Vers la correction automatique de textes bruités: Architecture générale et détermination de la langue d'un mot inconnu

Marion Baranes

Vers la correction automatique de textes bruités: Architecture générale et détermination de la langue d'un mot inconnu

Marion Baranes[1,2]
(1) Alpage, INRIA Paris–Rocquencourt & Université Paris Diderot, 175 rue du Chevaleret, 75013 Paris
(2) viavoo, 69 rue Danjou, 92100 Boulogne Billancourt
marion.baranes@viavoo.fr

RÉSUMÉ

Dans ce papier, nous introduisons le problème que pose la correction orthographique sur des corpus de qualité très dégradée tels que les messages publiés sur les forums, les sites d'avis ou les réseaux sociaux. Nous proposons une première architecture de correction qui a pour objectif d'éviter au maximum la sur-correction. Nous présentons, par ailleurs l'implémentation et les résultats d'un des modules de ce système qui a pour but de détecter si un mot inconnu, dans une phrase de langue connue, est un mot qui appartient à cette langue ou non.

ABSTRACT

Towards Automatic Spell-Checking of Noisy Texts : General Architecture and Language Identification for Unknown Words

This paper deals with the problem of spell checking on degraded-quality corpora such as blogs, review sites and social networks. We propose a first architecture of correction which aims at reducing overcorrection, and we describe its implementation. We also report and discuss the results obtained thanks to the module that detects whether an unknown word from a sentence in a known language belongs to this language or not.

MOTS-CLÉS : Correction automatique, détection de langue, données produite par l'utilisateur.

KEYWORDS: Spelling correction, language identification, User-Generated Content.

1 Introduction

Les outils de traduction, d'extraction de sentiments ou encore de fouille de textes sont de plus en plus utilisés. La majorité de ces outils s'appuient sur des corpus relativement propres. Si une personne choisit de travailler sur des données plus altérées, rédigées sur le web par exemple, sa tâche se complexifie. Il est donc important de pouvoir nettoyer ces textes afin d'appliquer par la suite les traitements voulus. Être capable de normaliser et de corriger automatiquement devient alors un réel besoin. Un besoin dans la mesure où ces derniers n'altèrent pas la qualité du texte. Un correcteur qui ferait de la sur-correction (qui corrigerait par exemple tous les mots inconnus) engendrerait la perte de nombreuses informations. Cette correction n'a pas besoin d'être parfaite. Dans le cadre de la fouille de texte par exemple, les informations détectées sont très souvent faites à l'aide de mots-clefs et de grammaires locales qui ne prennent pas forcément la flexion des mots en compte. Si une faute d'accord n'est pas corrigée, l'information sera tout de même détectée. Il vaut donc mieux un outil qui sous-corrige un texte plutôt qu'un qui le sur-corrige.

La correction orthographique n'est pas un sujet nouveau. Les travaux qui y font référence sont nombreux (voir section 2). Toutefois, ils sont rarement adaptés aux types de textes de qualité parfois très dégradée que l'on analyse lorsque l'on traite des données réelles dites « produites par l'utilisateur » (*User-Generated Content*) comme les blogs, les forums ou encore, les réseaux sociaux. Notre objectif est de combler ce manque, en développant une architecture et des technologies dédiées à la normalisation orthographique et typographique automatisée de corpus réels très bruités. Pour ce faire, nous nous appuierons sur un corpus provenant du web uniquement composé de messages clients (voir section 3) dont tous nos exemples sont tirés. Ce projet étant en cours, nous avons déjà une idée des résultats que nous souhaitons obtenir. L'architecture de correction est déjà en partie définie et est en cours d'implémentation. Elle est composée de nombreux modules qui, au fur et à mesure, amélioreront la qualité du texte et détermineront s'il convient ou non de corriger un mot inconnu (un mot inconnu pouvant correspondre à une faute d'orthographe, une entité nommée, un mot étranger, un emprunt ou encore un néologisme). Ces modules permettront, entre autres, de limiter les cas de sur-correction. Dans cet article, nous nous concentrerons sur l'un de ces modules dont le but est de déterminer si, dans un texte dont on connaît la langue, un mot inconnu correspond à un mot étranger ou non.
Cet article est structuré comme suit. Nous commencerons par dresser un état de l'art du domaine (section 2). Puis, nous décrirons les objectifs et principes généraux de correction que nous comptons implémenter (section 3). Nous préciserons dans cette même section les premières étapes préalables au module décrit en section 4 et pour lequel nous disposons de résultats évalués. Enfin, nous ferons un point sur le travail réalisé et sur nos perspectives (section 5).

2 État de l'art de la correction orthographique automatique

La mise en place d'un système de correction dépend beaucoup du type de corpus que l'on veut corriger. C'est pourquoi nous ferons état de la grande diversité de ces derniers dans le paragraphe qui suit avant de dresser un panorama des travaux réalisés dans le domaine.
Les correcteurs automatiques ne tendent pas à corriger les mêmes types de fautes, notamment en fonction de la provenance de ces corpus. Ainsi, un texte retranscrit par un processus de reconnaissance optique des caractères (OCR) ne contiendra pas les mêmes erreurs qu'un texte journalistique, qu'un texto (ou SMS), un mail ou encore un message posté sur un médias social du web (réseaux sociaux, forums, etc.). À tous ces canaux, correspondent des documents qui divergent en fonction de leurs tailles, de leurs contenus, de leurs objectifs, du type de vocabulaire utilisé (spécialisé ou non, familier ou soutenu), ou encore de l'aisance qu'a le locuteur avec la langue utilisée (écrit-il dans sa langue maternelle ou non ?). De ce fait, on peut supposer que les fautes produites dépendront aussi de ces critères et ne seront pas systématiquement de même nature. Par exemple, un texte OCR a plus de chance de contenir des fautes liées à la similarité typographique « graphique » (« *l* » vs « *1* ») tandis qu'un message de forum contiendra plus probablement des erreurs de proximité phonétique et/ou typographique (proximité des lettres sur un clavier). Le type de fautes à corriger variera ainsi en fonction des corpus sélectionnés.
On distingue généralement les deux types d'erreurs suivants dans les textes : les fautes lexicales et les fautes grammaticales (Kukich, 1992). Sont placés dans la catégorie des fautes lexicales tous les mots qui ne figurent pas dans le dictionnaire (voir exemple 1a), contrairement aux fautes grammaticales qui ne peuvent être détectées que si le contexte est pris en compte (voir exemple 1b). Ces deux types d'erreurs sont souvent traités séparément dans la littérature.

