Normalisation de textes par analogie: le cas des mots inconnus

Marion Baranes[1,2] Benoît Sagot[2]
(1) viavoo, 92100 Boulogne Billancourt
(2) Alpage, INRIA & Université Paris-Diderot, 75013 Paris
{prenom.nom}@inria.fr

Résumé. Dans cet article, nous proposons et évaluons un système permettant d'améliorer la qualité d'un texte bruité notamment par des erreurs orthographiques. Ce système a vocation à être intégré à une architecture complète d'extraction d'information, et a pour objectif d'améliorer les résultats d'une telle tâche. Pour chaque mot qui est inconnu d'un lexique de référence et qui n'est ni une entité nommée ni une création lexicale, notre système cherche à proposer une ou plusieurs normalisations possibles (une normalisation valide étant un mot connu dont le lemme est le même que celui de la forme orthographiquement correcte). Pour ce faire, ce système utilise des techniques de correction automatique lexicale par règle qui reposent sur un système d'induction de règles par analogie.

Abstract. Analogy-based Text Normalization : the case of unknowns words. In this paper, we describe and evaluate a system for improving the quality of noisy texts containing non-word errors. It is meant to be integrated into a full information extraction architecture, and aims at improving its results. For each word unknown to a reference lexicon which is neither a named entity nor a neologism, our system suggests one or several normalization candidates (any known word which has the same lemma as the spell-corrected form is a valid candidate). For this purpose, we use an analogy-based approach for acquiring normalisation rules and use them in the same way as lexical spelling correction rules.

Mots-clés : normalisation textuelle, correction orthographique, analogie.

Keywords: Text normalization, Spell checking, Analogy.

1 Introduction

La qualité orthographique d'un texte peut avoir de nombreux impacts sur les analyses faites en traitement automatique de la langue. Les outils pour le français par exemple sont très souvent conçus pour traiter du français standard. La qualité de leurs analyses et leurs résultats se retrouvent par conséquent assez dégradés dès qu'il s'agit de traiter des textes plus bruités. Le système présenté ici tend à améliorer la qualité d'un texte donné pour une tâche d'extraction d'informations. L'outil effectuant cette tâche réalise ses analyses en s'appuyant sur des séquences de mots représentés par leurs lemmes. Notre objectif est d'améliorer la qualité des textes qu'il traite avant leur passage dans l'outil en les normalisant.

La normalisation est une tâche qui a pour but de proposer, pour chaque mot considéré comme « fautif », sa forme normativement correcte ou une forme qui lui est flexionnellement liée. Nous considérons qu'une forme est « fautive » si son orthographe peut gêner une analyse ultérieure. Notre travail se concentre sur la normalisation des fautes lexicales, qui correspondent à des mots inconnus de notre lexique de référence, ici, le L*efff* (Sagot, 2010), en laissant de côté pour l'instant les fautes grammaticales non flexionnelles, où un mot est remplacé par un autre mot connu du lexique et de lemme différent (ex : *sont/font* ou *et/est*). Enfin, nous voulons normaliser un texte et non le corriger, traiter les fautes grammaticales flexionnelles (fautes d'accord ou de flexion) n'est donc pas pertinent. Bien que la majorité des formes que nous considérons ici comme fautives correspondent à des fautes lexicales, il peut aussi s'agir d'une forme réduite volontairement (cf. *changemt* pour *changement*). De manière générale, si nous voulons normaliser une forme « fautive », deux possibilités s'offrent à nous. Nous pouvons soit tenter de la normaliser comme le ferait un outil de correction automatique, en proposant sa forme normativement correcte (ex : *téléphonnez* → *téléphonez*), soit la remplacer par une forme qui partage le même lemme que sa forme attendue (ex : *téléphonnez* → *téléphoner* ou *téléphoniez*). Ce dernier rattachement ne provoquera aucun bruit puisque nous cherchons ici à améliorer un texte pour qu'il puisse être analysé par une tâche ultérieure qui, comme indiqué plus haut, s'appuie non pas sur les tokens mais sur les lemmes.

Notre objectif est donc de transformer un texte bruité en remplaçant les mots « fautifs » qu'il contient par leur correction idéale ou par une forme fléchie du lemme qui était attendu. Pour ce faire, nous faisons le postulat qu'en utilisant

des techniques de correction automatique nous obtiendrons un bon système de normalisation. Notre système, inspiré des systèmes par analogie, fonctionne ainsi à l'aide de règles de correction apprises et pondérées automatiquement. Naturellement, une tokenisation préalable est nécessaire. De plus, certains mots « fautifs » ne pourront pas être corrigés par analogie, et ne sont parfois pas véritablement fautifs (les abréviations, par exemple). Nous appliquons donc un certain nombre de traitements préalables avant de mettre en œuvre notre système de normalisation par analogie proprement dit.

Comme ce système est inspiré des systèmes de correction automatique lexicale et des systèmes utilisant l'analogie, nous commencerons par dresser un état de l'art de ces notions à la section 2. Puis, nous détaillerons le système présenté et les différents modules qui le composent à la section 3, y compris les traitements préalables à son application. Enfin, nous évaluons notre système à la section 4.

2 État de l'art

Le système de normalisation présenté ici s'inspire de techniques de correction automatique et plus particulièrement de la correction des fautes lexicales. Les travaux sur la correction lexicale ont débutés dans les années 1960 (Blair, 1960; Damerau, 1964). Au fil du temps et des avancées technologiques, ces travaux ont beaucoup évolué (Kukich, 1992; Mitton, 1996, 2010). L'état de l'art proposé ci-dessous, n'a donc pas pour objectif d'être exaustif mais plutôt de donner un aperçu des techniques utilisées. Les premiers systèmes proposés se limitaient au mot à corriger sans prendre en compte leurs mots avoisinants. Ils fonctionnaient principalement à base de règles typographiques et de distance d'édition (Damerau, 1964; Kernighan *et al.*, 1990) ou avec un système de vérification dans le lexique plus tolérant (Oflazer, 1996). Puis, le contexte a commencé à être pris en compte. Pour ce faire, certains travaux se sont appuyés sur des modèles de langue n-grammes de mots (Brill et Moore, 2000; Carlson et Fette, 2007; Park et Levy, 2011), phonétiques (Toutanova et Moore, 2002), ou encore sur des modèles de langue n-grammes qui combinaient ses deux caractéristiques (Boyd, 2009). Ces n-grammes étant, par la suite, souvent associés à d'autres paramètres tels que la catégorie grammaticale, la transcription phonétique ou encore la longueur du mot à corriger. Des études différentes ont préféré s'appuyer sur des mesures distributionnelles. C'est par exemple le cas de Suignard et Kerroua (2013), qui se sont appuyés sur les travaux de Li *et al.* (2006). Enfin d'autres approches, comme celle proposée par Yvon (2011) s'appuient plutôt sur des modèles probabilistes.

