Extraction du vocabulaire spécifique à partir d'un corpus web sélectionné pour un nuage de mots

Mikaël Morardo

UNIVERSITÉ Stendhal GRENOBLE 3

Extraction du vocabulaire spécifique à partir d'un corpus web sélectionné pour un nuage de mots

Nom : Morardo
Prénom : Mikaël

UFR Sciences du langage

Mémoire de master 2 professionnel - 20 crédits – Sciences du langage
Spécialité ou Parcours : Modélisation et traitements automatiques en industrie de la langue - Traitement Automatique de la Langue Écrite et de la Parole
Sous la direction de M. Lebarbé

Année universitaire 2009-2010

Remerciements

Je tiens à remercier en premier lieu mon tuteur entreprise Nils Grunwald pour les enseignements et l'aide qu'il m'a apporté tout au long du stage. Sa disponibilité et son amabilité auront été une des clés du bon déroulement de celui-ci.
Je remercie l'équipe des développeurs de Linkfluence : Camille Maussang, Nicolas M., Franck Cuny, Germain Maurice, Antonin Rohmer, Stéphane Raux et Rémi Damlencour pour le support dont ils ont fait preuve et leur bonne humeur tout au long de ces six mois.
Je remercie aussi Guillem Fouetillou et Alain Le Berre pour m'avoir accordé leur confiance et m'avoir accueilli au sein de leur entreprise.
Je souhaite remercier l'ensemble du pôle étude de Linkfluence avec qui j'ai travaillé indirectement, et plus directement en quelques occasions lors du stage et avec qui le contact a toujours été très sympathique.
Je remercie enfin mon tuteur universitaire Thomas Lebarbé qui aura suivi mon évolution au cours de ces mois et qui m'aura prodigué ses conseils pour réussir et appréhender correctement le monde professionnel, ainsi que Georges Antoniadis, directeur du Master qui aura accepté ma candidature à l'université Stendhal de Grenoble dans l'urgence et donc permis la réalisation de ce stage par la suite.

Table des matières

Introduction

Le présent rapport va mettre en lumière les solutions que nous avons apportées à l'extraction d'un vocabulaire spécifique à partir d'un corpus issu du web dans l'optique de l'afficher au moyen d'un nuage de mots. Au cours des six mois de déroulement du stage au sein de l'entreprise Linkfluence, nous avons été confronté à divers choix et problèmes que nous avons résolus suivant les méthodes que nous allons présenter et éclaircir ici. Nous avons dû aussi répondre à un certains nombre de critères établis par Linkfluence.
Parmi les spécificités des algorithmes mis en place, le calcul des n-grammes tient une part prépondérante dans notre système pour permettre l'affichage d'entités nommées complètes mais aussi d'expressions figées ou de mots composés.
Nous détaillerons le système de grammaires qui permet la sélection d'un type de n-gramme particulier, répondant aux critères que nous choisirons et qui permet d'exclure les formes dépourvues d'intérêt sémantique.
Nous aborderons notre manière de valoriser les entités nommées à l'aide d'une liste d'autorité fondée sur Wikipedia.
Le rapport décrira aussi en détail le système de calcul de score développé ainsi que le processus de fusion que nous avons établi afin de minimiser la redondance et le bruit lors de l'affichage des données en nous appuyant sur la distance de Levenshtein et différents moyens de pondération selon la fréquence d'apparition.
Des résultats seront proposés ainsi que leur critique et évaluation afin de garder un œil objectif sur la pertinence de ces derniers. Enfin, nous reviendrons sur l'ensemble du stage d'un point de vue global afin d'en tirer les enseignements qu'il nous a apporté, les choses qu'il nous a fait découvrir et l'avenir qui se profile.

Partie 1 : Gestation et naissance d'un nouveau projet

1 Environnement et cadre professionnel

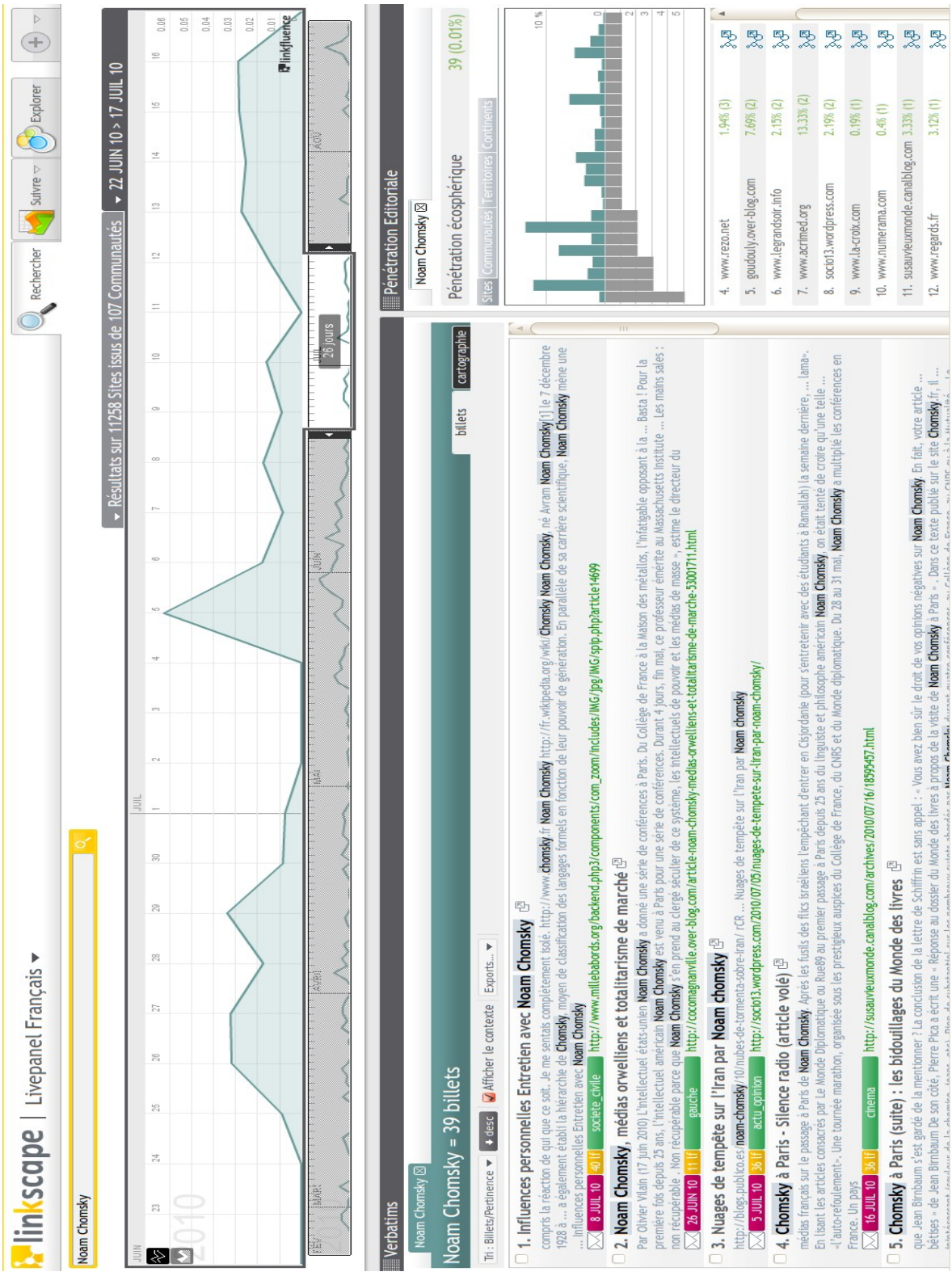
Au sommet de la tour

Le stage s'est déroulé dans le cadre d'un master 2 professionnel en modélisation et traitements automatiques en industries de la langue. Il a pris place dans la société Linkfluence située sur Saint-Denis, carrefour Pleyel dans la tour du même nom. Linkfluence est une jeune entreprise actuellement en expansion, elle compte une vingtaine de salariés. Structurellement, Linkfluence possède deux pôles, l'un consacré aux études qu'elle réalise et vend, l'autre au développement et à la maintenance de ses outils d'exploration du web. Étant donnée la nature du stage, il s'est naturellement ancré dans le pôle développement bien qu'il ne fut pas exempt d'interactions avec le pôle études, notamment pour répondre à des besoins et proposer des résultats exploitables.
Pour mener à bien la mission qui nous a été confiée à travers le stage, Linkfluence a mis à notre disposition les outils qu'elle a développés ainsi que les données qu'elle a récoltées et accumulées au fur et à mesure de son avancée. En plus du tuteur entreprise, l'équipe des développeurs s'est montrée disponible pour répondre à tout besoin et faciliter les démarches aussi bien conceptuelles que techniques.