(1) a. j'**aimrai resrever** 2 billets

b. **ces** nul tu captes pas en montagne je le sais car je **lait**

À ses débuts, la correction lexicale se faisait indépendamment du contexte et s'appuyait notamment sur des règles de correction typographique (suppression, ajout d'un caractère, substitution d'une lettre avec une autre et inversion de deux lettres) à effectuer afin d'obtenir un mot correctement orthographié (Damerau, 1964; Kernighan *et al.*, 1990). Néanmoins on a rapidement réalisé que cette technique ne suffisait pas. Par exemple, si on observe la phrase : « *J'espère que vous* **réallisez** *que vos produits sont* **tjs hor** *de* **pri**... », on constate que seul « *réallisez* » a de réelles chances d'être bien corrigé. Le mot « *pri* » pourrait l'être aussi mais ses corrections possibles sont nombreuses, donnant lieu à une ambiguïté : doit-il être corrigé par « *prie* », « *pris* », « *prit* » ou « *prix* » ? Il en est de même pour « *hor* ». D'autres solutions furent donc proposées par la suite.

Dès les années 1990, Kukich (1992) publie un panorama des diverses techniques existantes de l'époque. Beaucoup de travaux, y compris de très récents, ont choisi d'utiliser des modèles de langage *n*-grammes afin de prendre en compte le contexte du mot à corriger. Ces *n*-grammes sont généralement composés de tokens (Brill et Moore, 2000; Carlson et Fette, 2007; Park et Levy, 2011) mais cette solution n'est pas parfaitement satisfaisante. Si le contexte du mot à corriger est mal orthographié, ce qui est le cas dans l'exemple proposé ci-dessus, une solution intermédiaire serait alors d'utiliser des *n*-grammes phonétiques (Toutanova et Moore, 2002) ou de prendre en compte ces deux types de *n*-grammes (Boyd, 2009). De cette manière, les fautes d'orthographe ne modifiant pas la phonétique d'un mot ne pourraient pas altérer les résultats du correcteur : le mot « *hor* » de notre exemple ne gênerait pas la correction de « *pri* » et inversement. Ces modèles peuvent être associés ou non à un modèle d'erreur et s'appuient généralement sur de nombreux paramètres supplémentaires tels que la position d'une erreur dans un mot, la catégorie du mot à corriger, sa longueur ou encore sa phonétique. Une autre approche consiste à prendre en compte la mesure de similarité distributionnelle qui existe entre une phrase contenant une erreur et ses candidats de correction possibles (Li *et al.*, 2006).

Par ailleurs, avec l'essor des nouvelles formes de communication, de nouvelles approches ont été proposées pour traiter le langage texto (ou SMS) : en passant par la phonétisation du texte à corriger (Kobus *et al.*, 2008), en ajoutant des ressources lexicales à un correcteur lexical dit classique (Guimier De Neef et Fessard, 2007) (ce qui permet de systématiser certaines corrections comme « *tjs/toujours* ») ou encore en s'appuyant sur des modèles entraînés sur des textes bruités alignés avec leur contrepartie nettoyée (Beaufort *et al.*, 2010).

Enfin, l'émergence d'internet a eu des répercussions sur plusieurs techniques de correction qui l'ont considéré comme une source d'informations pertinente. C'est, par exemple, le cas de Chen *et al.* (2007) qui proposent une méthode enrichie par des résultats de requêtes, produites sur des moteurs de recherche.

Corriger tous les mots qui ne figurent pas dans le dictionnaire nécessite par ailleurs de détecter les entités nommées, les néologismes ou encore les emprunts. Les quelques approches qui ont choisi de traiter ce problème reposent généralement soit sur des lexiques spécifiques (Beaufort *et al.*, 2010; Kobus *et al.*, 2008) soit sur le contexte (Li *et al.*, 2006). Par exemple, Han et Baldwin (2011) proposent d'utiliser un classifieur qui s'appuie sur les paramètres de dépendances syntaxiques reliant le mot susceptible d'être mal orthographié aux mots présents dans son contexte afin de déterminer s'il doit être corrigé ou non.

Comme nous l'avons dit précédemment, les fautes grammaticales donnent lieu à des mots existants dans le dictionnaire (par exemple : *conseil/conseille* ou *ai/est/et/hait/...*). Pour les

corriger, s'appuyer sur le contexte est donc inévitable. Si on prend la phrase « *Enfin un objet qu'***ont** *peut emporter partout avec* **sois** », on ne pourra corriger le mot « *ont* » en « *on* » qu'à condition de prendre en compte ses mots voisins. Pour cela plusieurs méthodes ont été proposées comme utiliser des systèmes de règles (Mangu et Brill, 1997), faire de la classification (Rozovskaya et Roth, 2010), mettre en place des modèles *n*-grammes contenant des catégories grammaticales associées à des paramètres contextuels (Golding et Schabes, 1996) ou encore des modèles *n*-grammes de tokens. C'est cette dernière solution, souvent combinée à de gros corpus, à des modèles d'erreurs, à différents types de paramètres, et/ou à des mesures de similarité, qui est la plus utilisée (Carlson et Fette, 2007; Islam et Inkpen, 2009; Stehouwer et van Zaanen, 2009; Gao *et al.*, 2010). Certaines études proposent aussi la combinaison de ces différentes approches. C'est par exemple le cas de Xu *et al.* (2011) qui utilisent un modèle trigramme et un classifieur lors des différentes étapes de leur correcteur. La majorité de ces techniques, bien que différentes, fonctionne avec la même logique. Elles tentent dans un premier temps de détecter une erreur puis, créent ensuite une liste de candidats de correction possibles pour enfin choisir la correction la plus probable.