Le choix des techniques utilisées est souvent guidé par l'objectif visé par le correcteur. Un correcteur qui cherche à corriger des fautes grammaticales en plus des fautes lexicales (comme ceux proposés par Carlson et Fette (2007) et Yvon (2011)) devra automatiquement être plus robuste, la détection d'erreurs étant plus complexe à réaliser et la génération de candidats de correction plus risquée. Par ailleurs, si le correcteur en question doit traiter des textes plus dégradés (SMS, forum, blogs, etc.), les approches varieront aussi. On pourra, dans ce cas, ajouter des lexiques plus spécifiques au type de fautes à traiter (Guimier de Neef *et al.*, 2007; Seddah *et al.*, 2012), prendre en compte la phonétique des mots en phonétisant le texte à traiter (Kobus *et al.*, 2008), utiliser un système de classification (Han et Baldwin, 2011) ou encore procéder à un apprentissage par alignement (Beaufort *et al.*, 2010).

Dans ce travail, nous voulons corriger des mots sans prendre en compte le contexte dans lequel ils apparaissent. C'est pour cette raison que nous nous sommes intéressés aux systèmes par règles. Ces systèmes reposent souvent sur la notion de distance d'édition introduite par Damerau (1964) puis Levenshtein (1966). Cette notion met en oeuvre quatre types de règles (l'insertion et la suppression d'une lettre, la substitution d'une lettre par une autre, l'inversion de deux lettres consécutives). L'idée est de s'appuyer sur ces opérations, que l'on peut donc considérer comme des règles de correction, pour passer d'un mot mal orthographié à sa forme attendue. Cette méthode a été reprise et améliorée par de nombreux travaux. Des règles spécifiques aux fautes de proximité clavier sont proposées (Sagot et Boullier, 2008). Des connaissances linguistiques ont également été intégrées. Par exemple, Véronis (1988) et Sagot et Boullier (2008) traitent les fautes dites phonétiques (ex : remplacement de o par eau). Le contexte dans lequel s'applique la règle peut être pris en compte (Kernighan *et al.*, 1990). Par ailleurs, la pondération de ces règles ne se fait plus systématiquement en fonction du nombre d'opérations effectuées sur le mot. Le choix du poids d'une correction se fait, manuellement ou non, en fonction de l'opération effectuée et des lettres concernées par cette opération (Véronis, 1988; Mitton, 1996; Sagot et Boullier, 2008).

Ces règles s'appuient majoritairement soit sur des règles simples que l'on tente d'appliquer sur toutes les lettres d'un mot inconnu, soit sur des règles plus spécifiques (proximité clavier, phonétique) souvent listées manuellement. Notre objectif est d'acquérir automatiquement ces règles et de les pondérer en fonction de la fréquence de la faute correspondante. Pour cela, nous nous inspirons de techniques d'apprentissage par analogie pour induire des règles de correction.

L'analogie a déjà été employée pour plusieurs tâches relevant du traitement automatique des langues. Elle peut ainsi être utilisée pour des tâches de traduction automatique (Lepage et Denoual, 2005) ou de recherche d'informations (Moreau

et al., 2007), pour regrouper les mots d'un lexique en famille morphologique (Hathout, 2010) ou encore pour l'analyse morphologique (Stroppa et Yvon, 2005; Lavallée et Langlais, 2011). Cette notion permet d'établir un rapport de proportionnalité entre des unités linguistiques. Formellement, elle peut être représentée ainsi : « $a : b :: c : d$ », ce qui signifie « a est à b ce que c est à d ». Le couple a, b entretient ainsi une relation d'analogie avec c, d. En morphologie, par exemple, on pourra construire des analogies comme « *tutoiement* : *tutoyer* :: *vouvoiement* : *vouvoyer* ». Nous ne définirons pas ici dans le détail la notion d'analogie. Le lecteur intéressé pourra se référer aux travaux de (Lepage, 1998, 2000), de (Stroppa et Yvon, 2006) ou encore de (Lavallée et Langlais, 2009). Dans ce travail, l'analogie nous permettra d'analyser et de rattacher un mot inconnu dont l'orthographe est considérée comme « fautive » à son orthographe attendue par analogie avec des couples (forme fautive, forme corrigée) connues.

3 Système proposé

Les formes que nous pourrions considérer comme « fautives » sont nombreuses et de types très distincts. Toutefois, les prétraitements effectués au préalable de ce module nous permettront d'écarter à terme les fautes spécifiques aux corpus de type SMS ou les formes trop éloignées de leurs orthographes normativement correctes (ex : *tjs* ou *pk*). Ainsi la majorité des fautes restantes seront souvent dues à des fautes de proximité clavier (ex : *znalise*), de duplication de lettres (ex : *anallyse*), phonétiques (ex : *analize*) ou encore provenant d'un mécanisme de réduction volontaire de mot (ex : *pvoir*). Ces fautes semblent donc pouvoir être rapprochées d'autres fautes possibles résultant de mécanismes similaires. Nous faisons l'hypothèse, dans cet article, que la majorité de ces fautes résultent au plus d'une erreur lourde, comme montré par Damerau (1964), éventuellement associée à des fautes « légères d'accentuation, et que ces différents types de fautes peuvent être corrigées par des règles apprises automatiquement par analogie formelle. Par exemple, si nous avons déjà vu au préalable et extrait par généralisation appropriée la règle de correction permettant de passer de « *engagemt* » à « *engagement* » nous devrions être capable de prédire l'orthographe correcte de « *changemt* » (cf Figure 1).

	forme fautive		forme correcte
::	engagemt	:	engagement
::	changemt	:	?

$\rightarrow$

	forme fautive		forme correcte
::	engagemt	:	engagement
::	changemt	:	changement

FIGURE 1 – Correction du mot *changemt* par analogie

Le système proposé dans cet article utilise donc des techniques de correction par règles afin de normaliser les textes que nous souhaitons traiter. Avant de nous intéresser plus particulièrement au fonctionnement de ce dernier, nous décrirons rapidement, à la section 3.1, les traitements mis en oeuvre au préalable de notre module. Puis nous détaillerons, dans les sections 3.2, 3.3 et 3.4, l'apprentissage de nos différentes règles de correction pondérées, leur combinaison les unes avec les autres et l'application de ces dernières sur les mots considérés comme « fautifs » par notre système. L'objectif étant de proposer des candidats de correction pondérés aux formes inconnues du L*eff* que nous considérons comme potentiellement « fautives ».