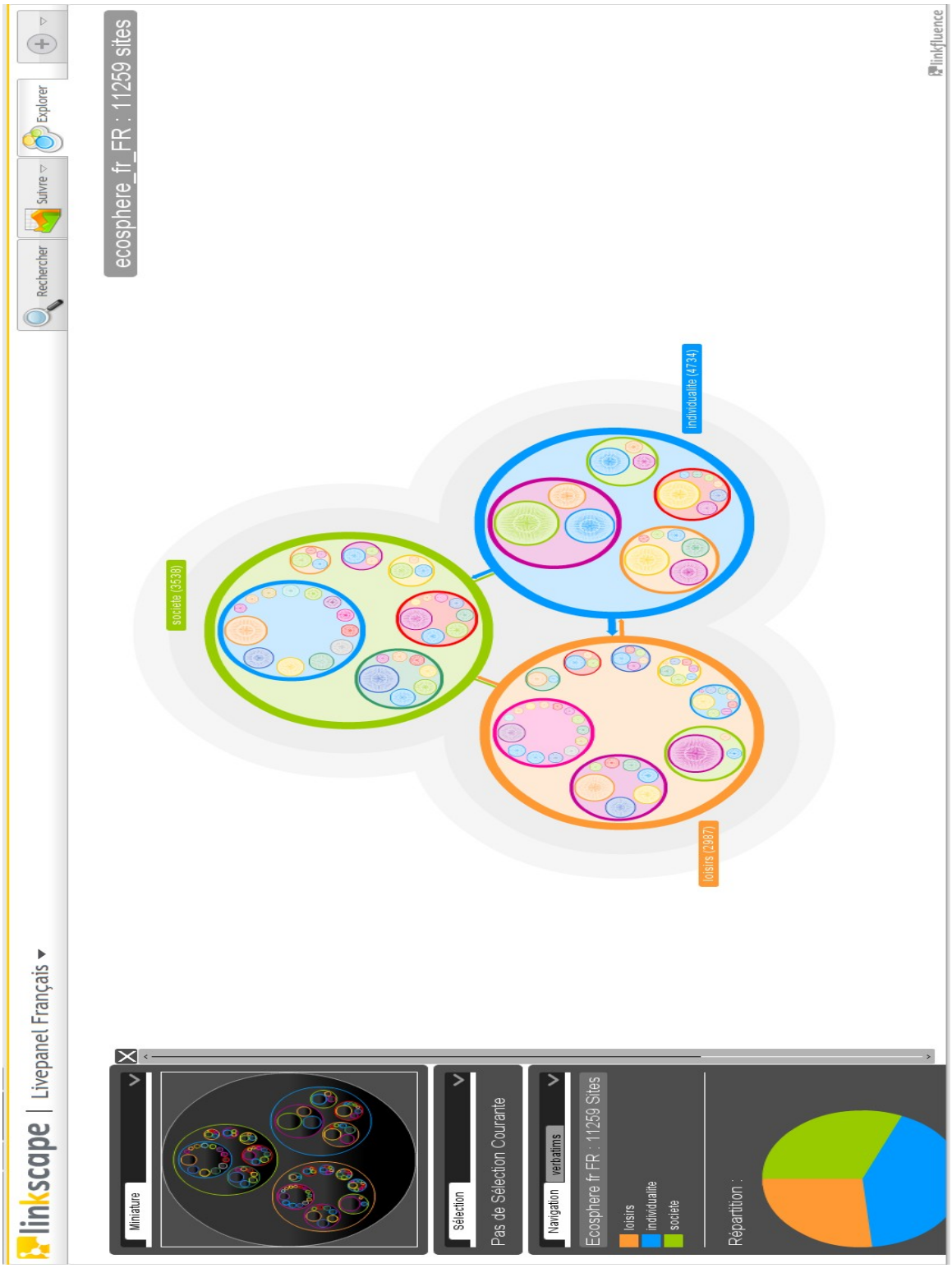
Linkfluence, Linkscape

Aujourd'hui, le web est un fourmillement de sites, blogs et autres qu'il serait bien difficile de dénombrer compte tenu de leur cycle de vie. Le volume de publications étant sans cesse croissant, suivre l'ensemble des discussions tenues sur ce jeune média n'apportera qu'une cacophonie dont il sera difficile d'extraire de l'information pour l'analyser. Cependant, si un choix avisé dans l'écoute des différents acteurs du web est fait, il devient alors envisageable de construire un échantillon représentatif des discussions qui ont lieu. Toute la finesse du processus est alors concentrée dans le choix stratégique des sites qualifiés d'*influenceurs* majeurs au sein de leur communauté.
D'après les travaux de [Jon Kleinberg, 1999], la structure géographique du web reposant sur les liens hypertextes a eu pour effet de cloisonner en partie les différentes communautés. Le constat est le suivant, un site (ou blog) s'incrustant dans une thématique précise aura tendance à davantage développer ses liens vers d'autres sites partageant les mêmes affinités. De manière inconsciente, les diverses communautés ont ainsi elles-mêmes posé leurs propres frontières topologiques. Kleinberg propose alors l'hypothèse qu'un site A disposant d'un lien vers un site B confère à ce dernier une forme d'autorité intellectuelle justifiée par ce même lien. Si A lie B, c'est parce que A reconnaît la valeur de B et par cette même reconnaissance confère à B un poids supplémentaire dans la hiérarchie communautaire dans laquelle s'inscrivent A et B. De la même manière qu'un auteur d'article en citant un confrère pour appuyer ses dires va indirectement conféré à ce dernier une forme d'autorité (sous entendu que le contexte de citation n'est pas établi dans un but dévalorisant). [Sergey Brin & Larry Page, 1998] élaboreront la même année le PageRank

et le modèle du surfeur aléatoire qui mèneront aux mêmes conclusions que Kleinberg sur la forme des communautés et leur organisation hiérarchique.
En observant donc la structure des liens hypertextes il devient alors possible d'esquisser une carte topologique des différentes communautés prenant place sur le web en dégageant une hiérarchie intra-communautaire et en conservant uniquement les acteurs dit majeurs.
A partir de ce point, Linkfluence a développé un outil, Linkscape, dont le but est de veiller et d'explorer les publications d'une sélection de sites et de blogs. C'est une équipe de documentalistes avec l'aide d'outils informatiques qui a été chargée de choisir les sources qui sont actuellement suivies quotidiennement. L'expertise de l'équipe a permis de s'assurer d'une qualité minimale pour chacun des sites ajoutés tout en minimisant le bruit naturel et inhérent du web. De la propreté des données découle la facilité d'exploration et de traitement qu'il est possible d'en faire, d'où le soin apporté à leur sélection. Les sites ont été classés selon leur langue, leur appartenance communautaire autour de thématiques définies ainsi que leur place hiérarchique au sein de ces communautés. Aujourd'hui, Linkscape totalise environ dix mille sites identifiés comme acteurs majeurs de la sphère francophone du web et dont il est envisageable de penser qu'ils forment un échantillon représentatif des discussions qui y sont tenues. Linkscape s'étend aussi à d'autres langues comme l'anglais, l'italien ou l'allemand.
Ainsi, Linkfluence dispose d'un outil de cartographie du web fondé notamment sur la théorie des graphes avec lequel il peut établir des relations entre les sites selon divers critères.
Pour récapituler, au cours du stage les corpus utilisés ont été directement extrait depuis le Linkscape à travers son moteur de recherche sur un panel d'une dizaine de milliers de sites représentatifs du web français, ordonnés par catégories, dont les publications journalières sont indexées systématiquement.



Ci-dessus, la page d'accueil du Linkscape à partir de laquelle sont effectuées les requêtes. Nous pouvons y consulter différend indices de mesures ainsi que les billets relatifs à notre requête. Il s'agit du moteur de recherche, et permet d'accéder à l'ensemble des données engrangées depuis la création de ce premier. Nous avons un aperçu des billets qui répondent à la requête émise, et sont directement consultables par simple clique.



Ci-dessus, une vue de l'ensemble des sites sélectionnés sous forme cartographique pour représenter le web français. Le premier niveau de sphère représente les continents, le second les territoires et le troisième les communautés. Ces niveaux sont la manifestation des liens entre les différents sites à travers une thématique commune et les hyperliens qu'ils possèdent entre eux. La carte permet d'explorer intuitivement le corpus de Linkfluence et d'observer concrètement les liens entre chaque site.

2 Le cœur du projet

Directions et directives

Le stage a été l'occasion d'exploiter les différentes pistes à notre disposition pour extraire l'information du corpus. Ceci de manière automatisée sans intervention manuelle dans le processus afin de la présenter sous forme d'un nuage de mots. Les contraintes fixées ont été de conserver une facilité de portage dans différentes langues et d'utiliser des procédés ne nécessitant pas une maintenance (mise à jour) par un documentaliste expert. Le temps ayant permis de développer l'outil jusqu'à sa phase de mise en production, il a fallu réécrire le code de manière à ce qu'il soit facilement intégrable à l'infrastructure existante. Enfin, il faudra l'avoir rendu robuste et capable de gérer les problèmes informatiques usuels (erreurs d'écriture, ouverture de fichiers inexistants... etc). Dernier point, un maximum de paramètres devront avoir été pensé comme réglables pour éviter toute réécriture ou intervention au sein du code.
Si le nuage de mots n'est par définition qu'une forme de présentation graphique de l'information, les algorithmes, qui permettent l'extraction de l'information en amont pour donner un poids et un sens autre qu'un effet stylistique au nuage, sont quant à eux la réalisation concrète d'une des multiples facettes du TAL. Ce traitement de la langue (dans notre cas sous forme textuelle) est envisageable de bien des façons, et c'est là tout l'enjeu du stage qui a été mené au sein de Linkfluence. Si nous pouvons tout à fait distinguer cette partie algorithmique du nuage (rien n'empêchant de représenter les résultats par un autre intermédiaire), elle a été ici pensé pour produire une information directement exploitable sous la forme d'un nuage de mot dynamique avec une donnée temporelle.

Tour d'horizon des nuages de mots

Sur le web, nous pouvons aujourd'hui trouver une grande variété de nuages de mots (nous utiliserons le terme nuage de mots pour désigner à l'avenir la représentation graphique de l'information et la partie algorithmique comme un tout). Dans la plupart des cas ils ne diffèrent que par la présentation et ne proposent pas d'extraction avancée de l'information. Ils se contentent de prendre en entrée une liste de mots associés à une valeur numérique qui servira de score pour attribuer des poids différents, ce qui se traduira par des couleurs et/ou des tailles de polices différentes. Ils ont aussi un rôle de navigation au sein du site qui les emploie. En effet, les mots affichés possèdent généralement un lien hypertexte qui redirige l'utilisateur vers du contenu associé au mot sur lequel ils ont cliqué. A noter que le terme « mot » désigne par un abus de langage une unité graphique séparée par deux espaces dans notre cas. Nous serons amenés à l'opposer au terme n-gramme.
Cette valeur numérique a souvent comme origine la fréquence d'apparition des mots au sein d'un texte. Si cette information n'est pas négligeable dans le cadre de notre travail, seule elle reste pauvre et limite grandement les observations que nous pourrions envisager.
L'une des plate-formes les plus fréquentées pour ce type de média est Wordle. Pourtant en

page d'accueil il est fait explicitement mention du fait que le générateur se contente d'utiliser les fréquences d'apparitions : « *Wordle is a toy for generating "word clouds" from text that you provide. The clouds give greater prominence to words that appear more frequently in the source text.* ». Dans la description même du produit, il est clairement établi qu'il est considéré comme un « jouet » (toute proportion gardée, les algorithmes de gestion de l'espace et d'affichage de l'information étant tout à fait remarquables). Son but reste purement visuel et n'offre pas une étude plus fine et poussée de l'information issue d'un texte source.

Ci-dessus, un exemple de nuage réalisé par le générateur Wordle. Le résultat graphique est extrêmement flatteur pour l'œil, cependant, il est issu d'une liste de mots pondérés par leur fréquence d'apparition. Le nuage n'apporte donc, en soi, pas plus d'information que la simple fréquence de chacun de ces termes.

Nous pouvons tout de même trouver d'autres utilisations plus avancée de ce mode d'affichage de l'information. L'entreprise Wikio, par l'intermédiaire de Jean Véronis, dispose de nuages de tags intéressants aux différentes fonctionnalités. S'ils n'ont pas tout à fait la même forme (les tags étant finalement un set de mots sélectionnés), ils tentent de répondre à la même problématique, afficher de l'information pertinente. Nous pourrons donc noter la présence d'un nuage qui affiche uniquement des noms de personnes en essayant de résoudre la difficile tâche qu'est la détection d'entités nommées. Nous ne pourrons davantage entrer en détails dans la chaîne d'algorithmes derrière puisqu'elle n'est pas mise à disposition du public.

Ci-dessus, un exemple de nuages développés pour Wikio par Jean Véronis. Nous y remarquons la présence d'un nuage dédié aux entités nommées qui font l'actualité. Leur but premier est de faciliter la navigation au sein du site en navigant par thème et catégorie.

Le panel de nuages de mots est donc relativement large dans ses fonctionnalités sous-jacentes, les deux exemples précédents n'étant qu'une partie de ce qu'il est possible de faire. Néanmoins, nous resterons quand même réservés, ces possibilités restent limités due à la nature même du nuage de mots, peu importe la qualité de traitement offerte en amont, l'information aura toujours la même représentation à la fin.

3 Le point de départ

Le tf-idf

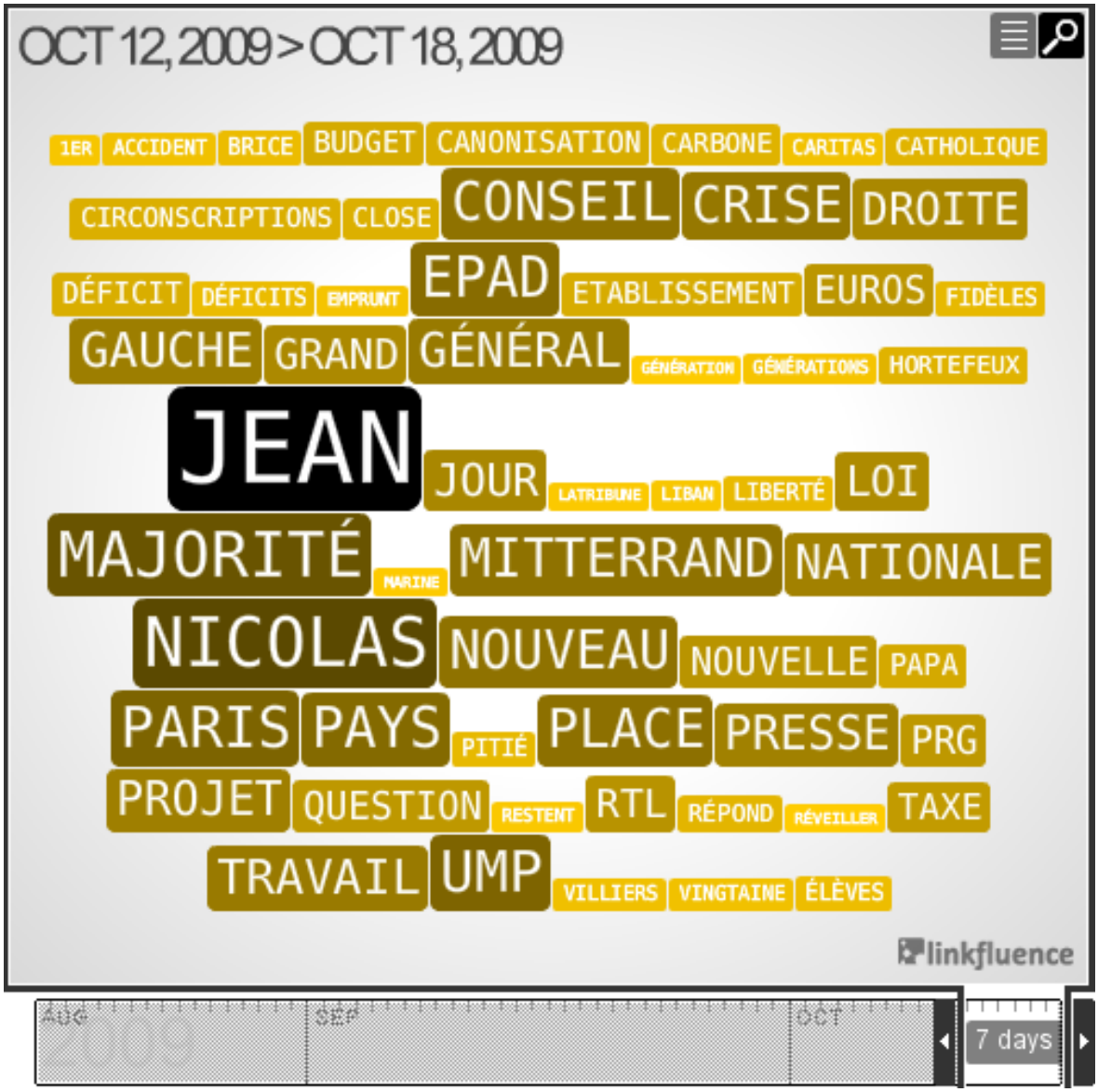
Définition :
- *term frequency* : la fréquence d'apparition d'un terme divisée par la somme des fréquences d'apparitions de l'ensemble des termes du document.
- *inverse document frequency* : le logarithme du résultat du nombre de documents du corpus divisé par le nombre de documents où au moins une occurrence du terme apparait.
- tf-idf : *term frequency * inverse document frequency*