3 Architecture du correcteur orthographique envisagé

3.1 Objectifs généraux

Cette thèse vise à mettre en place un correcteur capable de corriger des textes provenant du web. Ce sont des textes dont la taille, le contenu, la langue et la qualité rédactionnelle sont variables. En fonction du canal choisi par l'internaute et de l'internaute lui-même, le message sera plus ou moins bruité. De plus, si on observe plus attentivement le contenu de certains d'entre eux (voir l'exemple 2), on constate qu'on ne peut se restreindre à un module de correction uniquement lexical ou grammatical.

(2) regardée vraimen se don vous avez besoin et ne vous fait pas avoir par leurre pub com quoi il se souci des leur clientèle allée voir vite ailleur

Peu d'approches procèdent à une correction à la fois lexicale et grammaticale (cf. cependant Carlson et Fette (2007)). Dans cette optique, nous envisageons un correcteur contextuel. Les différentes études faites sur la correction se cantonnent souvent à une seule méthode, proposant ensuite de faire varier la valeur de certains paramètres afin d'optimiser les résultats de leur correcteur. Ne sachant quelle est la meilleure approche, nous songeons à aborder le problème en comparant et combinant ces dernières. Nous nous attarderons donc aussi bien sur des systèmes utilisant des *n*-grammes (phonétiques ou non) que sur des systèmes par règles ou sur des systèmes d'alignement automatique. Nous n'écarterons pas non plus la possibilité de nous appuyer sur des ressources lexicales plus adaptées à nos corpus ou d'inclure des informations propres à la sémantique distributionnelle. De cette manière nous pourrons ensuite choisir la combinaison la plus performante.

Bien que les travaux décrits dans cet article ne concernent que le français, nous voudrions, à terme, un correcteur qui puisse corriger plusieurs langues. Par conséquent, il ne s'agit pas de ré-implémenter des travaux adaptés à l'anglais, mais de trouver la bonne combinaison qui nous permettra de traiter de manière efficace et indépendante plusieurs langues différentes (telles que l'anglais, le français, l'allemand, l'espagnol ou encore l'italien). Le caractère multilingue que peut avoir un correcteur n'apparaît que très peu dans la littérature (cf. cependant Reynaert (2004)).

Le corpus que nous utilisons comme objet d'étude est uniquement constitué de messages client très divers les uns des autres. Cette hétérogénéité s'explique par le fait qu'ils proviennent de canaux différents (réseaux sociaux, forums, sites d'avis, mails, blogs, enquêtes de satisfaction). Après une rapide étude sur corpus, nous avons pu constater que le nombre et le type de faute d'un message varie en fonction du canal utilisé. Nous voudrions donc pouvoir adapter de manière automatique et dynamique notre correction au texte à corriger et ainsi être capable, par exemple, d'en proposer une plus légère pour un mail que pour un message provenant d'un réseau social.

3.2 Prétraitements nécessaires

Actuellement, nous n'avons pas encore défini tous les détails de l'architecture de notre système de correction. Nous ne donnerons donc pas d'indications supplémentaires à ce sujet. Néanmoins, le correcteur que nous voulons mettre en place étant contextuel, la qualité du texte l'entourant aura automatiquement des conséquences sur ses performances. Et ce, même si cette dernière est prise en compte dans l'implémentation du système. Nous proposons donc de procéder à une série de prétraitements qui ont pour but de normaliser en grande partie le texte, d'y détecter les mots ou groupes de mots que l'on pourrait ignorer pendant la suite du traitement ou encore de pré-corriger certains mots. Comme le montre la figure 1 ces prétraitements se partitionnent en plusieurs modules.

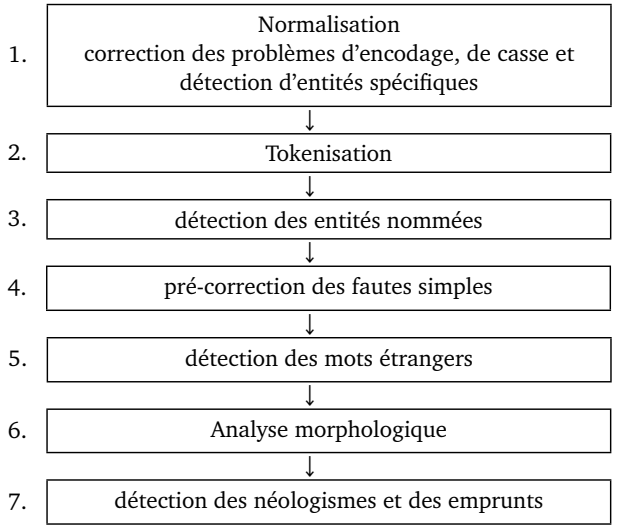

FIGURE 1 – Architecture des prétraitements nécessaires

Les modules 1, 2 et 3 visent à nettoyer le texte de toutes ses erreurs d'encodage, des fautes de casse [1] et tend à détecter les entités nommées [2] contenant de la ponctuation ou d'autres caractères spéciaux (URL, smiley, adresse mail, date,...). La reconnaissance de ces entités a lieu à

1. La correction des fautes de casse consistera entre autres à ajouter une majuscule manquante en début de phrase ou encore à renormaliser les phrases entièrement écrites en majuscules.