3.1 Traitements préalables

Avant d'appliquer notre module de normalisation, nous effectuons plusieurs traitements tels que la tokenisation de notre texte, la détection de motifs particuliers ou encore la correction de formes spécifiques à certains types de corpus (« *tkt* » qui signifie « *ne t'inquiète pas* » par exemple). Nous réalisons ces traitements à l'aide de la chaîne SXPipe (Sagot et Boullier, 2008). Plus précisément, nous sommes partis de sa version utilisée dans le cadre du projet EDyLex (Sagot *et al.*, 2013), qui cherche à pallier certains problèmes que pose l'incomplétude lexicale et qui a notamment pour but de détecter les néologismes. Nous y avons ajouté des modules de détection de certains phénomènes non encore traités, afin de ne pas modifier des formes qui n'auraient pas de raisons de l'être. C'est par exemple le cas des autocensures (ex : *m*#%@*, *p***n*), des onomatopées (ex : *hahaha*), des interjections (ex : *hum*), des étirements (ex : *nooooon*) ou des décompositions (ex : *dé-plo-rable*). Enfin, nous avons ajouté des modules de correction/normalisation déterministe pour traiter certaines fautes fréquentes qu'une approche reposant sur des règles induites par analogie ne pourra pas traiter. Nous utilisons pour cela un lexique de corrections [1] et une courte liste d'autres formes non standard [2], notamment des abréviations (ex : *ds* ou

1. Ce lexique peut-être mis en place et complété manuellement mais nous n'utilisons actuellement que les 924 fautes courantes figurant dans la liste de correction automatique de la wikipédia (cf : `http://fr.wikipedia.org/wiki/Wikip\%C3\%A9dia:Liste_de_fautes_d\%27orthographe_courantes`)
2. L'orthographe de ces mots étant souvent voulue, nous préférons parler de formes non standard plutôt que de fautes.

cad) ou des réductions propres au langage SMS (ex : *slt* ou *tkt*). Suite à ces traitements préalables, tous les mots encore inconnus de nos lexiques sont considérés comme des mots à normaliser.

Dans un premier temps, nous montrerons comment nous avons extrait automatiquement nos règles de correction, puis nous expliquerons comment nous combinons ces règles entre elles. Enfin, nous verrons comment pondérer les candidats de correction/normalisation proposés par notre système afin de ne conserver que le ou les candidats les plus probables.

3.2 Apprentissage des règles de correction

L'apprentissage de nos règles de correction s'appuie sur un corpus de fautes lexicales annotées, le corpus WiCoPaCo (Max et Wisniewski, 2010) sur lequel nous reviendrons à la section 4.1. Nous verrons notamment comment nous en avons extrait un ensemble de couples de formes (forme fautive, forme bien orthographiée) correspondant à des fautes lexicales. Un tel ensemble constitue l'entrée de notre système, et nous cherchons à en extraire automatiquement des règles pondérées de correction. Comme indiqué précédemment, nous faisons l'hypothèse que chaque mot « fautif » est le résultat d'au plus une erreur « lourde ». Ainsi, pour générer nos règles de correction, il nous suffit d'extraire automatiquement de chaque couple de forme : le contexte gauche complet précédant la faute, la faute et sa correction, le contexte droit complet suivant la faute. Les contextes gauche et droit complets étant identiques dans la forme fautive et dans la forme bien orthographiée, il est simple d'extraire les séquences de lettres constituants une faute et représentant la correction de cette faute. Pour le couple *souevnt*::*souvent* par exemple, l'alignement se fera comme suit :

sou | ev | nt
↕ ↕ ↕
sou | ve | nt

Nous apprendrons ainsi les informations suivantes. :
– la *zone fautive* et sa contrepartie corrigée, c'est-à-dire les deux chaînes de caractères à substituer pour passer d'une forme à une autre : « $ev \rightarrow ve$ » ;
– le contexte dans lequel s'applique cette substitution : « *sou_nt* ».

Si nous conservons ces informations en l'état nous obtiendrons des règles trop spécifiques (de type : *{sou}{ev→ve}{nt}*) qui s'appliqueraient à trop peu de fautes. Afin d'éviter les cas de sous-correction et de sur-correction, nous avons construit deux jeux de règles en généralisant de deux façons différentes les règles individuelles directement extraites des fautes, ou *règles de base*. Pour définir ces deux jeux de règles, nous avons besoin de définir quelques termes. Étant donnée une zone fautive, nous appellerons dans la suite *contexte de niveau 1* le contexte formé des deux lettres encadrant immédiatement cette zone (la lettre à sa gauche et celle à sa droite). Nous appellerons *contexte de niveau 2* les deux lettres situées à une distance de 2 de la faute, c'est-à-dire les deux lettres encadrant immédiatement le contexte de niveau 1. Dès lors qu'un contexte va au-delà des limites du mot, nous utilisons la pseudo-lettre #.

Règles spécifiques Le premier jeu de règles est produit par généralisation limitée à partir des règles de base. Elles ne conservent que le type (consonne (C), voyelle (V) ou #) des lettres de contexte de niveau 2 mais laisse inchangé le contexte de niveau 1. Ainsi, si l'on rencontre la paire *souevnt*::*souvent* lors de l'apprentissage, la règle spécifique qui sera extraite est : *{Vu}{ev→ve}{nC}*.

Lorsqu'une nouvelle règle spécifique ne diffère d'une autre que par son contexte de niveau 1, nous les fusionnons : leurs contextes droits et gauches sont unifiés. Ceci constitue une généralisation par rapport aux deux règles prises isolément. Par exemple, si la faute suivante à traiter est *pievrt*::*pivert*, on obtient la règle de base *{Vi}{ev→ve}{rC}* : la règle de substitution extraite et le contexte de niveau 2 sont identiques à la règle précédente. Seul le contexte droit de niveau 1 diffère. Nous fusionnons alors les contextes de niveau 1 par disjonction, la règle obtenue étant alors notée : *{V[iu]}{ev→ve}{[nr]C}*. Cette fusion constitue une généralisation puisque cette règle couvre alors également les règles de base *{Vu}{ev→ve}{rC}* et *{Vi}{ev→ve}{nC}*, pourtant non attestées. En revanche, dès lors que le contexte de niveau 2 ne correspond pas, nous créerons une nouvelle règle. Ainsi, la faute *réserev*::*réserve* induira l'ajout d'une nouvelle règle à notre jeu de règle : *{Vr}{ev→ve}{##}*.

Règles larges Le second jeu de règles est produit par généralisation plus forte des règles de base : le contexte de niveau 2 est effacé, et seul le type des lettres du contexte de niveau est conservé. Pour la faute *souevnt*::*souvent*, la règle large induite sera donc : *{V}{ev→ve}{C}*.