Le *tf-idf* est un algorithme qui a la particularité de donner un indice à chaque mot d'un texte plus ou moins fort selon sa fréquence d'apparition dans celui-ci par rapport à sa fréquence d'apparition dans l'ensemble des documents auxquel le texte est comparé. Ainsi, un terme extrêmement fréquent comme « de » verra son indice relativement faible puisque sa grosse fréquence d'apparition sera pondérée par celle dérivant de son apparition quasi systématique dans chaque texte du corpus. Cela permet d'attribuer aux mots les plus spécifiques du texte un indice plus important qu'à ceux communs à l'ensemble du corpus tout en conservant une forme de classement par leur fréquence d'apparition. Dans son utilisation première, le *tf-idf* permet donc de sélectionner les documents parmi un set qui paraissent comme les plus pertinents par rapport à une requête précise. Dans l'utilisation que nous en avons faite, l'algorithme nous permet de sélectionner uniquement les mots (ou groupes de mots) qui caractérisent le mieux un ensemble de documents générés par une requête donnée par rapport à un corpus de référence. En effet, nous considérons le corpus de la requête comme un seul et unique document et le corpus de référence (composé d'un ensemble de billets sélectionnés aléatoirement – aujourd'hui résultants d'une requête contenant des opérateurs logiques du type ' le OR de OR la ' qui a pour but d'être la moins restrictive possible) comme le set de documents.
Au cours du stage, nous avons ajouté à la formule du *tf-idf* une variable pour écraser l'échelle de grandeur des fréquences d'apparitions qui pouvait être disproportionnée (rapport de 1000 entre deux termes ce qui a pour effet d'attribuer à des mots non spécifiques mais sur-représentés un indice fort), cela se traduit par la présence d'une racine carré sur le *term frequency*.
Concrètement, le *tf-idf* nous permet de pondérer les mots d'un texte par rapport à ceux d'un corpus pris en référence.

Un premier essai réalisé par Linkfluence

L'équipe de Linkfluence a développé un premier prototype de nuage de mots. Sa principale caractéristique qui diffère grandement de ce qui a déjà été fait est d'utiliser la comparaison (à travers l'utilisation du *tf-idf*) des mots d'un texte par rapport à un ensemble d'autres pour y trouver ceux qui font la spécificité de ce premier.

A cela s'ajoute une ligne du temps qu'il est possible de parcourir et nous avons un nuage dynamique capable d'afficher sur une période définie, pour un jour en particulier, les mots qui sont spécifiques à cette journée par rapport aux autres jours. Nous pouvons donc suivre l'évolution de ces mots, les voir apparaître, disparaître ou simplement changer de poids selon l'importance qu'ils prennent ou perdent.
Le nuage ne fonctionne en revanche qu'avec des mots, il n'y a pas encore la prise en compte des n-grammes qui sera développée au cours du stage. De plus, c'est uniquement à l'aide d'une *stoplist* que sont filtrés les éventuels mots indésirables. Il n'y a pas non plus la moindre utilisation de linguistique plus classique (fondée sur des grammaires par exemple), l'ensemble est uniquement le résultat d'un algorithme statistique. Cependant, sur ce point, cela rend le déploiement du nuage rapide et facile pour d'autres langues.



Ci-dessus, un exemple de nuage réalisé par le premier prototype. A noter que le résultat à été retravaillé à la main par le pôle études. Le premier prototype n'offrait pas la possibilité d'afficher des n-grammes.

Les fondements du nouvel outil

L'hypothèse derrière le nuage de mots est qu'on puisse arriver à extraire de l'information pertinente grâce au calcul du vocabulaire spécifique d'un texte par rapport à un autre. Cela sous-entend une qualité des données comme point d'entrée. Dans l'ensemble, le domaine du TAL est souvent confronté au bruit et il est généralement décidé de privilégier le silence au premier. Bien entendu, le but étant de réduire le silence au stricte minimum sans pour autant laisser le bruit s'immiscer dans les résultats. Dans notre cas, le choix des données ayant été effectué méticuleusement lors de la création de Linkscape, notre travail n'en a été que plus facile. Ce n'est pas pour autant que nous avons été libéré de toute forme de nettoyage, mais nous avons pu nous concentrer en privilégiant la précision et la justesse plutôt que l'action de masse.
Nous sommes donc parti sur des bases expérimentales, sans préjugés lors de la création des algorithmes. Au fur et à mesure de nos avancées nous les avons affinés afin d'approcher le plus près possible ce que nous jugions comme un résultat exploitable en production. Bien qu'étant dans un cadre de recherche et d'essais, il ne fallait pas occulter le fait que le travail réalisé avait un but concret d'application derrière, et qu'au delà de l'intérêt scientifique, il était nécessaire d'arriver à trouver quelque chose d'utilisable.
C'est pourquoi notre champ d'action est resté large mais sans oublier nos objectifs. Ainsi, s'assurer de la validité de l'hypothèse est resté un point central tout au long du stage, l'ensemble du projet reposant sur ce vocabulaire spécifique et ce qu'il est possible d'en faire.
Notre travail repose aussi sur des bases linguistiques éprouvées. Les collocations sont un phénomène qui a été largement étudié et analysé et dont l'existence n'est plus à démontrer. Ainsi, le projet croise l'utilisation du vocabulaire spécifique avec celui des collocations dans le but d'obtenir des n-grammes dont l'intérêt sémantique et informatif se révèle pertinent (dans notre cadre). Ces n-grammes ne se limitent pas aux expressions figées, ils recouvrent aussi des extraits (parties) de phrases qui peuvent être issues de la répétition d'une citation à travers les différentes publications. Nous souhaitions donc pouvoir observer des phénomènes de langue lors de l'affichage de résultats.
C'est donc en partant de ce principe : afficher le spécifique d'un document ou corpus de documents que le développement du nouveau nuage de mots s'est déroulé. Le code du prototype n'a pas été réutilisé, seule l'idée de fonctionnement a été conservé. Il a donc fallu construire, brique après brique les différentes strates de l'analyse et les suites d'algorithmes pour permettre la réalisation du projet.
Les compétences informatiques ont été relativement mises en avant, le projet nécessitait une bonne connaissance du langage Perl, connaissance qui a été en parti bâti tout au long du développement. Tout ceci combiné avec l'utilisation de linguistique par l'intermédiaire de la compréhension des mécanismes de la langue ainsi que l'utilisation de grammaires.
Le stage a aussi été l'occasion d'être formé à de nouveaux outils, aussi bien orientés développeur que linguiste. Ainsi, le *framework* GATE nous a apporté des solutions, au moins provisoires, dans l'attente d'un développement plus avancé et personnalisé de nos applications.

Partie 2 : De l'idée au produit final

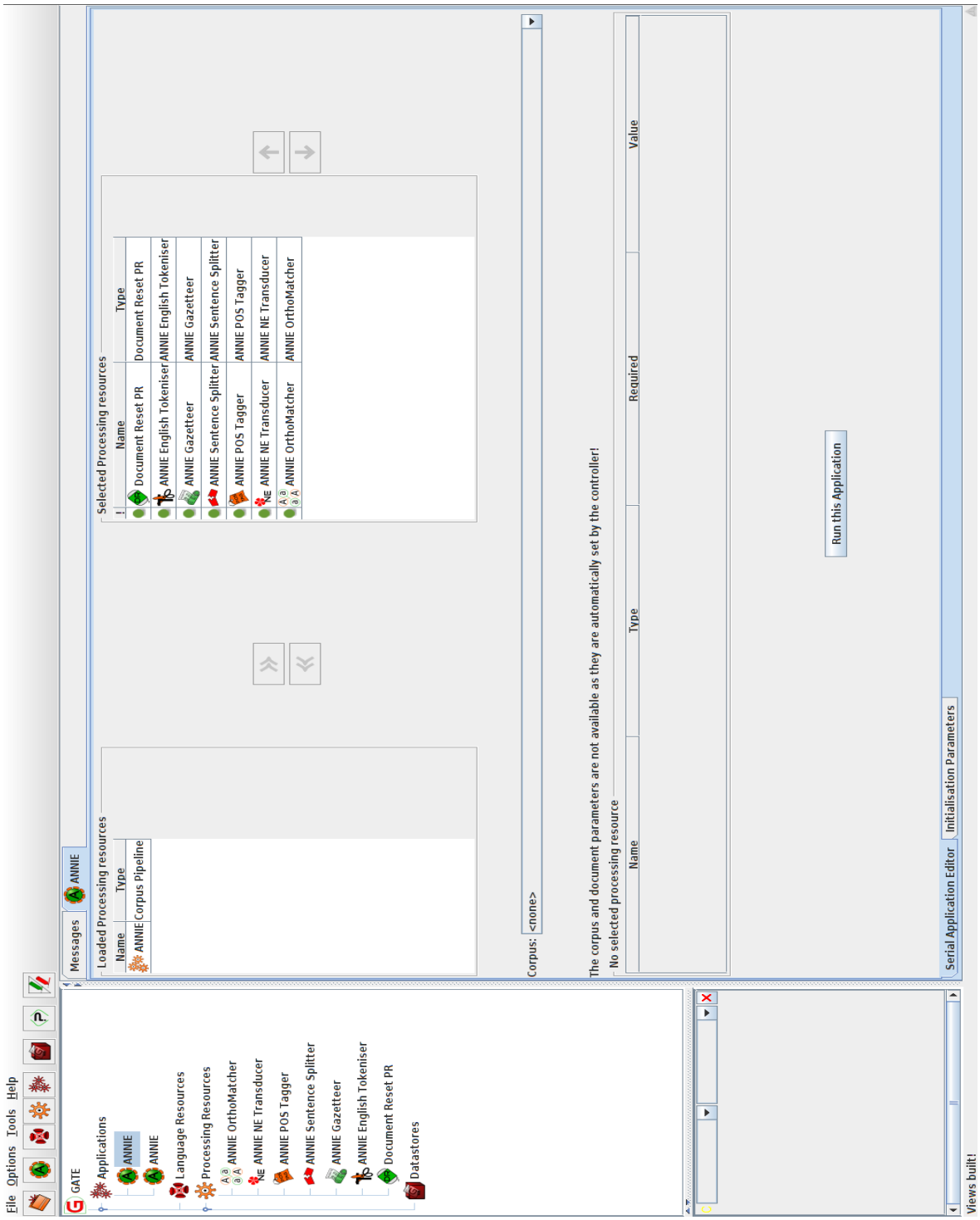
1 Les outils sollicités

TreeTagger

TreeTagger est ce que nous appelons communément un *pos-tagger*. Il a pour fonction d'associer à chaque mot d'un texte la partie du discours (*part of speech*) qui lui revient. Son utilisation est facilitée au sein de GATE par une intégration simple à mettre en place. TreeTagger est fondé sur les principes de *machine learning*.