2. Nous reprenons ici la notion d'entité nommées telle qu'elle est utilisée dans Sagot et Boullier (2008)

cet endroit afin d'éviter certains cas qui pourraient induire le tokeniseur en erreur. Le module de détection des entités nommées (module 3), quant à lui, fait référence aux autres types d'entités nommées tels que les noms propres ou encore les acronymes. Puisque nous ne traitons pour l'instant que des textes en français, ces premiers modules peuvent être en grande partie gérés par SxPipe (Sagot et Boullier, 2008). [3]
Nous souhaitons ensuite (dans le module 4) proposer une pré-correction aux fautes aisément détectables dont la correction ne laisserait place à aucune ambiguïté (ex : *nooooon* → *non*).
Enfin, il est important de pouvoir définir, lorsque l'on rencontre un mot inconnu non intégré à une entité nommée, si ce mot appartient à la langue du texte que l'on souhaite corriger ou non (module 5). Si c'est le cas, il sera analysé morphologiquement (module 6) afin de déterminer s'il correspond, ou non, à une faute, un emprunt, un néologisme ou encore à une entité nommée qui n'aurait pas été détectée dans les modules précédents (module 7). Dans le cas contraire, il y a fortes chances pour que ce mot soit un mot dit « étranger ». Il sera donc soit laissé tel quel soit corrigé avec un faible coût de correction (ex : faute d'accent). C'est uniquement une fois que tous ces prétraitements auront été mis en place que nous pourrons nous concentrer sur notre système de correction.

4 Détecter automatiquement l'appartenance d'un mot à une langue

4.1 Présentation du module

L'un de nos modules de prétraitement (module 5) doit donc déterminer si un mot, inconnu d'un lexique de référence du français et non traité par les modules précédents, correspond à un mot « étranger ». Nous définissons cette notion de mot « étranger » de façon opérationnelle comme suit. Un mot inconnu est « étranger » si, notamment parce qu'il est un emprunt à une autre langue n'ayant pas été adapté morphologiquement, il ne doit pas faire l'objet d'une correction orthographique par un outil traitant du français. À l'inverse, les inconnus qui sont catégorisés comme « français » seront notamment des fautes lexicales dont on sera amené à essayer de corriger l'orthographe, ou des néologismes (y compris des emprunts adaptés). Par ailleurs, après avoir observé les mots inconnus qui apparaissaient dans nos corpus, nous avons constaté que la quasi-totalité des mots que nous voudrions annoter comme « étrangers » sont en réalité des mots anglais (cf. exemples 3c et 3d et table 2). Nous allons appuyer notre module sur l'approximation suivante : l'identification des inconnus « étrangers » pourra se faire à l'aide d'approches cherchant à distinguer d'une part des inconnus *anglais* et d'autre part des inconnus français [4]. Considérons les exemples suivants :

(3) a. Il y a **marqer ke** la carte n'est pas disponible

b. J'ai tellement débranché pour **rebooter**, **reseter** que je pourrais le faire les yeux fermés

c. y a-t'il la possibilité de les utiliser **online** ?

d. **OMFG** si **no fake go** sinon joli montage

3. Cette chaîne traite déjà d'autres langues que le français, mais elle devra être améliorée avant que nous ne l'utilisions pour traiter d'autres langues.

4. Parmi les 36 000 mots parcourus pour annoter nos mots étrangers, nous n'avons trouvé que des mots anglais.

Les mots en gras de l'exemple 3a correspondent à des mots mal orthographiés. Quant à ceux de l'exemple 3b, ils pourraient plutôt être considérés comme des emprunts adaptés. Dans ces deux cas, nous considérons que ces mots appartiennent à la langue française. Nous voudrions, par la suite, pouvoir les analyser morphologiquement afin de déterminer s'il s'agit d'emprunts adaptés, de fautes d'orthographe ou encore de néologismes. Les mots en gras des exemples 3c et 3d sont des mots anglais, donc « étranger ». Ils détiennent un sens particulier dans ces phrases et nous ne voudrions surtout pas tenter de les corriger en les considérant comme des erreurs produites par l'internaute.
Cette tâche peut sembler similaire aux travaux réalisés dans le domaine de la détection de langue. La détection de langue s'appuie souvent, soit sur des connaissances linguistiques, soit sur des méthodes statistiques (Grefenstette, 1995; Giguet, 1998). Plusieurs approches permettent de s'appuyer sur des connaissances linguistiques. On peut choisir de n'exploiter que les mots grammaticaux ou les mots les plus courts d'une langue (Johnson, 1993; Ingle, 1976), de s'appuyer sur un lexique de chaque langue, ou encore de ne prendre en compte que les suites de lettres qui n'apparaissent que dans une langue (Dunning, 1994). En parallèle, on peut aussi choisir de s'appuyer sur des méthodes plus statistiques telles que les *n*-grammes simples (Cavnar et Trenkle, 1994; Martins et Silva, 2005), en utilisant l'entropie relative (Sibun et Reynar, 1996) ou associés à des modèles de Markov (Dunning, 1994). Par exemple, sur des données issues du Web, qui sont plus proches de nos données que des textes plus littéraires, Martins et Silva (2005) rapportent une précision variant de 80% à 100% pour la classification de pages parmi 12 langues en utilisant les *n*-grammes ainsi que quelques heuristiques complémentaires. Bien que notre module a une tâche similaire aux travaux décrits ci-dessus, il se distingue de ces derniers de part le fait qu'il ne veut non pas connaître la langue d'un texte, mais uniquement celle d'un mot isolé au sein d'un texte dont la langue est connue. Nous ne pouvons donc prendre en compte le contexte de ce mot.

4.2 Mise en place du système de classification

Les systèmes de classification permettent de prédire la valeur/classe d'un objet à partir d'un ensemble de données. Dans notre cas, nous voulons prédire la classe (mot français ou mot étranger) d'un mot inconnu. Pour mettre en place notre système il nous faut :
– définir un corpus duquel on extraira diverses informations (par exemple, des fréquences, divers types de traits, etc.) ; ces informations serviront de données d'entraînement ; ces données dépendent à la fois du corpus choisi et de la façon dont elles en sont extraites ;
– sélectionner un système de classification permettant d'apprendre un modèle probabiliste à partir des données d'entraînement ;
– mettre en place un corpus d'évaluation représentatif de nos données réelles qui nous permettra d'évaluer notre module.