Nous avons également appliqué certaines généralisations à ces deux jeux de règles, au niveau de la règle de substitution elle-même. Tout d'abord, nous avons généralisé les erreurs de duplication d'une lettre (ex : « *fautte* ») et de suppression d'une lettre doublée (ex : « *ereur* »), en les représentant comme suit :

fautte::*faute* : *{Vt}{+_→_}{e#}*
ereur::*erreur* : *{Vr}{_→+_}{eV}*

Si le symbole « + » est à droite du tiret bas, alors on duplique le contexte droit de niveau 1, s'il est à gauche, on duplique le contexte gauche de niveau 1. Selon que le « + » est à gauche ou à droite de la flèche, on couvre ainsi respectivement les cas de duplication manquante et de duplication erronnée. Nous avons par ailleurs généralisé les fautes d'accents et de cédille. Pour ce faire, nous représentons une lettre accentuée par la combinaison de deux caracères : l'accent en question suivi de la lettre concernée. Ainsi, nous noterons le mot « *arrêt* » comme ceci : « *ˆe* » et, pour la faute « arret », nous obtiendrons la règle suivante : *{Cr}{_→ˆ}{eC}*, qui couvre le rajout d'un accent circonflexe sur toute voyelle, pour peu que les contextes correspondent.

Ainsi, pour construire ces deux jeux de règles, nous parcourons chaque couple (forme mal orthographiée, forme corrigée) du corpus annoté et nous extrayons les règles spécifiques et larges comme décrit ci-dessus. Toutes ces règles sont pondérées comme suit. À chaque fois qu'une règle est extraite ou modifiée par un couple, son nombre d'« occurrences » est incrémenté de 1. Nous prenons alors le logarithme de ce nombre d'occurrences, puis nous le normalisons par transformation affine entre 0 (règle de nombres d'occurrence minimal) à 1 (règle de nombres d'occurence maximal). Le résultat constitue le poids de la règle. La table 1 illustre quelques exemples de règles de correction apprises par notre système.

TYPE DE LA RÈGLE	RÈGLE	POIDS	EXEMPLE DE FAUTES CONCERNÉES
Spécifique	{V[pfnlmctbsredg]}{_ → +_}{[aieuonyr]C}	0,970	**aten**dre → **atten**dre
	{V[rmltdvcsnpyxg]}{a → e}{[nmilu]C}	0,660	r**alant**ir → r**alent**ir
Large	{C}{io → oi}{C}	0,298	**tiol**e → **toil**e
	{[CV#]}{´ → ` }{V}	0,738	é**lé**ve → é**lè**ve

TABLE 1 – Exemples de règles de correction extraites

3.3 Génération de règles de correction génériques

Afin de pallier l'absence possible de règles de correction propre aux fautes typographiques nous avons généré un nouveau jeu de règle de correction. Par fautes typographiques nous entendons ici les fautes telles que l'ajout et la suppression d'une lettre (ex : *boneur*, *bonheure*), la substitution d'une lettre par une autre (ex : *bomheur*) et l'inversion de deux lettres ensemble (ex : *bohneur*). Ces quatre opérations sont à la base de la notion de distance d'édition (Damerau, 1964). Nous avons donc ajouter une seconde passe de correction, qui cherche tous les mots du L*efff* qui sont à une distance d'édition de 1 du mot « fautif ». Nous dénoterons ce module par le terme de *correction générique*.

Cette méthode, plus risquée que la précédente, est suceptible de provoquer du bruit dans nos résultats. C'est pourquoi la correction générique ne corrigera que les mots « fautifs » que notre système par règles n'aura pas traité. Chaque candidat proposé par le module de correction générique est pondéré par le logarithme de sa fréquence dans la Wikipedia française. Il est ensuite normalisé de façon affine de telle sorte qu'un mot inconnu de la Wikipédia, qui sera considéré comme y étant attesté 0,1 fois, obtienne un poids de 0 et que le mot le plus fréquent de la Wikipedia ait un poids de 1. Ainsi les mots les plus fréquents de la langue auront plus de chance d'être choisis comme candidat de correction lors de l'évaluation, dès lors que l'on prendra la fréquence en compte.

3.4 Génération et sélection des candidats de correction

Notre système vise à proposer les candidats de correction les plus probables pour une faute donnée, en espérant obtenir une ou plusieurs normalisations valides. Pour ce faire nous nous appuyons sur nos deux premiers ensembles de règles (spécifiques et larges). Chaque règle qui peut s'appliquer sur le mot « fautif » permet de générer un candidat de correction/normalisation, qui n'est retenu que s'il est présent dans notre lexique de référence, le L*efff*. En l'absence de candidat, nous autorisions chaque règle à être complétée par des changements d'accentuation. Pour cela, nous vérifions pour chaque candidat obtenu par l'application d'une règle si sa contrepartie non accentuée est présente dans une version préalablement désaccentuée du L*efff*. Si c'est le cas, nous considérons toutes les versions accentuées correspondantes comme des corrections possibles. Pour le mot *elevve* par exemple, nous proposerons *élevé* ou *élève* [3]. Dans le cas où nous n'avons toujours pas de candidat, nous tentons d'en produire à l'aide de nos règles de correction génériques.

3. Le nombre de fautes d'accentuation autorisé est le nombre minimum qu'il faut effectuer pour obtenir un candidat. Par exemple, pour le mot *elevvé*, seul *élevé* sera proposé car *élève* supposerait 3 modifications d'accentuation, alors qu'*élevé* n'en implique qu'une seule.

Nos deux premiers jeux de règles proposent chacuns des candidats de correction accompagnés de leurs scores de correction (cf. table 2) [4] pour un mot « fautif ». Les règles spécifiques proposeront automatiquement moins de candidats que les règles larges [5]. Par ailleurs, les scores proposés par ces deux jeux pour une même correction ne seront jamais identiques. Il faut donc pouvoir prendre en compte leur deux analyses afin d'attribuer à chaque candidat de correction un score global.