GATE

GATE est un *framework* de TAL qui offre de multiples fonctionnalités pour traiter du texte et en extraire de l'information à travers son système d'annotation. En effet, GATE donne l'opportunité d'annoter un texte selon des critères et des règles que nous aurons nous-même établis. Ainsi, il offre la possibilité d'écrire des grammaires dont il se servira pour repérer une partie quelconque du texte source qui correspondra aux règles fixées. Il propose aussi d'effectuer des chaînes d'opérations sans que nous n'ayons à intervenir dans le processus. Il sera possible, par exemple, de tokeniser un texte, puis de le taguer et enfin de l'annoter. Les différentes strates d'opération sont sélectionnées par l'utilisateur mais c'est GATE qui se chargera de les exécuter consécutivement. GATE utilise notamment ce qu'il appelle des *gazetteers* : ce sont des listes de mots auxquels il attachera une annotation particulière que nous aurons déclarée dans un fichier contenant des métadonnées sur un ou plusieurs *gazetteers*, voire une annotation qui se trouve au sein de l'un d'entre eux.
Une autre de ses forces et l'un des points clés qui ont permis son utilisation : sa version dite *embedded* et qui offre une interface écrite en Java pour pouvoir utiliser GATE à travers nos propre applications sans devoir passer par la version usant de l'interface graphique.
GATE reste néanmoins un grand consommateur de ressources et son coût dans le traitement peut être important suivant la tâche qui lui est dédiée.
Notons enfin que GATE dispose d'autres fonctionnalités que nous n'avons pas exploitées pour notre projet puisqu'elles n'étaient simplement pas requises.



Ci-dessus, l'interface utilisateur de GATE. Elle permet d'utiliser le *framework* sans nécessiter de connaissances dans le domaine de la programmation. Elle aura été utile pour tester la validité de nos hypothèses avant de développer une version *embedded* pour maximiser l'investissement du temps passé dessus (le coup d'un développement étant conséquent, il n'était pas possible de se permettre de perdre notre temps dans une direction sans avenir). Sur la gauche se trouvent les différents outils auxquels peut faire appel l'utilisateur. Au centre, nous voyons la création d'un mécanisme d'automatisation d'une chaîne de traitement.

2 Calculs et n-grammes

Les n-grammes

L'un des points clés du nuage a été la volonté de proposer un outil capable d'afficher, à la différence de nombreux autres, des mots simples mais aussi des mots composés (ou des suites de mots, syntagmes figés, entités nommées...). Utilisant le principe des n-grammes (un 'gramme" étant ici une unité graphique délimitée par un espace devant et derrière elle), le premier stade de développement a été de mettre en place un système de calcul automatique des différents n-grammes d'un corpus donné. L'idée sous-jacente étant que les n-grammes spécifiques au corpus de la requête seraient composés notamment d'entités nommées complètes. Nous retrouverions alors à l'affichage des formes telle que «Noam Chomsky» en lieu et place de «Noam» d'un côté et «Chomsky» de l'autre.
Pour arriver à ce résultat, nous avons choisi de calculer le *tf-idf* des n-grammes en ayant limité à 4 le *n* maximum pour un n-gramme. En effet, après différents essais, au delà de cette limite, le nombre de n-grammes pertinents à afficher chute drastiquement, il devient alors bien plus difficile et contraignant d'extraire ceux qui répondent aux critères de validité pour être affichés.
Nous commençons donc par calculer le nombre de documents du corpus de référence contenant un n-gramme donné et ce, pour tous les n-grammes de ce même corpus. Cette première passe nous permet d'obtenir la partie *idf* du *tf-idf*, c'est à dire que nous avons les chiffres nécessaires à son calcul (le nombre de documents total et le nombre de documents contenant pour chaque n-grammes le-dit n-gramme).
Dans un second temps, nous calculons la fréquence d'apparition de chaque n-gramme dans le corpus de la requête (nous permettant d'établir la somme des fréquences de l'ensemble des n-grammes). Nous avons distingué à ce moment différents types de n-grammes, types correspondant au *n* du n-gramme. En d'autres termes, le *tf-idf* des n-grammes dont le *n* est égale à 1 est calculé distinctement du *tf-idf* des n-grammes dont le *n* est égale à 2 et ainsi de suite. Ainsi, la somme des fréquences des 1-grammes (nous utiliserons cette forme de notation par commodité) est différente de celle des 2-grammes, des 3-grammes et des 4-grammes. Avec cette passe, nous avons maintenant à disposition la partie *tf* du *tf-idf*, partie qui nécessite pour chaque n-gramme sa fréquence d'apparition dans le texte (ici le corpus de la requête) et la somme des fréquences de tous les n-grammes (du même type donc) de ce même texte.

Stoplist et accentuation

En parallèle de cette seconde opération, dans l'optique de traiter l'accentuation, nous conservons pour chaque n-gramme une forme que nous qualifions de méta et qui correspond au n-gramme désaccentué à laquelle nous associons l'ensemble des formes accentuées lui correspondant, avec pour chacune d'entre elles leur fréquence d'apparition.

Ci-dessous, un extrait d'un fichier généré pour la conservation de l'accentuation. Le format de ce dernier est '.json', il permet l'organisation des données de manière à ce qu'elles soient facilement récupérables par la suite. Nous pouvons voir les différentes réalisations des n-grammes dans le corpus.


```
      "lemma" : "woerth denonce des",
      "forms" : [
         {
            "count" : 15,
            "word" : "woerth dénonce des"
         }
      ]
   },
   {
      "lemma" : "dise que mme bettencourt",
      "forms" : [
         {
            "count" : 18,
            "word" : "dise que mme bettencourt"
         }
      ]
   },
   {
      "lemma" : "revelations de mediapart",
      "forms" : [
         {
            "count" : 8,
            "word" : "révélations de médiapart"
         },
         {
            "count" : 15,
            "word" : "révélations de mediapart"
         }
      ]
   }
```

Nous transformons aussi toutes les majuscules en minuscules pour cette partie. Ces deux actions sont le résultat d'un parti-pris de notre part. En effet, nous avons convenu qu'à l'affichage nous sélectionnerions la forme accentuée d'un n-gramme dont la fréquence d'apparition est la plus haute, et nous avons décidé de fusionner deux n-grammes qui ne diffèrent que par leurs majuscules et minuscules. Pour l'établissement de la liste des n-grammes nous conservons les majuscules, en revanche tous les n-grammes ont été désaccentués et un double du corpus sans forme d'accentuation est créé par la même occasion. Quelques transformations sont effectuées sur ce double, notamment au niveau des tirets pour s'assurer de la bonne tokenisation de celui-ci. En effet, lors du traitement par GATE qui aura lieu par la suite, il est nécessaire que les deux tokeniseurs (celui de GATE et celui de TreeTagger) arrivent au même résultat, ce qui n'est pas possible sans ces petites modifications (sans quoi une erreur interrompt la chaîne).
Parmi les n-grammes, nous excluons toute forme de ponctuation sauf les apostrophes (pour le français). Celles-ci sont conservées dans le but d'obtenir des formes telles que « liberté d'expression ». Elles bénéficient donc d'un statut particulier mais qui n'influe pas dans le traitement général, elles sont considérées comme un gramme quelconque.
A la suite de ces deux phases, nous avons tout ce qui est nécessaire pour établir l'indice du *tf-idf* pour chaque n-gramme de notre corpus issu de la requête utilisateur par rapport à notre corpus de référence. Cependant nous n'avons pas encore notre liste définitive, pour le moment, même en ne sélectionnant que les n-grammes dont l'indice est le plus fort il reste beaucoup de bruit. Pour remédier en partie à cela, nous utilisons deux *stoplists*. La première est une *stoplist* classique. Elle contient des n-grammes que nous souhaitons bannir purement et simplement si nous les rencontrons. La deuxième est plus subtile. Elle contient des expressions rationnelles associées à des règles. Ces règles sont au nombre de trois : *First*, *Last* et *Exact*. Leur utilité est de cibler avec précision une partie du n-gramme. *First* exige la chose suivante : si le premier mot d'un n-gramme donné correspond à l'expression rationnelle associée, le n-gramme est banni. *Last* propose la même chose mais avec le dernier mot du n-gramme. Enfin, *Exact* demande à ce que l'ensemble du n-gramme corresponde à l'expression rationnelle (sa fonction est similaire à la première *stoplist* si ce n'est qu'une expression rationnelle facilite la construction de la stoplist en permettant avec une seule entrée d'en matcher plusieurs).

Ci-dessous, un extrait du fichier de *stoplist* fondé sur les règles énoncées précédemment.

```
million(s)?                                  F/E
milliard(s)?                                 F/E
millier(s)?                                  F/E
centaine(s)?                                 F/E
dizaine(s)?                                  F/E
douzaine(s)?                                 F/E
beaucoup                                     F/E
un(s|e)?                                     F/L/E
l[aeiouy]                                    F/L/E
les                                          F/L/E
je|tu|il(s)?|elle(s)?|on|nous|vous           F/L/E
qu[aeiouy]                                   F/L/E
quoi                                         F/L/E
```

Sélection

Dernier critère de sélection, apparaître dans un minimum de billets. Bien que le corpus issu de la requête soit considéré comme un seul et unique document, pour éviter des effets indésirables, nous nous assurons qu'un n-gramme ait au moins une apparition dans un minimum X de billets du corpus (X étant paramétrable pour répondre à la variation de taille des corpus). Par exemple, pour un corpus de 1000 billets, nous situons le minimum à 10. Il faut que le n-gramme soit présent dans 10 billets différents au minimum pour être éligible. Ce dispositif permet notamment de rendre le nuage plus robuste au bruit. Si un billet contenant du spam venait à se glisser dans le corpus, les n-grammes qu'il contiendrait auraient peu de chance de remonter malgré leur possible spécificité par rapport au corpus de référence. Ce paramètre est variable d'un type de n-gramme à l'autre, les 1-grammes ont par défaut un X supérieur à celui des 4-grammes.
Nos n-grammes épurés, nous pouvons sélectionner ceux dont le *tf-idf* est le plus fort. Nous avons choisis de ne conserver que les 150 premiers 1-grammes. Pour les autres, nous conservons les 1500 premiers pour chaque type restant. Nous obtenons une liste de 4650 n-grammes. Si nous avons décidé de limiter le nombre des 1-grammes, c'est dans le but d'atténuer le bruit qui a tendance à augmenter de manière plus importante pour cette catégorie précise malgré l'utilisation du *tf-idf* ainsi que limiter leur présence dans la partie de post-traitement.

3 Grammaires et automates à état fini

A la suite du calcul du *tf-idf* et de la création d'une liste de n-grammes, ces derniers ne sont pas tous éligibles à l'affichage. Bien qu'ils soient spécifiques au corpus issu de la requête par rapport au corpus de référence, cela n'implique pas qu'ils soient porteurs d'un sens comme nous le souhaiterions. Il faut donc encore effectuer un tri parmi eux. Pour cela, nous avons utilisé un système de masques. Derrière ce nom se cache une liste de formes syntaxiques tel que '*noun* de *noun*' fonctionnant sur le principe d'automates à état fini. Sur la base des travaux de [Michel Mathieu-Colas, 2009], nous avons choisi celles qui sont les plus fréquentes et les plus pertinentes du français tout en excluant celles composées de mots outils (ou mots vides – mots qui ne portent pas de sens mais n'ont qu'une fonction grammaticale). Dans ce but, nous avons choisi d'utiliser le *framework* GATE (incluant TreeTagger).