4.2.1 Construction des données d'entraînement

Pour mettre en place notre module, nous avons besoin de choisir un corpus d'entraînement pour chaque classe de notre classifieur. Il nous faut donc un corpus de mots français et un corpus de mots anglais (qui correspond à la classe des mots étrangers). L'extraction des données d'entraînement peut se faire à partir de différents types de corpus.

– Des ressources lexicales utilisées comme corpus, qui prennent en compte les formes fléchies de chaque lemme du français et de l'anglais. Nous avons ainsi réalisé des expériences préliminaires en utilisant le lexique Le*fff* du français (Sagot, 2010) et sa contrepartie anglaise EnLex.
– Des corpus correspondant à des textes « propres », comme des corpus journalistiques, des extraits de livres ou encore des articles Wikipédia.
– Des corpus plus « bruités », dits *produits par l'utilisateur* (*User-Generated Content*). C'est par exemple le cas des corpus WaCKy Baroni *et al.* (2009), construits par l'aspiration d'un grand nombre de pages Internet, qui vont d'articles journalistiques à des messages extraits de forums. Ce type de corpus contient donc en quantité importante des textes de qualité dégradée et contenant de nombreux néologismes, plus proches de ceux que nous avons à traiter que les corpus « propres ». Des corpus WaCKy ont été constitués pour plusieurs langues. Le corpus WaCKy de référence pour le français est frWaC, celui pour l'anglais est ukWaC.

Nous avons entraîné nos systèmes de classification sur des corpus relevant de ces trois cas. Le corpus provenant de ressources lexicales nous semblait pertinent de part sa richesse en mots distincts. Néanmoins, des évaluations préliminaires sur le corpus de référence nous ont permis de constater que la présence de trop nombreux mots rares dans les données d'entraînement impactait la qualité de nos modèles. Un second corpus contenant des textes assez propres, constitué de Wikipedia (français et anglais), du corpus Brown (anglais) et du corpus de l'Est Républicain (français) s'est avéré légèrement meilleur mais insuffisant. Cela s'explique par la différence de qualité rédactionnelle présente entre les données d'entraînement et les données de référence. Nous avons obtenu de meilleurs résultats en nous entraînant sur les corpus WaCKy, plus proches des nôtres. Les systèmes de classification introduits par la suite auront, par conséquent, tous été entraînés sur ces corpus. Les mots contenus dans frWaC ne sont pas tous des mots en français : il contient également des mots inconnus, dont de nombreux mots français mal orthographiés (comme dans nos corpus) ainsi que des mots étrangers. Toutefois, nous faisons l'approximation consistant à ignorer ces derniers et à considérer frWaC comme un corpus approprié pour apprendre ce qu'est un mot « français ». Autrement dit, un mot à annoter « anglais » sera plus caractéristique de ukWaC que de frWaC, même s'il apparaît dans ce dernier.

4.2.2 Systèmes mis en place

Baseline Nous avons, dans un premier temps, implémenté un système de classification naïf qui repose simplement sur les fréquences des tokens dans frWaC et dans ukWaC. Nous faisons l'hypothèse que les mots présents dans les données que nous avons à traiter ont de fortes chances d'apparaître dans ces corpus. Notamment, un mot « français » mal orthographié a de bonnes chances d'apparaître avec la même orthographe fautive dans frWaC, mais pas dans ukWaC, ou du moins à une fréquence moindre.

Notre baseline fonctionne de la manière suivante. Lorsqu'un mot est inconnu du lexique de référence utilisé, ici le Le*fff*, on compare son nombre d'occurrences dans les corpus ukWaC et frWaK. Si ce mot est plus fréquent dans le premier, on considère qu'il est étranger, dans le cas contraire, il est annoté comme français. Si le mot inconnu n'apparaît dans aucun des deux corpus, ce système naïf lui attribue aléatoirement une des deux langues.

Système proposé Pour aller au-delà de ce classifieur naïf, nous avons défini plusieurs jeux de traits permettant de modéliser les mots présents dans nos corpus d'entraînement. Les expériences présentées ci-dessous reposent sur trois jeux de traits (illustrés dans la table 1).

– Comme indiqué plus haut, les travaux sur la reconnaissance de langue s'appuient beaucoup sur les systèmes *n*-grammes. Nous avons donc extrait des mots du frWaC et ukWaC les *n*-grammes qui les composent [5] et nous avons construit un trait booléen pour chaque *n*-gramme obtenu. Nous avons fait diverses expériences en utilisant soit une seule classe de *n*-grammes (par exemple, seulement les trigrammes), soit deux (par exemple, les bigrammes et les trigrammes).
– Nous avons également rajouté des traits booléens issus de la discrétisation du rapport entre la fréquence d'un mot donné dans le frWaC et celle du même mot dans le ukWaC [6]. L'utilisation de ces traits est indiquée par l'abréviation « freq-ratio » dans les tableaux ci-dessous.
– Un inconvénient des traits de type freq-ratio est qu'ils ne prennent pas en compte la significativité statistique du rapport de fréquences : un mot attesté une fois dans l'un des corpus et deux fois dans l'autre sera dans la même classe qu'un mot attesté 1 000 fois dans le premier et 2 000 fois dans le second. C'est pourquoi nous avons également réalisé des expériences en utilisant comme traits des classes de t-test permettant de mesurer la significativité de l'écart entre la fréquence d'un mot dans frWaC et celle de ce même mot dans ukWaC [7]. Ces traits sont indiqués par la mention « t-test » dans les tableaux ci-dessous.