FAUTE	CANDIDATS PROPOSÉS	POIDS DES RÈGLES SPÉCIFIQUES	POIDS DES RÈGLES LARGES
onférence	conférence	0.459	0.754
	inférence	0.236	0.543
fitrage	titrage	_	0.188
	filtrage	_	0.781
	vitrage	0.001	0.094
arquéologues	archéologues	0.101	0.149
innacompli	inaccompli	0.351	0.094

TABLE 2 – Exemples de candidats (cas de base : sans changement d'accent, règles génériques non déclenchées)

Nous avons par ailleurs constaté que certaines propositions de correction correspondent à des mots assez rares dans la langue et sont, malgré tout, trop bien pondérés. Pour pallier cela, nous prenons en compte la fréquence de chaque candidat de correction dans la langue. Pour ce faire nous avons extrait un score de fréquence compris entre 0 et 1 de la même façon que pour la correction générique (cf. plus haut). Ainsi, pour évaluer, scorer et trier chaque candidat de correction c du mot d'origine w, nous prenons en compte trois paramètres :

1. le score $S_s(c, w)$ égal au poids de la règle spécifique qui permet de passer de w à c, s'il y en a une,
2. le score $S_l(c, w)$ égal au poids de la règle large qui permet de passer de w à c,
3. la score de fréquence $F(c)$ de la correction/normalisation proposée c.

Le score global d'un candidat sera ensuite calculé par combinaison linéaire de ces trois scores. Soit λ_s le coefficient assigné au score de la correction spécifique et λ_l celui assigné au score de la correction large. Nous définissons le score global d'une correction par $S_{\lambda_s,\lambda_l}(c, w) = \lambda_s S_s(c, w) + \lambda_l S_l(c, w) + (1 - \lambda_s - \lambda_l)F(c)$, où λ_s et λ_l sont compris entre 0 et 1. C'est à partir de ce score que nous définissons le meilleur ou les n meilleurs candidats pour une faute donnée. Naturellement, ce score dépend des valeurs choisies pour λ_s et λ_l. Nous avons donc fait varier ces deux paramètres lors de l'évaluation du scorage et donc du choix des candidats parmi l'ensemble des candidats proposés pour chaque faute. Au préalable, nous avons évalué la pertinence des candidats eux-mêmes.

4 Évaluation

4.1 Données d'entraînement et de test

Afin d'évaluer notre système, nous avons utilisé le corpus d'erreurs WiCoPaCo (Max et Wisniewski, 2010). Ce corpus, créé à partir des révisions des pages de la Wikipédia francophone, est composé de 72 483 erreurs lexicales et de 74 100 erreurs grammaticales qui ont été annotées comme telles dans le corpus par leurs auteurs au moyen d'un processus automatique. Chacune des fautes est associée à sa correction, effectuée par un contributeur de la Wikipédia. Puisque dans ce travail nous nous intéressons uniquement aux fautes lexicales, nous n'avons utilisé que les fautes lexicales, c'est-à-dire les fautes annotées « *non_word_error* ». Par ailleurs, nous ne voulons pas que la fréquence d'un mot puisse biaiser la pondération des règles qui en seront extraites. Nous n'avons donc conservé qu'une seule occurrence de chaque faute annotée (soit un total de 36 344 fautes). Nous avons utilisé 60% de ces dernières, soit 21 581 fautes, comme données d'entraînement pour l'apprentissage de nos deux jeux de règles [6]. Nous avons obtenu de la sorte un jeu de 4 795 règles

4. Si le candidat de correction résulte d'une correction par règle associé à une faute d'accentuation, son score de correction sera celui de la correction par règle. La faute d'accentuation ne modifie donc pas le poids de correction d'une règle.

5. Toute correction proposée par les règles spécifiques le sera par les règles larges. L'inverse n'est pas vrai.

6. Suite à plusieurs tests, nous avons constaté que prendre un corpus plus grand (90% de l'ensemble des fautes lexicales, par exemple) n'améliorait pas nos résultats de manière significative. Nous avons par ailleurs l'intention d'étendre nos tests en travaillant sur un autre corpus d'erreurs non décrit ici, en cours de développement, d'environ 20 000 fautes. Afin que nos expériences puissent être comparables, il est préférable que ces deux corpus fassent la même taille, c'est pourquoi nous n'avons conservé que 60% des fautes lexicales de WiCoPaCo pour l'apprentissage.

spécifiques et un jeu de 3 073 règles larges. Par ailleurs, environ 7,5% des fautes lexicales de WiCoPaCo (soit 2 731 fautes), sans recouvrement avec les données d'entraînement, ont été conservées pour constituer notre corpus de test.

Comme indiqué plus haut, notre système s'insère dans une chaîne complète de normalisation textuelle. Cette chaîne a notamment pour tâche, préalablement à l'application de notre système de correction par analogie, de détecter et de préserver de toute correction ultérieure différents types de mots inconnus, tels que les créations lexicales ou les mots étrangers. Toutefois, dans la mesure où nous cherchons ici à évaluer notre système de normalisation, indépendamment du reste d'une telle chaîne, nous avons d'une part désactivé les modules de détection des emprunts et des néologismes et d'autre part vérifié qu'il n'y avait pas de créations lexicales ou de mots étrangers dans nos données d'évaluation, supposées ne contenir que des fautes lexicales et leur correction. En réalité, nous en avons trouvé quelques cas ne relevant pas de fautes lexicales, mais plutôt de substitutions d'un mot par un autre. Nous avons ainsi identifié et retiré manuellement de nos données d'évaluation 38 occurrences de créations lexicales (ex : *herbologiste*, *windobe*, *zitanoisme*) ou à des mots étrangers (ex : *poesía*, *musgos*), qui ont été remplacées dans la Wikipedia par d'autres mots. Ces cas constituaient environ 1,4% des données d'évaluation, ce retrait n'a donc pas eu d'impact significatif sur nos résultats. Par ailleurs, 137 « fautes » ont été écartées parce qu'elles appartenaient déjà à notre lexique de référence, le Le*fff*, et ne constituaient pas à ce titre des candidats à la correction pour notre système (ex : *téléscope*, *soeur*). Suite à ces retraits, nos données de test contiennent 2 556 fautes lexicales pour lesquelles nous disposons d'une correction spécifiée par les contributeurs de la Wikipédia.

4.2 Résultats obtenus

Bien que nous ayons développé des techniques de correction automatique dans notre système, rappelons que notre objectif reste la normalisation textuelle. Nous avons donc évalué notre système aussi bien sur ses résultats en correction automatique qu'en normalisation. Ces deux évaluations se distinguent par leur manière de valider un candidat pour un mot « fautif » donné. En normalisation, on attendra de notre système qu'il nous propose n'importe quelle forme fléchie du lemme « correct ». En correction automatique, on n'acceptera que la forme normativement correcte du mot mal orthographié. Il se peut que, parmi les normalisations proposées pour une faute donnée, plusieurs soient valides alors qu'aucune ne corresponde pour autant à la bonne correction [7].