Ci-dessous, un extrait des grammaires qui ont été écrites pour sélectionner les n-grammes désirés. La syntaxe du langage utilisé est très proche de celle du Java. Nous pouvons constater la précision des règles pour nous assurer de ne cibler que des n-grammes très particuliers.

```
Rule: Trigram
Priority: 30
(
    { Target.rule == "3gram", Token.category == NOM }
    ({ Target.rule == "3gram", Token.category == ADJ }|
{ Target.rule == "3gram", Token.string ==~ "[Dd]e|[Dd]es|
[Dd]ans|[Ll]e|[Ll]a|[Ll]es|[Uu]n|[Uu]ne" })
    { Target.rule == "3gram", Token.category == NOM }
):trigram
-->
:trigram.three_grams = { rule = "three_grams" }



Rule: NNpN
Priority: 45
(
    { Target.rule == "4gram", Token.category == NAM }
    { Target.rule == "4gram", Token.category == NAM }
    { Target.rule == "4gram", Token.string ==~ "[dD][eu]|[lL]
[ea]|[vV][oa]n" }
    { Target.rule == "4gram", Token.category == NAM }
):nnpn
-->
:nnpn.four_grams_ne = { rule = "four_grams_ne" }
```

Nous procédons de la manière suivante pour effectuer le tri : la liste de n-grammes précédemment récupérée est convertie sous forme de *gazetteer*, un fichier particulier de GATE dont il se sert pour annoter un texte avec les entrées qu'il contient.

Ci-dessous, un extrait d'un *gazetteer* qui a été généré. Nous pouvons y remarquer le marqueur « `&type=Xgram` » en fin de chacune des lignes. Le marqueur permet de conserver la notion de n-gramme lors de la phase d'annotation.

```
politique&type=1gram
fortunes&type=1gram
eurodeputee&type=1gram
Robert&type=1gram
dit&type=1gram
Etat&type=1gram
amalgame&type=1gram
situation&type=1gram
Eric Woerth&type=2gram
Liliane Bettencourt&type=2gram
de Liliane&type=2gram
ministre du&type=2gram
Florence Woerth&type=2gram
de Maistre&type=2gram
affaire Bettencourt&type=2gram
Mme Bettencourt&type=2gram
la milliardaire&type=2gram
controle fiscal&type=2gram
du Budget&type=2gram
tresorier de&type=2gram
```

Le double du corpus désaccentué que nous avons généré est donné en entrée à GATE. Toutes les annotations ont lieu sur lui. Dans un premier temps, GATE (à l'aide de TreeTagger) tague le corpus. Dans un second temps, sur la base du *gazetteer* établi, tous les n-grammes compris dans le corpus sont annotés. A cet instant nous possédons un corpus dont les n-grammes existants dans le *gazetteer* ont été annotés et dont chaque mot les composants possède un tag qui correspond à la partie du discours identifiée par TreeTagger. Cependant, certains n-grammes annotés se chevauchent (lorsque l'un est compris dans l'autre). Pour opérer une sélection nous sommes parti du principe que les n-grammes les plus longs sont par nature plus rares (puisqu'ils sont dépendants de plus grandes restrictions au niveau des règles, en plus de leur longueur naturelle), c'est pourquoi nous les favorisons en commençant la sélection par eux. Dès lors qu'un n-gramme est sélectionné, tout n-gramme qui le chevauche est ignoré. Le chevauchement consiste à la superposition de deux (ou plus) n-grammes sur un même mot (ou plusieurs). L'intérêt de cette opération est de créer une unité au sein des différents types de n-grammes pour qu'une comparaison entre eux ait plus de sens par la suite. Nous maintenons aussi un rapport équilibré dans l'échelle des fréquences d'apparition même si les 1-grammes conservent toujours une proportion plus importante sur ce point par rapport aux autres, puisque par essence un mot unique a statistiquement plus de chances d'apparaître (en terme de fréquence) qu'une suite de mots ordonnés.

Après cette sélection, les n-grammes auront récupéré une nouvelle annotation dépendante de la règle à laquelle ils ont correspondu et qui nous permettra d'appliquer différents traitements selon cette même annotation. En d'autres termes, nous parlerons maintenant de types de n-grammes non plus sur la base de leur *n*, mais sur l'annotation qui leur a été attribué.
Voici les différentes annotations sous lesquelles peuvent être rangés les n-grammes : un_gramme, deux_gramme, trois_gramme, quatre_gramme, deux_gramme_en, trois_gramme_en, quatre_gramme_en, n_gramme_punct.
Ces noms d'annotations représentent les formes générales que nous souhaitons conserver. Quelle que soit la langue, quelle que soit la classe grammaticale (ou la partie du discours) des mots composant les n-grammes, nous souhaitons pouvoir les classer sous l'une de ces annotations qui servent de références, facilitant le traitement d'une langue à l'autre par la suite. Le choix implicite est de proposer un nombre défini de formes que nous souhaiterions obtenir par l'intermédiaire des grammaires et de faire correspondre le résultat de ces dernières à celui-ci.

Ci-dessous, un exemple de billet récupéré suite à son annotation par GATE. Il est sous format XML, sortie que propose GATE. Nous voyons les n-grammes qui ont matché les règles de grammaires être annotés selon leur type. Les apostrophes seront décodées dans la suite du traitement.

```
20100705     <one_grams>Eric</one_grams> WOERTH, nous rabache
t on, est un <two_grams>homme honnete</two_grams>, tres
honnete, impeccable de rigueur et d&apos;honnetete. Bon tres
bien. Admettons le. Reconnaissons le. Il a droit a la
<n_grams_punct>presomption d&apos;innocence</n_grams_punct>,
non ? Evidemment ! Il n&apos;empeche : etre Ministre du
Budget et Tresorier de l&apos;UMP, ca ne se fait pas ...
surtout quand on organise des soirees pour les grands
<one_grams>donateurs</one_grams> de l&apos;UMP qui
beneficient ...de deductions fiscales ! Etre Ministre du
Budget et avoir une <one_grams>femme</one_grams> […]
```

4 Intelligence linguistique et attributions des scores

Les entités nommées

Maintenant que nous possédons un corpus annoté, nous pouvons en extraire les annotations pour travailler directement dessus. Dans l'ensemble, tous nos n-grammes annotés sont des candidats idéaux pour être affichés. Néanmoins, nous allons tout de même ajouter quelques restrictions ainsi que traiter certains d'entre eux de manière particulière.
Nous allons calculer un set de candidats sans tenir compte d'une quelconque donnée temporelle en premier lieu, et de ce set nous calculerons le score des candidats jour par jour selon les dates des billets de notre corpus dans un second temps.
Pour commencer, l'un des principaux objectifs que nous voulions atteindre par l'utilisation de n-grammes était d'arriver à extraire les entités nommées d'un corpus. Si notre système ne permet pas d'identifier de façon précise si tel ou tel n-gramme est une entité nommée, il n'en reste pas moins qu'elles sont présentes dans leur forme entière (la majeure partie du temps) puisqu'elles sont, pour la plupart, des n-grammes spécifiques de notre corpus. Nous allons donc essayer de les favoriser par un filtrage simple fondé sur une liste d'autorité construite à partir des données de Wikipédia.
Wikipédia propose régulièrement en téléchargement un *dump* de ses bases de données. A partir de celui-ci, nous avons extrait de la façon la plus précise possible les données relatives aux entités nommées. Pour procéder à cette extraction, nous avons utilisé le contenu des articles, notamment celui des *infobox*. De ces dernières, nous avons établi une liste de titres. Titres qui sont attribués aux *infobox* qui (après observation manuelle) correspondaient à une entité nommée dont nous souhaitions récupérer les informations. Nous avons concentré notre choix sur les personnes (fictives ou non), les lieux, les noms d'entreprises et de logiciels (ou produits). En automatisant la tâche, nous avons construit une liste à partir des titres d'articles et du contenu des *infobox* en veillant à conserver les relations entre les données lorsqu'elles avaient attrait à la même entité. En complément de cette première liste, par le croisement des informations que Wikipédia met à disposition, nous avons pu ajouter à chaque entité nommée ses différentes formes de réalisations (formes qui correspondent aux redirections vers l'article source de l'entité nommée).

Ci-dessous, un extrait du fichier des entités nommées récoltées depuis le *dump* de Wikipédia où les différentes formes d'une entité nommée sont observables.

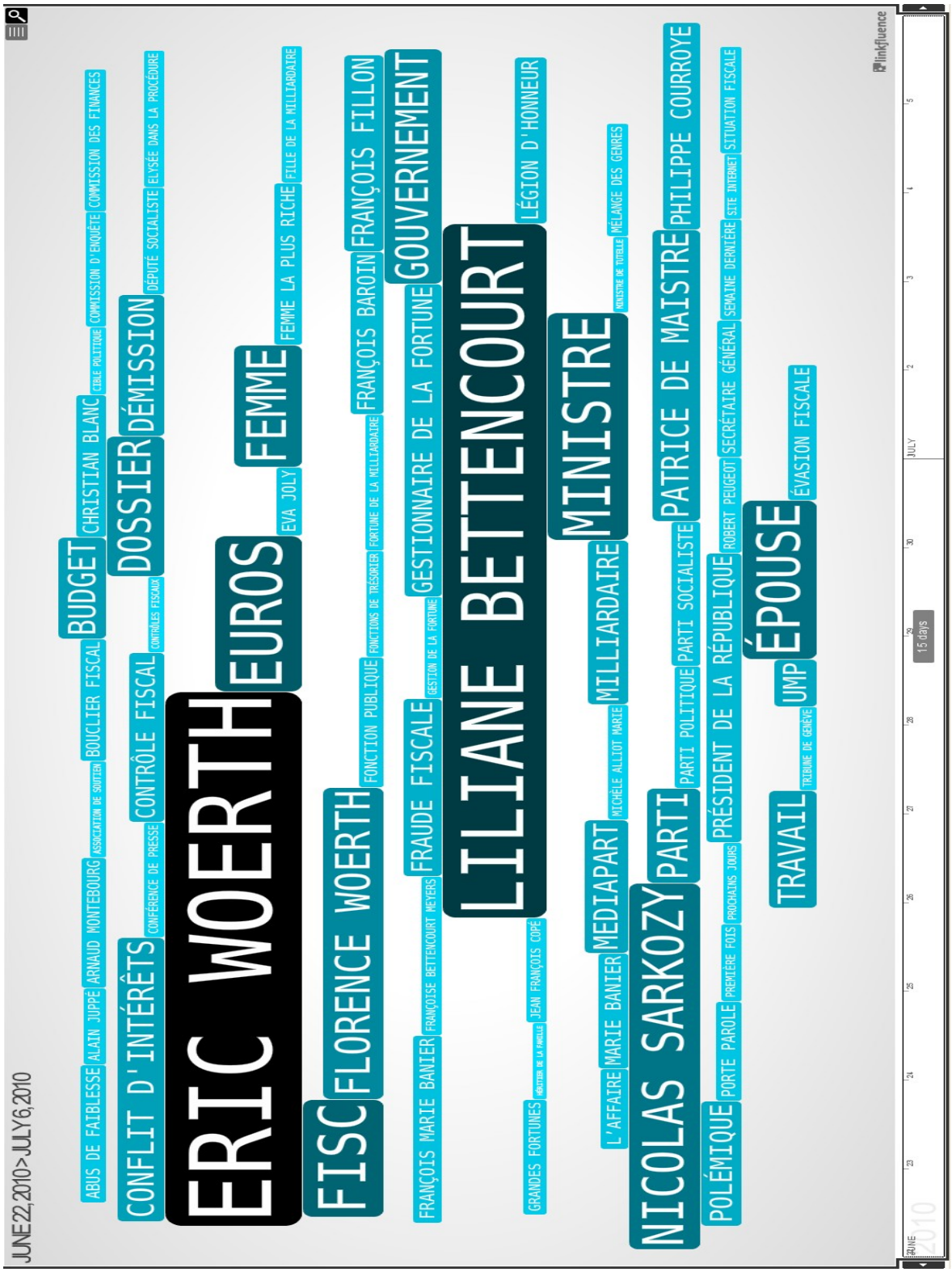
```
{
         "nom_de_naissance" : "Willis Marie Van Schaack",
         "redirections" : [
            {
               "forme" : "Lily Saint-Cyr"
            },
            {
               "forme" : "Lily St-Cyr"
            },
            {
               "forme" : "Lili Saint-Cyr"
            },
            {
               "forme" : "Lili St. Cyr"
            }
         ],
         "title" : "Lili St-Cyr"
      },
      {
         "surnom" : "Serey Die",
         "title" : "Die Serey Geoffrey"
      }
}
```