Inconnu	fréq. ukWaC/frWaC	2-grammes	freq-ratio [8]	t-test[8]
access	797 734/8 898	_a, ac, cc, ce, es, ss, s_	F-R6 $(33 \leq 89 < 100)$	TT5 $(-6476 \leq -728 < -11)$
vanquish	641/51	_v, va, an, nq, qu, ui,...	F-R7 $(3 \leq 12 < 33)$	TT5 $(-6476 \leq -18 < -11)$
activié	0/27	_a, ac, ct, ti, iv, vi,...	F-R1 $(0 \leq 0 < 0,01)$	TT3 $(3,6 \leq 6,2 < 16)$
regler	12/970	_r, re, eg, gl, le, er, r_	F-R2 $(0,01 \leq 0,01 < 0,05)$	TT1 $(16 \leq 37 < 7425)$

TABLE 1 – Illustration des traits de notre module : *n*-grammes (ici bigrammes), freq-ratio et t-test

Nous avons alors construit nos données d'apprentissage en assignant à tous les mots de frWaC (resp. ukWaC) la classe « français » (resp. « anglais ») et diverses combinaisons des traits ci-dessus. Nous avons entraîné sur ces différents jeux de données d'entraînement le système de régression binomiale implémenté dans MegaM [9] (Daumé III, 2004). Chaque combinaison de traits (par exemple, bigrammes + t-test) conduit à un modèle différent. Face à un inconnu à classifier, il suffit alors d'en extraire les traits correspondant à l'un des modèles puis de calculer la prédiction de ce modèle au vu de ces traits.

4.3 Évaluation

4.3.1 Données de référence

Nous avons constitué manuellement, à partir de notre corpus, des données d'évaluation contenant des mots annotés comme « français » et comme « étranger ». Afin que ces mots soient représentatifs

5. *n* varie de 1 à 4 dans les résultats rapportés ici.
6. Cette discrétisation a été réalisée comme suit : les données conduisant à un rapport de fréquences supérieur ou égal à 1 ont été réparties en 4 classes de taille identique ; il en est de même pour les données conduisant à un rapport de fréquences inférieur à 1. On obtient donc 8 classes au total.
7. La discrétisation a été réalisée également en deux fois 4 classes, avec un t-test de 0 comme pivot.
8. Traits représentés ainsi : « Classe (x<val<y) » : Classe contenant un mot dont la valeur est comprise entre x et y.
9. http ://www.cs.utah.edu/ hal/megam/

des cas d'inconnus à traiter, nous avons conservé, pour chaque classe (« français » et « étranger »), les 564 premiers inconnus rencontrés dans notre corpus [10]. Nos données de référence, constituées par conséquent de 1 128 mots inconnus, correspondent à l'ensemble de ces inconnus annotés. Elles contiennent donc autant de mots « français » que de mots « étrangers ». Un échantillon des 15 premiers mots inconnus annotés de chaque type est représenté dans la table2.

Inconnus annotés comme « français »	*abitacle, abonment, achet, actionet, activié, additionels, adébloquée, adhére, adhére, adoore, afi, agreabl, aimmerais, aixenProvence, ala*
Inconnus annotés comme « étranger »	*access, add, advanced, advantage, adventure, adventures, after, again, agency, agreement, airline, airport, all, allOffTheLights, american*

TABLE 2 – Échantillon de mots inconnus présents dans les données de référence

Les mots inconnus annotés français correspondent en grande partie à des fautes d'orthographe et, à quelques néologismes et emprunts. Ceux annotés anglais sont plutôt des mots utilisés dans le monde du web, des noms de jeux ou films non traduits et des mots mal orthographiés. Ils correspondent aussi à des mots composant une phrase isolée en anglais au sein d'un message en français. Ces derniers apparaissent peu dans nos textes. La quantité de mots inconnus annotés comme français et étranger n'est donc pas représentative de leurs fréquences d'apparition. Néanmoins, les considérer directement comme « français » conduirait à faire de la sur-correction. Bien que ces mots n'apparaissent pas dans les dictionnaires du français, beaucoup sont présents dans les corpus WaCKy. La répartition des mots inconnus présents ou non dans ces corpus en fonction de leur annotation est représentée à la table 3.

	Annotés « français »	Annotés « étranger »	Total
Mots présents dans les corpus WaCKy	402	556	958
Mots absents des corpus WaCKy	162	8	170
Total	564	564	1 128

TABLE 3 – Informations quantitatives sur les données de référence

En constituant manuellement ce corpus, nous avons choisi d'évaluer les performances de notre système de manière isolée. Cela suppose le bon fonctionnement des étapes préalables à ce module. Il est donc évident que, si l'un de ces prétraitements génèrent des erreurs, les résultats de ce module seront impactés et seront très probablement moins bons.

4.3.2 Résultats et discussion

Cette évaluation s'appuie sur les taux d'erreurs de chacun de nos modèles avec notre corpus de référence. La présence d'un mot dans frWaC et ukWaC peut avoir de réelles conséquences sur nos résultats. Pour cette raison, nous présenterons tout d'abord les résultats obtenus avec uniquement les mots de notre corpus de référence existants dans les corpus frWak et ukWak puis, les résultats obtenus avec ceux absents [11]. Considérons la table 4 qui contient les taux d'erreurs de nos modèles en fonction des traits choisis et combinés. On constate qu'utiliser la fréquence

10. Environ 26 000 mots ont été parcourues pour trouver les inconnus « français » et près de 34 000 pour les « étrangers »
11. Les données concernant les mots de notre corpus de référence présents ou non dans les corpus WaCKy sont indiquées table 3.

d'un mot améliore considérablement nos résultats si ce mot est présent dans les corpus WaCKy. Avec les unigramme, ce taux d'erreur passe ainsi de 0,305 à 0,066. On constate, par ailleurs, que notre baseline, s'appuyant sur la fréquence des mots connus du corpus WaCKy, obtient un moins bon taux d'erreur (0,073) que ceux obtenu par notre système. Ce constat est satisfaisant puisqu'il illustre le fait que l'entraînement de nos traits *n*-grammes a un effet positif sur nos résultats.
Nous avons ensuite évalué les mots absents des corpus WaCKy (cf. la table 5). Dans ce contexte, les résultats obtenus avec nos traits de fréquence et notre baseline (50%) ne sont pas significatifs puisqu'ils s'appuient sur les mots présents de ces corpus. Les résultats des *n*-grammes seuls ne sont, quant à eux, pas surprenants puisqu'ils sont similaires à ceux des mots connus (table 4).