Nous avons évalué nos résultats en deux temps : (i) Nous nous sommes tout d'abord intéressés à la qualité des candidats proposés par notre système ; (ii) Puis nous nous sommes concentrés sur la pertinence de notre système de scorage. Pour la suite de cette évaluation, nous avons utilisé les notions habituelles de précision, rappel et f-mesure. Si l'on ne prend en compte qu'un seul candidat par faute, la précision P correspond au rapport entre le nombre de mots bien corrigés (resp. normalisés) et le nombre de mots pour lesquels nous avons proposé au moins un candidat de correction. Le rappel R sera calculé comme le rapport entre le nombre de mots bien corrigés (resp. normalisés) et le nombre de mots donnés en entrée. Si l'on cherche à prendre en compte plusieurs candidats, on considérera qu'un mot est bien corrigé/normalisé dès lors qu'une bonne correction/normalisation se trouve parmi lesdits candidats. La f-mesure F, quant à elle, sera calculée comme habituellement : $F = 2PR/(P + R)$.

4.2.1 Génération des candidats

Cette première évaluation, effectuée grâce aux données d'évaluation décrites ci-dessus, ne s'appuie sur aucun des scores proposés par notre système. Son but est principalement d'évaluer la qualité des candidats de correction/normalisation proposées par ce dernier. Elle permet de vérifier dans quelle mesure ces candidats ne contiennent pas trop de bruit, tout en proposant une correction/normalisation valide parmi les candidats proposés pour un mot « fautif » donné.

Nombre de candidats Nous avons commencé par vérifier que notre système ne proposait pas trop de candidats, notamment au niveau du module de correction générique. Nous avons donc compté les candidats obtenus avec et sans ce dernier. Le nombre de candidats proposés par notre système reste raisonnable, même avec le module de correction générique. En effet, comme le montre la figure 2, seule une faute sur quatre se voit attribuer plus de 4 candidats. Le nombre de ces derniers décroît par ailleurs assez rapidement puisque même avec le module de correction générique, près de 45% des mots « fautifs » de notre corpus de test ne se voient attribuer qu'une seule proposition de correction.

Couverture Cette figure nous montre aussi l'utilité de la correction générique. En se limitant aux seuls jeux de règles, seuls 5,3% des mots ne reçoivent aucun candidat. La correction générique permet de réduire ce taux à 3,6%. Les 93 mots

7. Pour la faute *dormion* par exemple, si notre système nous propose les candidats *dormir* ou *dormons*, nous considérerons que faute a été normalisée correctement bien que la correction attendue ne figure pas parmi les candidats.

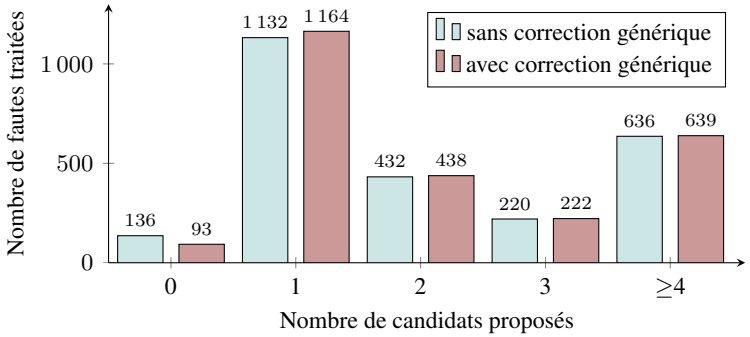

FIGURE 2 – Nombre de candidats par fautes avec et sans la correction générique

non traités ont été étudiés manuellement : 85% d'entre eux correspondent à des fautes trop complexes ou trop nombreuses (ex : *arondisemernts*, *aérdorme*) et 15% à des séquences de caractères difficilement interprétables (ex : *klàkoes*, *piwut*).

Par ailleurs, nous avons cherché à estimer la qualité des candidats proposés. Pour ce faire, étant donné un inventaire de candidats pour chaque faute, nous pouvons calculer une borne inférieure et une borne supérieure pour le système dans son ensemble. Ces bornes encadrent les performances que nous pourrons obtenir lorsque nous attribuerons des valeurs aux coefficients λ_r et λ_l, c'est-à-dire lorsque nous évaluerons la qualité de notre système de scorage à la section 4.2.2. Ces bornes ne prennent pas en compte les scores de correction de chaque candidat, et sont définies comme suit.

Borne supérieure pour le système complet Afin de déterminer le score maximum que notre système pourrait obtenir étant donné les candidats proposés, nous utilisons un oracle. Pour chaque faute, dès lors qu'une bonne correction est proposée, l'oracle le choisit. À défaut, si au moins une bonne normalisation est proposée, l'une d'entre elles est choisie au hasard par l'oracle. Si aucune normalisation valable ne se trouve parmi les candidats proposées par notre système, le choix de l'oracle importera peu. Les résultats obtenus avec un tel oracle, présentés dans le tableau 3, sont nécessairement meilleurs que ceux que nous obtiendrons avec notre système de scorage.

	Sans correction générique (éval sur 2 556 mots)		Avec correction générique (éval sur 2 556 mots)		Uniquement sur les fautes concernées par la correction générique (éval sur 93 mots)	
	correction	normalisation	correction	normalisation	correction	normalisation
précision	94,1	95,6	93,7	95,1	69,8	69,8
rappel	89,1	90,4	90,3	91,6	22,1	22,1
f-mesure	91,6	92,9	92	93,3	33,5	33,5

TABLE 3 – Évaluation des candidats de correction faite par l'oracle

Nous avons calculé la précision, le rappel et la f-mesure que notre système obtient avec et sans la correction générique afin de voir l'impact de cette dernière. Cette évaluation a été faite de manière systématique pour la tâche de correction automatique (en vérifiant que le mot attendu fait partie les candidats proposés) et pour la tâche de normalisation (en vérifiant qu'au moins un des candidats appartienne au même lemme que le token attendu). Bien que l'écart entre nos scores avec et sans la correction générique ne soit pas très élevé, cette dernière nous permet de traiter plus de mots (cf. ci-dessus), sans trop faire diminuer notre précision. Dans les deux dernières colonnes du tableau 3, nous pouvons par ailleurs faire l'observation suivante : bien que le rappel de la correction générique soit très faible (pour les raisons citées en section 4.2.1), sa précision reste acceptable (70% environ). Nous l'avons donc conservée dans la suite de nos expériences.

Par ailleurs, nous constatons que les scores obtenus pour les tâches de correction et de normalisation diffèrent peu. Cela montre que notre système de correction se trompe rarement dans la flexion du mot qu'il tente de corriger, dès lors qu'il a correctement identifié le bon lemme. On note que les scores du module de correction générique sont identiques en correction et en normalisation. Cela s'explique par le fait que la correction générique, n'effectuant que des opérations non pondérées sur un caractère, a peu de chances de proposer une correction fautive qui soit une forme fléchie du bon lemme [8].