Plus haut, nous avons vu que les n-grammes possèdent maintenant un type défini par l'une des catégories que nous avons établies et à laquelle une ou des règles de grammaires sont attachées. Parmi nos n-grammes, ceux du type deux_gramme_en ont été identifiés comme une suite de deux grammes tagués *name*. Ce tague est attribué par TreeTagger à ce qui répond aux critères d'une entité nommée (critères établi par TreeTagger auxquels nous n'avons pas accès). L'identification reste donc légère, mais nous nous en servons comme point de départ. Dans l'ensemble, si toutes les entités nommées ne sont pas détectées, celles comprises sous l'annotation deux_gramme_en sont très satisfaisantes. Le problème se situant dans l'exhaustivité plus que dans la qualité. Nous soumettons ces n-grammes à notre liste, le but étant de trouver une correspondance. Si elle se produit, nous récupérons toutes les formes de l'entité nommée.
Cela fait, nous nous attachons à parcourir de nouveau notre corpus original pour ajouter à la fréquence d'apparition de notre entité nommée actuellement traitée (et dont la forme canonique qui sera affichée est celle de notre n-gramme annoté deux_gramme_en) toutes les occurrences de ses différentes réalisations. Et cela uniquement si elles ne sont pas présentes dans notre liste source de n-grammes. Enfin, nous fusionnons parmi notre liste de n-grammes globale ceux qui correspondent à l'une des formes de notre entité nommée. Le dernier traitement particulier pour les entités nommées sera de vérifier parmi les n-grammes du type un_gramme ceux présents à l'intérieur des n-grammes du type deux_gramme_en. Lorsque l'un d'entre eux est repéré, une comparaison des fréquences d'apparitions des deux formes est effectuée, et selon un seuil choisi (de valeur X, car paramétrable), c'est l'une ou l'autre (celle répondant au critère du seuil) de ces formes qui

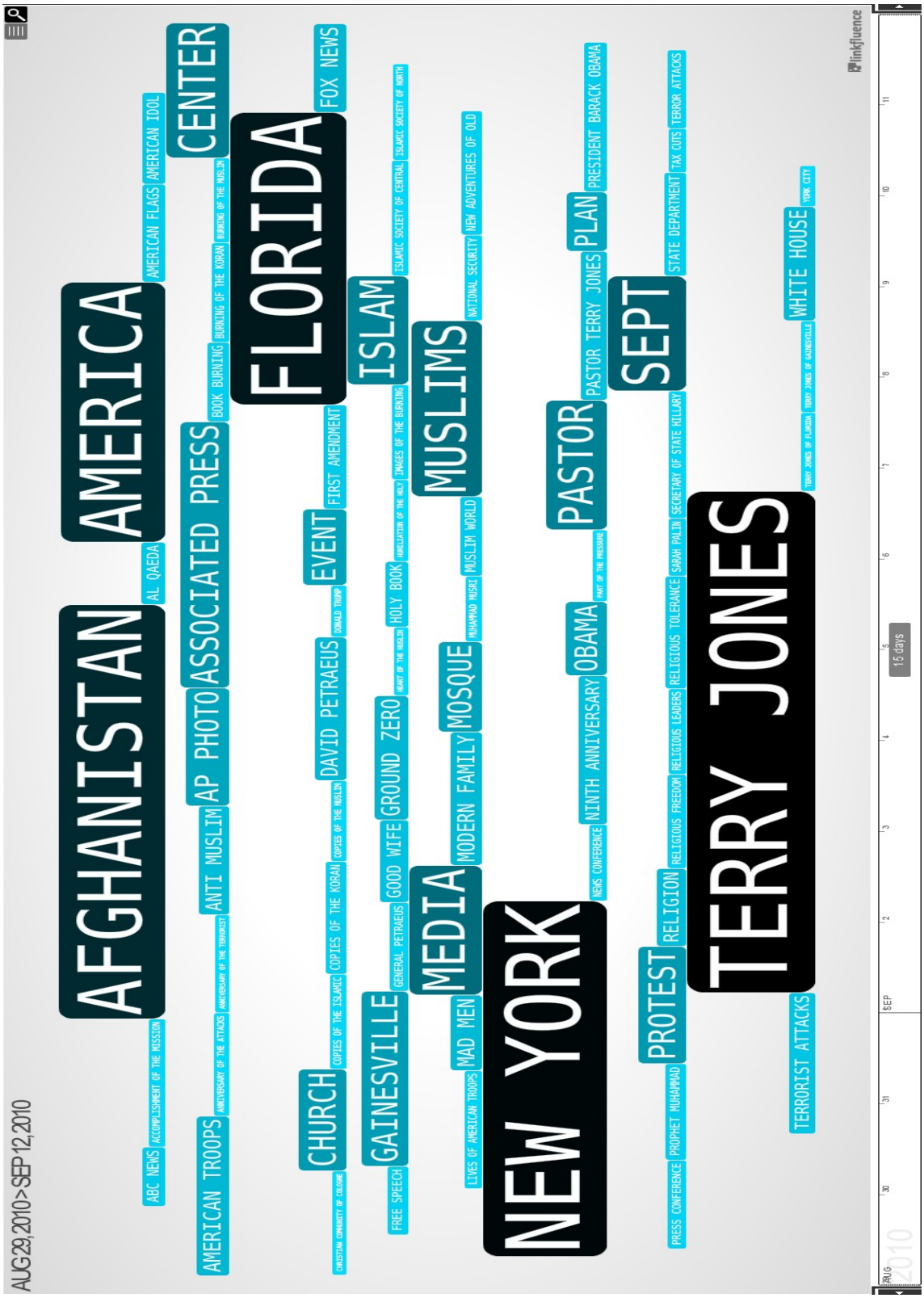
sera fusionnée au profit de sa correspondante.

Limiter la redondance à travers la fusion

Pour ce qui est de nos n-grammes en général, nous appliquons aussi une fusion basée sur la distance de Levenshtein. Elle est calculée pour chaque n-gramme d'un type par rapport aux n-grammes du même type. Si la distance est égale ou inférieur à un seuil choisi (de valeur Z car paramétrable), le n-gramme dont la fréquence d'apparition est la plus faible se verra fusionné au profit de celui dont elle est la plus forte. La même opération est effectuée pour chaque n-grammes de type deux_gramme au profit des n-grammes de type trois_gramme mais sans tenir compte de la fréquence d'apparition. Cela permet de favoriser les n-grammes longs pour augmenter leur visibilité. La fusion est automatique si la distance est inférieur à X (paramétrable). Ainsi, nous faisons de même pour les n-grammes de type quatre_gramme qui bénéficient d'une fusion avec ceux de type trois_gramme.
Nous avons aussi mis en place une deuxième opération de fusion par l'intermédiaire d'une comparaison entre un n-gramme de type trois_gramme et un autre de type quatre_gramme. Si le premier est inclus dans le second, en se basant sur leur fréquence d'apparition, nous conservons celui des deux qui possède la plus haute, avec en cas d'égalité un choix pour le quatre_gramme, toujours dans l'optique de favoriser le plus long.
Nous récupérons par la suite la forme accentuée la plus fréquente pour chaque n-gramme de notre liste. Enfin, nous passons les n-grammes restant à travers un filtre : un n-gramme doit avoir une fréquence d'apparition qui représente au moins X% de la somme des fréquences d'apparitions des n-grammes du même type (avec X paramétrable), et qui doit aussi représenter Y% de la moyenne des sommes des fréquences de l'ensemble des n-grammes (tout type confondu, avec Y paramétrable).
Nous possédons maintenant un set de candidats qu'il faut replacer dans un contexte temporel. Nous avons tout de même la possibilité à ce stade de créer un nuage sans cette notion de temporalité. La deuxième boucle de traitement consiste à calculer, pour chaque jour dont nous avons un billet dans notre corpus, parmi l'ensemble des n-grammes issus de l'extraction des annotations, le score de ces derniers s'ils apparaissent dans le set de candidats. Les opérations sont semblables à celles effectuées précédemment tout en tenant compte cette fois-ci de la présence dans le set des candidats du n-gramme en phase de traitement. Ainsi, nous avons par date les n-grammes que nous afficherons et nous pourrons suivre leur présence au cours de la période ou l'importance du nombre de leurs occurrences.



Ci-dessus, le résultat d'une génération de nuage de mots fondé sur la requête : « eric woerth » suite aux affaires qui ont frappé le ministre. Nous pouvons constater que le vocabulaire spécifique propose un véritable condensé d'information, et que sa précision est tout à fait intéressante.



Ci-dessus, le résultat d'un nuage de mots anglais fondé sur la requête : «terry jones» suite aux déclarations du pasteur. Les règles de grammaires anglaises ne possèdent pas encore le niveau d'affinage qu'ont les règles françaises. Le résultat, bien que très satisfaisant, est amené à s'améliorer avec le développement de règles mieux adaptées.

Partie 3 : Prise de recul, analyse et commentaires

1 Difficultés du projet

Perl

Au cours de la réalisation du projet, les difficultés et les problèmes auront fait leur apparition. La première source de ces contraintes a été le code en lui-même. Il a nécessité à plusieurs reprises, au fur et à mesure de sa complexification, une réécriture pour le e rendre plus lisible (réduire sa longueur) ou améliorer ses performances (optimiser le parcours et l'organisation des structures de données). Dans l'optique de faciliter son intégration, il aura aussi été totalement repensé orienté objet, le rendant modulable et paramétrable. La programmation orientée objet a nécessité une phase d'apprentissage sérieuse pour être mise en pratique. Néanmoins, ses avantages étant indéniables pour notre projet, le temps investi n'a pas été perdu et le résultat s'est avéré payant.

Un outil à dompter

Le *framework* GATE a posé aussi divers problèmes, dont certains que nous n'avons pas résolus. Dans l'ensemble, la mise en place d'un script faisant appel à l'interface *embedded* de GATE n'aura pas été facile. Cela commence par le langage même dans lequel est écrit cette interface, Java. Pour pouvoir écrire un script en Java il a d'abord fallu apprendre la syntaxe du langage et ses spécificités au moyen de tutoriels ou d'enseignements de l'équipe. Passé cet obstacle, c'est la documentation, parfois peut bavarde (se reposant essentiellement sur les javadoc), qui aura été un frein en imposant des recherches pour mettre en place un système opérationnel. A la suite de l'écriture du script, nous avons pu constater l'utilisation importante des ressources machine de la part de GATE. Chose amplifiée par une fuite mémoire que nous n'avons pas pu corriger à l'heure actuelle. La solution que nous avons mise en place pour le moment consiste à découper le traitement de GATE en plusieurs passes.
Nous avons donc été confronté en majorité à des problèmes techniques plutôt que des failles théoriques. Dans l'ensemble, nous avons pu passer outre en les résolvant de manière propre et efficace. Il était important de trouver un compromis entre solution et temps, puisqu'une solution qui nous aurait coûté un grand temps de développement n'aurait eu de solution que le nom. C'est cet équilibre qui nous a momentanément forcé à délaisser le problème de fuite mémoire. Néanmoins, il sera important d'y remédier dans un avenir proche, bien que cela ne nuise pas au fonctionnement du nuage de mots.

L'encodage

Problème classique qui surgit dès que l'on traite des données provenant du web, nous avons pourtant été épargné à ce niveau, puisque les corpus issus de Linkscape sont déjà tous préalablement encodés en UTF-8. Cependant, les outils GATE et TreeTagger ne proposent pas forcément la possibilité de traiter nos données dans un même encodage selon les langues (plus exactement, l'anglais ne bénéficie pas d'un module UTF-8 dans TreeTagger). Il aura donc fallu être capable de gérer au sein du projet la multiplicité des encodages en proposant des paramètres adaptés. Actuellement, nous avons donc la possibilité de préciser l'encodage d'entrée (inhérent aux fichiers) et l'encodage de sortie. A l'intérieur même du traitement, l'encodage n'a pas d'incidence, l'interpréteur Perl gérant ses propres représentations, il (l'encodage) n'intervient alors qu'au moment d'écrire dans un fichier ou lorsqu'il est lu.

