel* **then**
    $diff(père(n) \leftarrow poids(père(n)) - poids(n))$
    if $diff(père(n)) > maxDiff$ **then** $majorNode \leftarrow père(n)$;
    ;
  end
end
return majorNode

Algorithm 5: Trouver l'élément porteur dans un arbre pondéré par un calcul de différentiel de poids.

tableau alphabet $\leftarrow \{abcdefghijklmnopqrstuvwxyz\}$
HashMap index1 <caractère,valeur>
HashMap index2 <caractère,valeur>
for $0 < entieri < longueur(alphabet)$ **do**
  $caractèreCourant \leftarrow alphabet[i]$ $index1$[caractèreCourant] $\leftarrow 0$
  $index2$[caractèreCourant] $\leftarrow 0$
end
for $0 < entieri < longeur(chaine1)$ **do**
  caractèreCourant $= chaine1[i]$ **if** *index1 contient caractèreCourant* **then**
    $index1$[caractèreCourant] $= valeurcourante + 1$
  end
end
for $0 < entieri < longeur(chaine2)$ **do**
  caractèreCourant $= chaine2[i]$ **if** *index2 contient caractèreCourant* **then**
    $index2$[caractèreCourant] $= valeurcourante + 1$
  end
end
somme $\leftarrow 0$
compteur $\leftarrow 0$
foreach $Caractèrecdansl'alphabet$ **do**
  $maximum \leftarrow max(index1[c], index2[c])$ **if** $!(index1[c] = 0 AND index2[c] = 0)$
  then
    $cmpt++$
  end
  if $maximum == 0$ **then**
    $somme = 0$
  else
    $somme = somme + 1 - \frac{|index1[c] - index2[c]|}{maximum}$
  end
end
if $cmpt = 0$ **then**
  $return 0$
else
  $return \frac{somme}{cmpt}$
end

Algorithm 6: Comparaison de textes en proportion de caractères.