8. Supposons ainsi que l'on demande au module de correction générique de proposer un candidat pour le mot fautif *prèt*. Il pourra proposer par exemple, grâce à une unique substitution, *prêt*, *prit*, *près*. Cet exemple illustre le fait que, de façon générale, les mots les plus « proches » (au sens de la distance de Levenshtein) d'une même faute ne sont pas tous, loin de là, des formes fléchies d'un même lemme. C'est d'autant plus vrai que le module de correction générique n'est appliqué qu'aux fautes suffisamment « complexes » ou « inattendues » pour que nos règles de correction, qui s'appliquent à plus de 96% des fautes, ne puissent pas s'appliquer.

Baseline pour le système complet Notre système de scorage doit être plus performant qu'un système de sélection aléatoire parmi les candidats proposés. C'est pour cette raison que nous avons choisi d'évaluer notre système en le laissant sélectionner aléatoirement l'un des candidats de correction associé à chaque faute. Les scores obtenus (cf. table 4) sont assez élevés pour un système effectuant un choix aléatoire. Cela montre que les propositions de correction/normalisation faites par notre système sont dans l'ensemble assez pertinentes.

	Sans correction générique (éval sur 2 556 mots)		Avec correction générique (éval sur 2 556 mots)		Uniquement sur les fautes sur lesquelles la correction générique est essayée (éval sur 93 mots)	
	correction	normalisation	correction	normalisation	correction	normalisation
précision	58,3	69,4	58,4	69,2	60,5	60,5
rappel	55,2	65,7	56,2	66,7	19,1	19,1
f-mesure	56,7	67,5	57,3	67,9	29,1	29,1

TABLE 4 – Évaluation des candidats de correction faite de manière aléatoire

4.2.2 Sélection des meilleurs candidats de normalisation

La correction générique améliorant nos résultats, nous avons choisi de la conserver dans notre système. Tous les scores figurants dans la suite de cette section proviendront donc de notre système de normalisation combinant nos jeux de règles de corrections et la correction générique.

Dans un premier temps, nous avons comparé la qualité de système sur les tâches de correction et de normalisation lorsque l'on ne conserve pour chaque faute que le candidat de score maximal [9]. Nous avons ensuite évalué notre système lorsque l'on conserve les deux puis les trois meilleurs candidats, configuration qui est pertinente si l'on décide par exemple d'en ressortir à un modèle de langage ou à tout autre module ultérieur pour effectuer le choix final, ou bien si l'étape de normalisation est un préalable à un traitement capable de prendre en entrée un graphe de mots [10]. Pour l'ensemble de ces configurations, nous avons procédé à de multiples évaluations en faisant varier les poids assignés aux jeux de règles et à la fréquence, c'est à dire en faisant varier la valeur de λ_r et de λ_l (la fréquence étant pondérée par $1 - \lambda_r - \lambda_l$) [11].

La somme des valeurs de λ_r et de λ_l doit être égale ou inférieure à 1. Nous avons donc commencé par tester toutes les combinaisons possibles de ces deux coefficients en les faisant varier entre 0 et 1 avec un pas de 0,1. Toutefois, il est apparu que ces coefficients étaient trop élevés et empêchaient la prise en compte de la fréquence. En effet, même si les scores de fréquence sont normalisés entre 0 et 1 (tout comme ceux de nos jeux de règles), ces derniers restent très faibles sur la grande majorité des candidats. Nous avons donc réévalué notre système en faisant cette fois-ci varier λ_r et λ_l entre 0 et 0,001 avec un pas de 0,0001. Nous illustrons ces derniers résultats aux figures 3 à 8 sur la précision et la f-mesure. En théorie, les résultats de notre système peuvent varier entre les bornes inférieures et supérieures calculées à la section 4.2.1, à savoir respectivement 69,2% et 95,1% pour la précision et respectivement 67,9% et 93,3% pour la f-mesure.

Les figures 3 et 4 illustrent la précision et la f-mesure obtenues par notre système lorsque l'on ne conserve que le candidat de score maximum pour chaque faute. Plus la zone d'une figure est claire, plus la précision ou la f-mesure représentée dans cette figure sera élevée. Par ailleurs, nous avons inséré des courbes de niveau (iso-précision ou iso-f-mesure) : on peut voir par exemple sur ces deux figures que les meilleurs scores pour la précision s'élèvent à plus de 87,2 (la précision maximum atteinte est en réalité de 87,7%) et ceux concernant la f-mesure à plus de 85,7 (la f-mesure maximale atteinte est en réalité de 86,1%). Ces scores sont donc bien supérieurs aux bornes inférieures calculées précédemment. La similarité entre les figures 3 et 4 s'explique par le fait que les scores de rappel obtenus, bien que plus faibles que ceux de la précision, se répartissent de la même manière en fonction des coefficients λ_r et λ_l. À titre indicatif, lorsque λ_r et λ_l varient entre 0 et 0,001, le rappel est presque partout entre 83% et 84%, avec un minimum à 81,4% (lorsque $\lambda_r = \lambda_l = 0$) et un maximum

9. Lorsque plusieurs candidats ont le même score et qu'il est maximal, l'un d'entre eux est choisi aléatoirement.
10. Des expériences conservant les cinq candidats les mieux classés ont donné des résultats très proches de ceux obtenus avec les trois meilleurs.
11. Plutôt que chercher la valeur optimale de nos trois variables de cette façon, nous aurions pu nous appuyer sur un système reposant sur la maximisation de l'entropie ou sur un perceptron et ainsi les apprendre automatiquement. Néanmoins une telle approche ne répondrait pas parfaitement à l'objectif que nous nous sommes posé ici. Nous ne voulons en effet pas classifier les corrections proposées en deux catégories mais les ordonner selon la confiance que l'on a en le fait qu'ils soient des corrections/normalisations valides de la faute à traiter. Nous avons toutefois tenté d'évaluer notre système avec des scores obtenus avec un système par maximum d'entropie. Les résultats obtenus furent néanmoins beaucoup moins bons que ceux présentés précédemment (par exemple, sur la meilleure normalisation, nous perdons près de 5.5% de précision et de F-mesure si le système prend en compte le biais, 7% sinon).