2 Évaluation et critiques

Un constat, des avis

Il va sans dire que la qualité des résultats est allée crescendo avec l'avancée du stage et du projet. Dans l'ensemble, le bruit est relativement absent. Relativement parce que selon le point de vue il peut l'être plus ou moins. En effet, la perception de la validité d'un mot peut varier d'un observateur à l'autre. Au sein même de l'équipe les avis ont été assez différents pour en tenir compte. L'un des sujets principaux a été le traitement que devaient subir les 1-grammes. Devions-nous systématiquement les inclure dans leur homologues supérieurs ? La question a vu s'opposer les points de vue, il en ressort que nous avons décidé de les conserver quitte à voir apparaître une forme de redondance dans l'information affichée. Nous noterons tout de même qu'un processus de fusion a eu lieu en se fondant sur la distance de Levenshtein. Néanmoins, cette opération a davantage pour but de fusionner des n-grammes entretenant un rapport de singulier à pluriel ou féminin («bleu», «bleus», «bleue»).
Concernant la validité des résultats, au-delà de l'absence de bruit, ils se montrent tout à fait satisfaisants. Lorsque nous les parcourons, il faut garder en tête que les algorithmes favorisent les entités nommées mais ne les détectent pas à proprement parler. Combiné à l'attribution du score qui utilise des filtres selon la fréquence d'apparition d'un n-gramme, nous comprendrons que pour un texte donné, toutes les entités nommées qu'il possède n'apparaîtront pas dans le nuage de mots et que cela est normal (et non un échec).
Nous observons tout de même la présence de celles qui ont une fréquence suffisamment élevée pour être éligibles et donc pertinentes par rapport au reste du document. Nous avons délibérément choisi d'écarter un mot qui aurait une fréquence trop faible pour rendre le nuage visuellement plus homogène. Malgré tout, il reste toujours possible de paramétrer le processus de différentes manières selon le résultat désiré.
Les entités nommées mises à part, parmi les n-grammes restant, nous pouvons constater la présence de certains termes qui, s'ils ne sont pas considérés comme des mots vides, ne semblent pourtant pas posséder de réels liens associés au billet dont ils sont issus. Cela s'explique par l'extraction qui a eu lieu en amont, au niveau de Linkscape. Si les sites sont fiables et proposent du contenu de qualité, pour traiter un billet il faut arriver à l'isoler du reste de la page HTML dont il provient (menu, publicité, Javascript). Cette opération délicate est dans l'ensemble très bien réalisée. Il arrive cependant qu'elle laisse passer quelques parties de la page qui ne nous intéressent pas et viennent donc polluer en partie notre corpus. Cette intervention étant indépendante de toute action du nuage de mots, il n'est pas possible de la corriger dans nos algorithmes si ce n'est par le moyen de filtres. Il reste néanmoins difficile de supprimer des mots qui apparaissent aléatoirement et qui se révèlent être spécifiques au corpus malgré notre bon vouloir. De plus, il ne faut pas ignorer la présence de texte indésirable au sein même d'un billet, comme une publicité glissée entre deux paragraphes ou des phrases d'annonces placées ça et là, parfois pour résumer un billet publié sur le même site. Nous conviendrons donc que l'une des principales formes d'action pour atténuer le bruit reste l'extraction d'information le plus précisément possible.

Le spécifique

Parmi les résultats constatés, nous avons pu nous apercevoir de la limite associée au fonctionnement par extraction du spécifique d'un corpus. En effet, si pour des requêtes précises les résultats sont intéressants, lorsque l'on s'attarde sur des requêtes d'ordre plus générique, la pertinence même du spécifique est remise en cause. Pouvons nous parler d'un vocabulaire spécifique pour un corpus qui serait trop généraliste, et qui donc n'aurait en soit-même aucune spécificité ? Dès lors, une requête qui aurait comme aboutissement la génération d'un corpus de billets au trop grand nombre de sujets et qui par essence s'approcherait plus d'un corpus d'échantillonnage des discussions sur le web que d'un corpus ciblé sur un faible nombre de discussions ne mènerait qu'à un résultat peu satisfaisant d'un point de vue exploratoire. Nous aurions bien un nuage composé du spécifique du corpus, mais ce spécifique ne présenterait que peu d'intérêts. Nous avons donc une limite inhérente au principe même que nous utilisons, cela ne peut fonctionner que si le corpus possède intrinsèquement un vocabulaire spécifique par rapport à un autre.
Dans l'ensemble, cela reste dans le spectre d'utilisation que nous avions envisagé, la plupart des requêtes effectuées ciblent un sujet extrêmement précis. Cela soulève tout de même un autre problème. Les tailles de corpus sont très variables, d'une cinquantaine de billets à plusieurs milliers. Pour les corpus de petite taille (moins de 300 billets), le vocabulaire spécifique est étouffé par quelques termes qui vont monopoliser l'attention et occulter toutes formes qu'il aurait été intéressant de conserver par ailleurs. Leur intérêt sera donc grandement limité. Ceci est plus préoccupant puisqu'il n'est pas rare d'obtenir de petits corpus sur les requêtes régulièrement réalisées par le pôle études de Linkfluence.

Technique, statistiques et linguistique

D'un point de vue purement technique, l'ensemble de nos algorithmes restent relativement long dans leur traitement des données (il faut compter environ cinq à six minutes pour un corpus d'un millier de billets). C'est une chose qui devra faire l'objet d'une amélioration. Si dans une utilisation, pour le moment, restreinte et relativement expérimentale (même en production) cela n'est pas trop dommageable, ce temps de calcul reste un handicap. Des solutions ont déjà été envisagées, notamment le remplacement de GATE par un module créé de toute pièce ou un équivalent plus rapide (nous n'utilisons qu'une partie des possibilités offertes par GATE, il ne nous est donc pas nécessairement pertinent de le conserver pour les tâches qui lui sont demandées). Nous pourrions envisager d'inverser des parties de la chaîne de travail, en commençant par extraire tous les n-grammes qui correspondent aux règles de grammaires puis de calculer leur *tf-idf*. Cependant cette option risque d'apporter un peu de confusion si nous exportons le nuage de mots sur d'autres langues pour lesquelles nous n'avons pas le personnel capable d'adapter la partie linguistique du projet aux nouvelles langues. Il faudra alors de nouveaux se tourner vers la première chaîne de traitement imaginée.
Nous avons donc des solutions qui nécessitent plus de maturation avant d'être déployées.
Le procédé que nous avons mis en place offre à l'origine une grande part dans l'approche statistique des sciences du langage. En effet, en nous détachant d'une approche plus linguistique, nous pouvons plus facilement transposer dans différentes langues notre

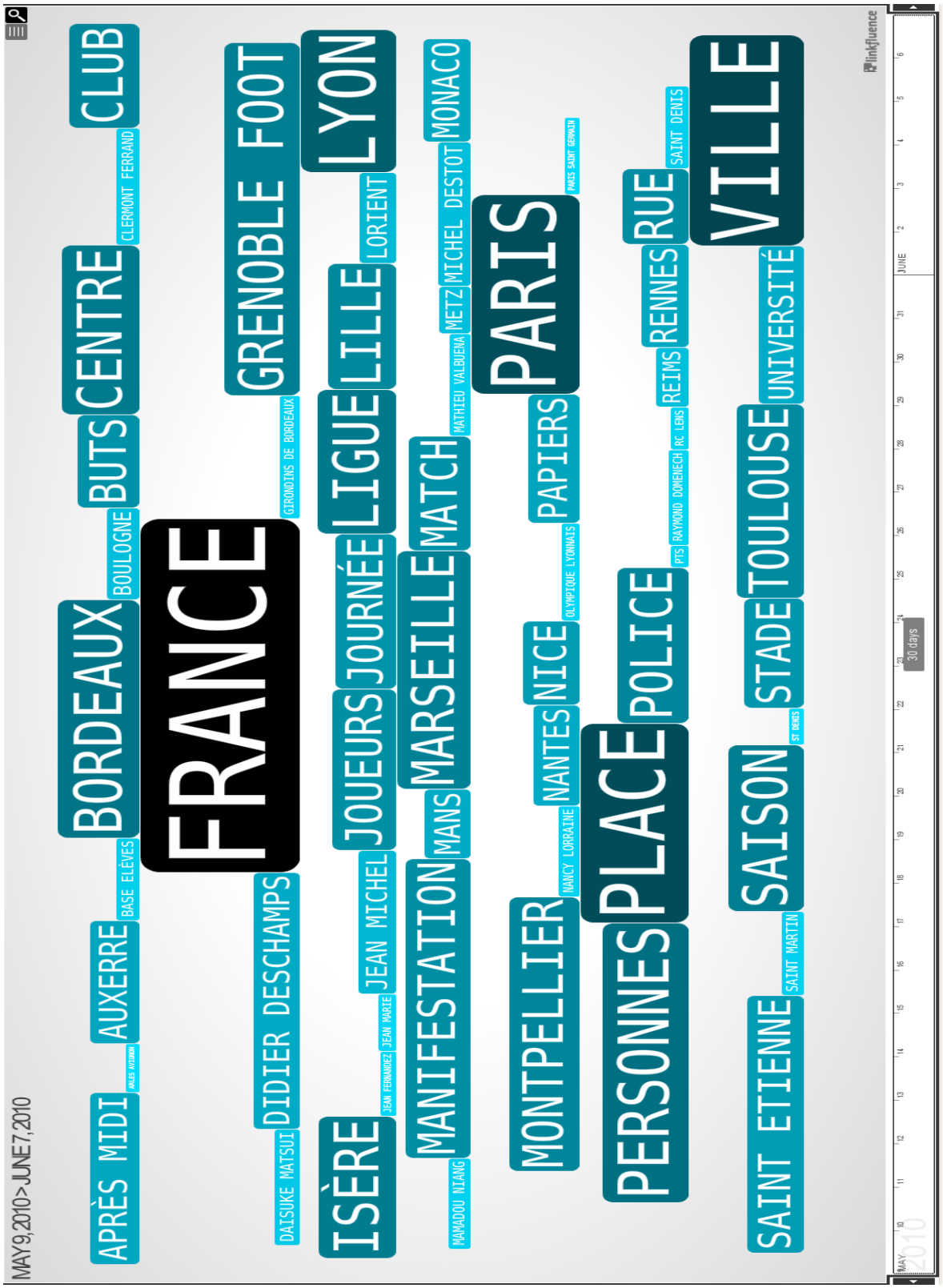
générateur de nuages de mots. Cependant, pour affiner nos résultats, nous avons développé une partie de nos algorithmes à l'aide de grammaires, déviant alors vers une approche plus hybride. Dans l'ensemble nous avons tout de même chercher à conserver au maximum l'indépendance de notre processus par rapport à la langue traitée en minimisant la taille des grammaires. Sur ce point là, nous sommes plutôt satisfaits des caractéristiques de portabilité de notre plateforme. Cependant, l'extraction d'une liste d'entités nommées depuis Wikipédia représente la plus grosse contrainte linguistique et nécessitera un expert dans la langue ciblée. La possibilité d'ignorer la phase allouée à la liste issue de Wikipédia reste très largement envisageable, ses conséquences se résument à la perte de l'augmentation potentielle en terme de fréquence des entités nommées. Dans l'ensemble cela reste relativement faible comme impact sur le fonctionnement de notre nuage de mots.