	n-grammes uniquement	*n*-grammes + freq-ratio	*n*-grammes + t-test
1-gramme	0,305	0,073	0,066
2-gramme	0,216	0,073	0,070
3-gramme	0,167	0,073	0,080
4-gramme	0,119	0,082	0,082
1 à 2-gramme	0,212	0,073	0,070
1 à 3-gramme	0,157	0,074	0,082
2 à 3-gramme	0,169	0,074	0,080

TABLE 4 – Taux d'erreur de nos modèles sur les mots présents dans le WaCKy

	n-grammes uniquement	*n*-grammes + freq-ratio	*n*-grammes + t-test
1-gramme	0,322	0,953	0,333
2-gramme	0,222	0,883	0,257
3-gramme	0,152	0,784	0,199
4-gramme	0,134	0,678	0,211
1 à 2-gramme	0,240	0,901	0,263
1 à 3-gramme	0,193	0,871	0,205
2 à 3-gramme	0,170	0,743	0,181

TABLE 5 – Taux d'erreur de nos modèles sur les mots absents du WaCKy

La dernière table présente les taux d'erreur obtenus lorsqu'on évalue la totalité de notre corpus de référence. Il montre que si nous utilisons uniquement des modèles *n*-grammes, nos modèles sont peu satisfaisants. Notre baseline, dont le taux d'erreur général est de 0,136, se révèle même meilleur. Cela s'explique par le fait que les corpus frWaC et ukWaCk contiennent beaucoup de mots que nous souhaitons annoter (958/1128). De plus, on constate que nos modèles purement *n*-gramme ont un taux d'erreur qui décroit au fur et à mesure que la valeur de *n* augmente. Cela est dû au fait que plus la taille d'un *n*-gramme grandit plus on a de fortes chances d'y stocker des mots entiers. Ce sont donc les approches qui prennent en compte uniquement les modèles *n*-gramme qui produisent le plus d'erreurs. Les modèles combinant les *n*-grammes et les traits de fréquence freq-ratio et t-test sont pour ces mêmes raisons plus performants. Enfin, les résultats obtenus par la combinaison des *n*-grammes et du t-test valide bien l'idée qu'avoir des valeurs statistiques plus significatives du rapport de fréquence permet d'optimiser nos résultats.
Notre modèle atteint 90% de bonnes classifications. Ces résultats sont satisfaisants dans la mesure où, si on se réfère par exemple aux résultats de Grefenstette (1995), on constate que sur un texte français restreint à un ou deux mots il obtient 69,2% de bonnes détections avec un modèle trigramme et de 30,8% de bonnes détections avec un modèle linguistique qui ne prend en compte que les mots courts de la langue.

	n-grammes uniquement	*n*-grammes + freq-ratio	*n*-grammes + t-test
1-gram	0,307	0,206	0,107
2-gram	0,217	0,195	**0,098**
3-gram	0,165	0,180	**0,098**
4-gram	0,129	0,172	0,102
1 à 2-gram	0,216	0,198	**0,099**
1 à 3-gram	0,162	0,194	0,101
2 à 3-gram	0,169	0,175	**0,095**

TABLE 6 – Taux d'erreur de nos modèles sur la totalité de nos données de référence

5 Conclusion et perspectives

À l'heure où le traitement automatique des langues s'intéresse de plus en plus aux données réelles dites « produites par l'utilisateur » (*User-Generated Content*), nous avons expliqué la nécessité d'avoir un outil de normalisation et de correction qui permettrait un meilleur fonctionnement des outils plus adaptés à des corpus propres de type journalistique. Cette tâche délicate est nécessaire lorsqu'on travaille sur des textes de qualité dégradée puisqu'elle doit manipuler le texte précautionneusement sans l'altérer en le sur-corrigeant. L'architecture modulaire de correction décrite ici vise à réduire ces risques. Bien qu'encore en cours d'implémentation, certains modules de notre correcteur sont d'ores et déjà fonctionnels. C'est le cas du module présenté à la section 4. Ce module, qui permet de détecter si un mot qui ne figure pas dans un dictionnaire de référence du français correspond à un mot dit « étranger » , obtient des résultats satisfaisants (plus de 90% de classification correcte). Et ce, d'autant plus si on prend en compte la complexité de cette tâche de par le fait que l'on ne peut s'appuyer sur le contexte des mots qui nous intéressent. La suite de nos travaux seront dans la continuité du schéma présenté (section 3). Notre prochain objectif sera donc de mettre en place un analyseur morphologique qui nous guidera dans la classification des mots inconnus annotés « français » par le module de détection des mots étrangers. Ces mots inconnus, ayant déjà été filtrés par les modules de détection d'entités nommées et de mots étrangers, il nous ne restera plus qu'à essayer de prédire s'il s'agit d'emprunts adaptés, de néologismes ou encore de fautes d'orthographe afin d'achever cette phase de prétraitements pour les mots inconnus.

6 Remerciements

Je remercie Benoît Sagot et Geoffrey Doucy (directeur R&D de viavoo) pour tous leurs conseils.

Références

BARONI, M., BERNARDINI, S., FERRARESI, A. et ZANCHETTA, E. (2009). The Wacky Wide Web : A Collection of Very Large Linguistically Processed Web-Crawled Corpora. *Language Resources and Evaluation*, 43(3):209–226.

BEAUFORT, R., ROEKHAUT, S., COUGNON, L.-A. et FAIRON, C. (2010). A Hybrid Rule/Model-Based Finite-State Framework for Normalizing SMS Messages. *In Proceedings of the 48th Annual*

Meeting of the Association for Computational Linguistics (ACL'10), pages 770–779, Uppsala, Suède.

BOYD, A. (2009). Pronunciation modeling in spelling correction for writers of English as a foreign language. *In Proceedings of Human Language Technologies : The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume : Student Research Workshop and Doctoral Consortium*, pages 31–36, Boulder, Colorado.