Licences

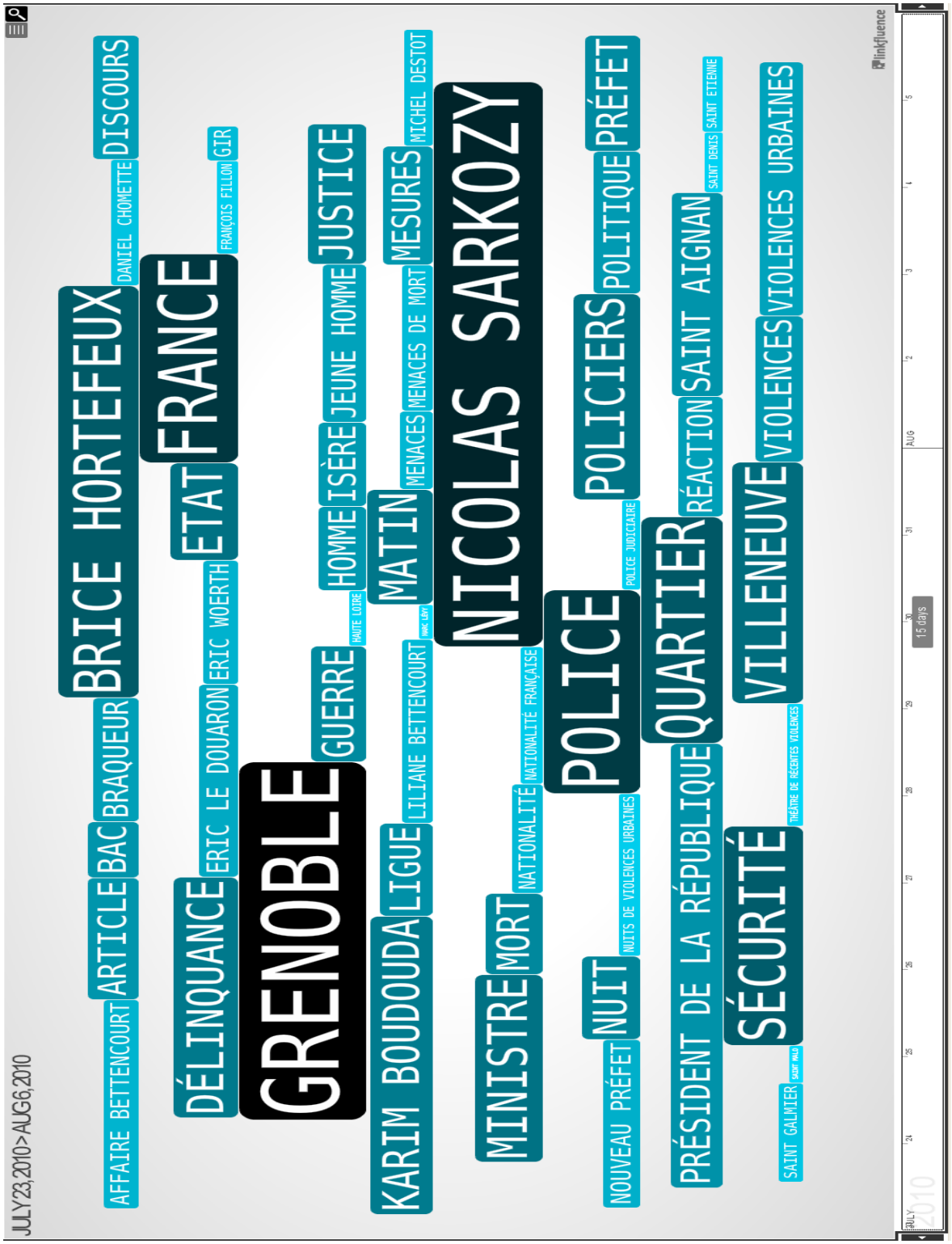
Les modules Perl utilisés (issus du CPAN) sont sous licence artistique tout comme le code Perl lui-même. Ils autorisent leur utilisation au sein de code propriétaire. Le langage Java est sous licence SCSL et nous octroie le droit de développer des outils sans contraintes (y compris commerciales). GATE est un logiciel libre, sous licence GNU/GPL 3. Concernant TreeTagger, sa licence est plus restrictive que les précédentes (le limitant à un usage privé ou de recherche/enseignement). L'usage de notre projet reste strictement confiné en interne. Seul le produit issu du résultat du projet peut être proposé à un client. Ainsi, de la même façon qu'un document créé par le logiciel Word (de Microsoft) peut être vendu (puisque ce document n'est pas la propriété de Microsoft), le résultat (ici le nuage de mots) pourrait être commercialisé. Actuellement, il n'est présent dans les études qu'en tant que valeur ajoutée (il ne figure pas sur les contrats en tant que service à fournir).

Une ville, un cas : Grenoble

Voici une illustration de ce que le vocabulaire spécifique est en mesure de produire lorsqu'il est extrait et mis en forme. Nous pouvons voir dans les deux nuages de mots ci-dessous l'apparition d'un phénomène médiatique qui fait écho aux évènements qui se sont déroulés à Grenoble le 16 juillet 2010. Le vocabulaire spécifique des billets qui répondent à la requête « grenoble » dans Linkscape change complètement de champ sémantique. Le premier nuage est issu d'un corpus précédant le 16 juillet, le second d'un corpus constitué de billets émis après. Nous noterons d'ailleurs que le premier nuage propose un résultat relativement général, la requête « grenoble » précédant les évènements du 16 juillet n'étant pas assez précise pour constituer un corpus représentatif de quelques sujets particuliers, le vocabulaire spécifique, bien qu'il semble s'orienter vers le sport et le football en particulier, n'offre pas une vue très intéressante à exploiter seule. En revanche, mis côte à côte avec le second nuage, sa valeur est toute autre et sert de point de comparaison pour constater l'évolution et le changement opéré dans le traitement de Grenoble par les médias (aussi bien officiels que les blogs d'anonymes).



Ci-dessus, le nuage issu de la requête « grenoble » avant les évènements. Nous pouvons constater un vocabulaire gravitant autour du domaine footballistique. La requête, avant événement, n'amenant pas sur une discussion particulière du web, le nuage n'est en soit, pas porteur d'une grande pertinence.



Ci-dessus, le nuage issu de la requête « grenoble » après les évènements. Le champ sémantique n'a que très peu de similitudes avec le précédent. Nous pouvons observer l'impact du traitement médiatique associé à Grenoble. La comparaison avec le nuage précédant permet de mettre en avant le bouleversement thématique qui peut être associé à un sujet selon des évènements circonstanciels.

3 Retours sur le déroulement du stage

Le projet

Les résultats obtenus répondent à la demande initiale qui a été formulée par Linkfluence. Le prototype créé va être intégré dans l'infrastructure existante et être utilisé dans la production d'études que mène l'entreprise. Cependant, comme tout prototype, il reste des points à améliorer, points identifiés et dont les solutions sont connues.
Le stage aura donc été l'occasion de pouvoir participer à toute la chaîne de mise en production d'un produit, depuis les premières ébauches en passant par les essais parfois concluants, parfois non, et enfin la réalisation finale qui suivra son propre chemin dans les mains des utilisateurs auxquels elle est destinée.
Ceci constitue une première entrée dans les possibilités d'intelligence linguistique qu'il est envisageable d'intégrer à Linkscape. Ce premier cap a été pleinement satisfaisant et augure une continuation dans la recherche et mise en place de dispositifs pour améliorer et développer le moteur de recherche.
Au niveau de la liberté accordée dans les recherches que nous avons pu mener, cela a été très appréciable. Cependant, la situation était en partie exceptionnelle puisqu'il s'agissait d'une première phase exploratoire des possibilités envisageables à travers le TAL. Il est plus que probable qu'à l'avenir, les options soient plus limitées ou plus contraintes par l'employeur. Contraintes qui n'ont pas été complètement absentes comme nous l'avons déjà évoqué.
Néanmoins, le tout a été très positif et s'est déroulé au-delà de nos espérances puisqu'en premier lieu, la mise en production d'un outil n'était pas clairement demandée (elle était sous-entendu, c'est à dire que les méthodes employées devaient être exploitables par la suite en production, mais la réalisation même de l'outil n'était pas un point requis). Cela aura eu pour effet de pousser les connaissances en programmation (principalement en Perl) plus loin qu'il ne l'était envisagé au départ.

Jour après jour

Le stage aura été l'occasion d'approfondir des connaissances développées au cours des études suivies à l'université. Parmi celles-ci, la programmation aura eu une part prépondérante. En effet, pour intégrer le travail effectué tout au long de ces six mois à une structure informatique professionnelle déjà existante au sein de Linkfluence. La nécessité de développer un code robuste et performant aura été obligatoire. Répondre aux contraintes professionnelles fut dans l'ensemble un point formateur et concret du monde du travail. Malgré la présence d'un tuteur, le besoin d'autonomie aura été important, le stagiaire doit être capable de travailler seul sur une période de temps variable puisqu'un employé ne peut pas abandonner son propre travail pour réaliser celui d'un autre (sauf cas exceptionnels). Néanmoins, nous ne sommes pas seul et isolé lors de nos journées et l'interaction avec l'équipe aura été source de détente et de bonne ambiance mais aussi de progression à

travers divers enseignements dispensés à la suite de questions ou simplement en suivant une discussion entre deux collaborateurs. Ainsi, naturellement, le travail d'équipe a été aussi un point clé de la réussite du stage. Au delà de l'intégration dans une équipe qui est déjà un pas dans la bonne direction, savoir écouter les différents avis, accepter les critiques et en tenir compte dans notre travail en sont d'autres qui ont permis de mener à bien ce projet.

Malgré l'attribution d'un projet, les besoins d'une entreprise étant fluctuant, il est nécessaire parfois de se détourner quelques temps de ses objectifs pour remplir une autre mission prioritaire. Nous avons été ainsi amené à travailler sur le système de masques d'extraction dans le moteur d'indexation. Travail qui aura consisté à chercher les optimisations possibles grâce au calcul des fréquences d'apparitions des mots, de la ponctuation et de leur pourcentage de représentativité au sein de balises HTML cibles.

L'arrivée dans un nouvel environnement demande aussi des capacités d'adaptations. Apprendre à gérer rapidement les outils mis à dispositions pour être opérationnel le plus rapidement possible et réduire la perte de temps qui est directement amputée sur la réalisation du projet. Capacité que nous aurons dû conserver tout au long du stage puisque selon les directions que prend le développement, il aura fallu régulièrement résoudre des problèmes en faisant appelle à de nouveaux logiciels pour lesquels l'apprentissage s'est fait par l'utilisation, de manière autodidacte.

A ce propos, nous pouvons apprécier l'utilisation du *framework* GATE qui est actuellement l'un des plus usité au sein de la communauté linguistique internationale et dont l'utilité sera sûrement requise pour des missions futures au sein de différentes entreprises.

Ces mois passés à Linkfluence auront aussi été le moyen de voir évoluer une entreprise, de constater les problèmes auxquels elle est confrontée ainsi que les solutions qu'elle trouve pour y remédier. Par la taille de sa structure, il aura été possible d'observer l'ensemble du fonctionnement interne (dans une certaine mesure), chose qui aura été très instructive.

Globalement, la bonne humeur et l'ambiance détendue au sein des équipes auront contribué au développement d'un environnement propice à la bonne volonté et au travail, véritable générateur de motivation et d'exigence de qualité. Loin d'être anodin, ces points ont été la source de l'envie de résoudre les différents problèmes et de proposer un travail répondant à des critères de qualité, parfois au delà de ce qui était demandé. L'expérience aura donc été pleinement satisfaisante et rassurante sur l'avenir qui s'offre à nous, et restera un point clé de notre parcours universitaire tout autant que professionnel.

Sources

Jean Véronis : www.blog.veronis.fr (17/09/2010)

Wikipédia : www.wikipedia.fr (17/09/2010)

Wordle : www.wordle.net (17/09/2010)

Wikio : www.wikio.fr (17/09/2010)

Bibliographie

Jon Kleinberg, « Authoritative sources in a hyperlinked environment », 1999

Michel Mathieu-Colas, « Essai de typologie des noms composés français », 2009

Sergey Brin & Larry page, « The anatomy of a large-scale hypertextual Web search engine », 1998

Glossaire

tf-idf : se référer partie 1, chapitre 3

GATE : se référer partie 2, chapitre 1

TreeTagger : se référer partie 2, chapitre 1

Distance de Levenshtein : elle est la mesure de similarité entre deux chaînes de caractères. Son principe est de proposer un résultat (représentatif de cette similarité) qui est le nombre minimum de transformations (ajout, suppression, remplacement) que doit subir une chaîne pour correspondre à une autre.

Résumé

Mots clés : nuage de mots, vocabulaire spécifique, extraction, web, n-grammes, grammaires, levenshtein.

Le nuage de mots n'est qu'une disposition graphique particulière de l'information. En général relativement faible d'intérêt, qu'en est-il si nous lui demandions d'afficher le vocabulaire spécifique d'un corpus ? Mais, pour commencer, qu'est-ce que ce vocabulaire ? Nous tenterons de répondre à ces questions, mais aussi d'expliciter les solutions que nous avons mises en place pour l'extraire et n'en conserver que l'essence, notamment grâce aux calcul des n-grammes, draguant les formes composées et les expressions figées ainsi que les entités nommées. A travers les différents algorithmes développés, nous exposerons les difficultés auxquelles nous avons été confronté ainsi que les critiques et la validité des résultats obtenus. Nous proposerons ainsi pas à pas notre démarche en justifiant nos choix et les parti-pris effectués, l'intérêt d'utiliser des n-grammes, l'utilité des grammaires et les systèmes de pondération.