BRILL, E. et MOORE, R. C. (2000). An Improved Error Model for Noisy Channel Spelling Correction. *In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'00)*, Hong Kong.

CARLSON, A. et FETTE, I. (2007). Memory-based context-sensitive spelling correction at web scale. *In Proceedings of the Sixth International Conference on Machine Learning and Applications (ICMLA'07)*, pages 166–171.

CAVNAR, W. B. et TRENKLE, J. M. (1994). N-gram based text categorization. *In Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*.

CHEN, Q., LI, M. et ZHOU, M. (2007). Improving Query Spelling Correction Using Web Search Results. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing and the Conference on Computational Natural Language Learning (EMNLP-CoNLL'07)*, pages 181–189, Prague, Czech Republic.

DAMERAU, F. (1964). A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176.

DAUMÉ III, H. (2004). Notes on cg and lm-bfgs optimization of logistic regression.

DUNNING, T. (1994). Statistical Identification of Language. *In Technical report CRL MCCS-94-273*, Computing Research Lab,New Mexico State University.

GAO, J., LI, X., MICOL, D., QUIRK, C. et SUN, X. (2010). A large scale ranker-based system for search query spelling correction. *In Proceedings of the 23rd International Conference on Computational Linguistics (COLING'10)*, pages 358–366, Beijing, Chine.

GIGUET, E. (1998). Méthode pour l'analyse automatique de structures formelles sur documents multilingues. *Thèse de doctorat, spécialité Informatique*.

GOLDING, A. R. et SCHABES, Y. (1996). Combining Trigram-based and Feature-based Methods for Context-sensitive Spelling Correction. *In Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL'96)*, pages 71–78, Santa Cruz, États-Unis.

GREFENSTETTE, G. (1995). Comparing two language identification schemes. *In Proceedings of the 3rd International Conference on the Statistical Analysis of Textual Data (JADT 1995)*, Rome, Italie.

GUIMIER DE NEEF, É. et FESSARD, S. (2007). Évaluation d'un système de transcription de SMS. *In Proceedings of the 26th International Conference on Lexis and Grammar*, Bonifacio, France.

HAN, B. et BALDWIN, T. (2011). Lexical normalisation of short text messages : makn sens a #twitter. *In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies*, pages 368–378, Portland, États-Unis.

INGLE, N. C. (1976). A language identification table. *The Incorporated Linguist*, 15(4):98–101.

ISLAM, A. et INKPEN, D. (2009). Real-word spelling correction using Google Web IT 3-grams. *In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1241–1249, Singapour. Association for Computational Linguistics.

JOHNSON, S. (1993). Solving the problem of language recognition. *In Technical report*, School of Computer Studies, University of Leeds.

KERNIGHAN, M. D., CHURCH, K. W. et GALE, W. A. (1990). A Spelling Correction Program Based on a Noisy Channel Model. *In Proceedings of the 13th conference on Computational linguistics (CoLing'90)*, pages 205–210, Helsinki, Finland.

KOBUS, C., YVON, F. et DAMNATI, G. (2008). Transcrire les SMS comme on reconnaît la parole. *In Actes de TALN 2008*, pages 128–138, Avignon, France.

KUKICH, K. (1992). Techniques for Automatically Correcting Words in Text. *ACM Computing Surveys*, 24(4):377–439.

LI, M., ZHANG, Y., ZHU, M. et ZHOU, M. (2006). Exploring distributional similarity based models for query spelling correction. *In Proceedings the 44th Annual Meeting of the Association for Computational Linguistics and of the 21th conference on Computational linguistics (ACL-CoLing 2006)*, pages 1025–1032, Syndey, Australie.

MANGU, L. et BRILL, E. (1997). Automatic Rule Acquisition for Spelling Correction. *In Proceedings of the 14th International Conference on Machine Learning (ICML'97)*, pages 187–194, Nashville, États-Unis.

MARTINS, B. et SILVA, M. J. (2005). Language identification in web pages. *In Proceedings of the 2005 ACM symposium on Applied computing*, SAC '05, pages 764–768, New York, NY, USA.

PARK, Y. A. et LEVY, R. (2011). Automated whole sentence grammar correction using a noisy channel model. *In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies*, pages 934–944.

REYNAERT, M. (2004). Multilingual Text Induced Spelling Correction. *In Proceedings of the 20th International Conference on Computational Linguistics*, Genève, Suisse.

ROZOVSKAYA, A. et ROTH, D. (2010). Generating Confusion Sets for Context-Sensitive Error Correction. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, MIT Stata Center, États-Unis.

SAGOT, B. (2010). The Lefff, a freely available and large-coverage morphological and syntactic lexicon for French. *In Proceedings of LREC 2010*, La Valette, Malte.

SAGOT, B. et BOULLIER, P. (2008). SxPipe 2 : architecture pour le traitement pré-syntaxique de corpus bruts. *Traitement Automatique des Langues*, pages 155–188.

SIBUN, P. et REYNAR, J. C. (1996). Language Identification : Examining the Issues. *In Proceedings of SDAIR-96, the 5th Symposium on Document Analysis an Information Retrieval*, pages 125–135.

STEHOUWER, H. et van ZAANEN, M. (2009). Language models for contextual error detection and correction. *In Proceedings of the EACL 2009 Workshop on Computational Linguistic Aspects of Grammatical Inference (CLAGI'09)*, pages 41–48, Athènes, Grèce.

TOUTANOVA, K. et MOORE, R. C. (2002). Pronunciation Modeling for Improved Spelling Correction. *In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 144–151, Philadelphie, États-Unis.

XU, W., TETREAULT, J., CHODOROW, M., GRISHMAN, R. et ZHAO, L. (2011). Exploiting Syntactic and Distributional Information for Spelling Correction with Web-Scale N-gram Models. *In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1291–1300, Edinburgh, Royaume-Uni.