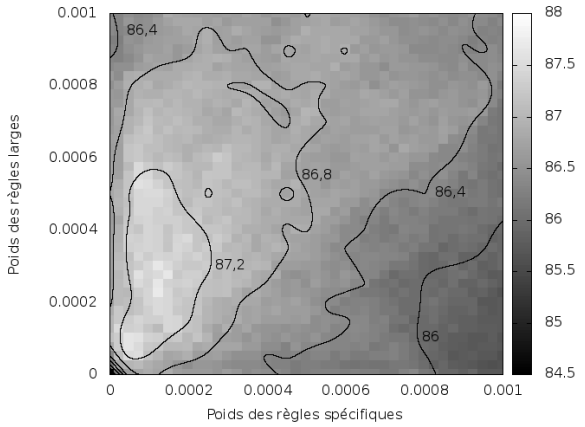

FIGURE 3 – Précision pour la meilleure normalisation

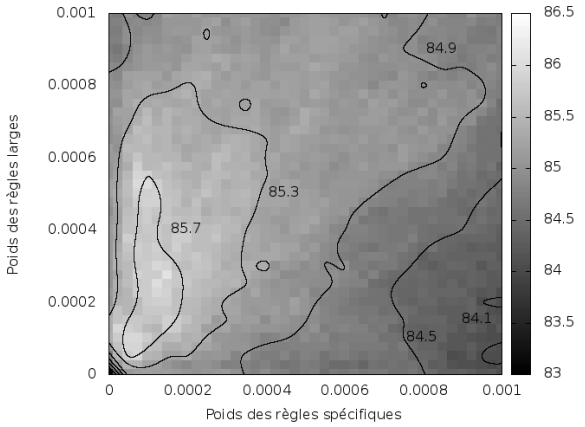

FIGURE 4 – F-mesure pour la meilleure normalisation

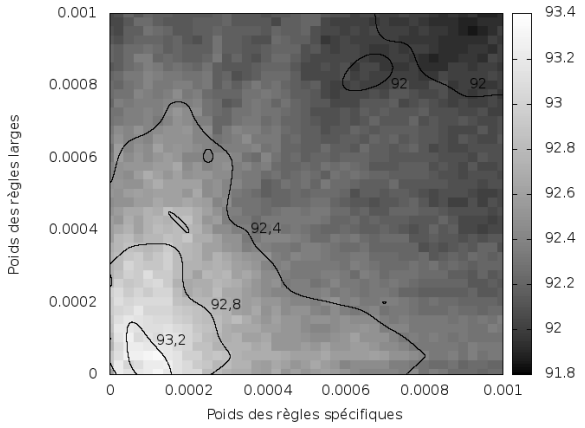

FIGURE 5 – Précision pour les 2 meilleures normalisations

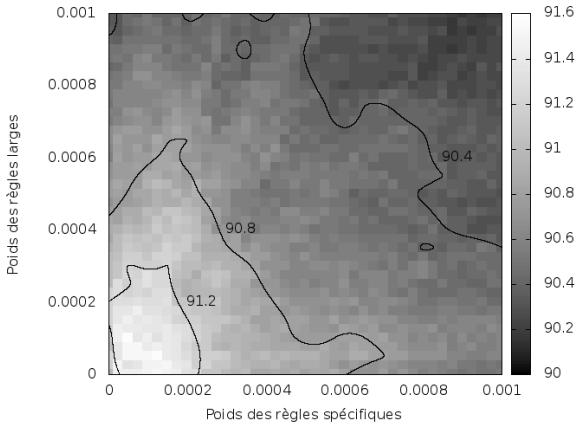

FIGURE 6 – F-mesure pour les 2 meilleures normalisations

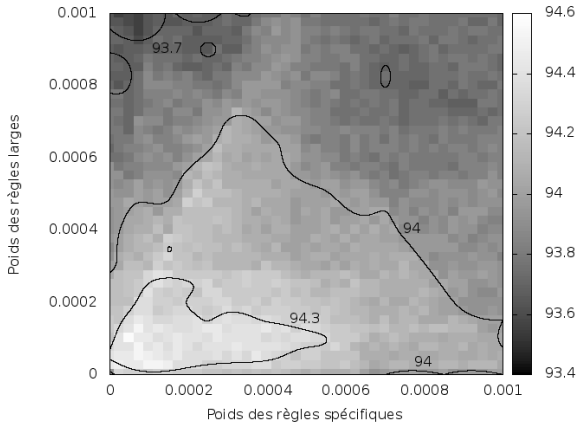

FIGURE 7 – Précision pour les 3 meilleures normalisations

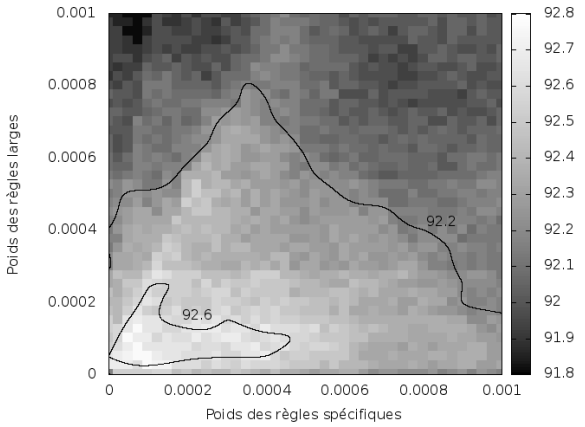

FIGURE 8 – F-mesure pour les 3 meilleures normalisations

à 84,3%. Ces deux figures démontrent clairement l'utilité des scores proposés par nos règles lorsque l'on ne garde que le meilleur candidat : si nous ne prenons que la fréquence en compte ($\lambda_r = \lambda_l = 0$), notre f-mesure est de 82,9%, soit 3,2% de moins que la f-mesure maximale). Bien que de façon moins nette, ce constat reste vrai lorsque l'on conserve les deux ou trois meilleurs candidats les mieux scorés. La f-mesure gagne ainsi 1,5% si on prend en compte les règles pour deux candidats de correction et 0,7% pour trois candidats.

Bien que le passage d'un candidat à deux nous permette d'améliorer nos scores de près de 5% absolus, le passage de deux candidats à trois est légèrement moins significatif. Cela s'explique notamment par le faible nombre de corrections

proposées pour chaque mot « fautif ». Notre système n'attribue trois candidats ou plus qu'à environ 40% de notre corpus de test (cf. figure 2). Le nombre de mots « fautifs » pour lesquels nous devons sélectionner les meilleurs candidats est par conséquent automatiquement réduit. Autre conséquence de ce fait : nos scores de précision et de f-mesure se rapprochent de la borne supérieure théorique dès la prise en compte de seulement deux candidats. En effet, notre précision s'élève à plus de 93,3% (cf. figure 5) et la f-mesure à plus de 91,5% (cf. figure 6).

Ces figures montrent que nous obtenons nos meilleurs résultats lorsque l'on donne plus de poids aux règles larges qu'aux règles restreintes. En effet, bien que les règles restreintes soit plus précises et plus fiables, elles s'appliquent moins souvent. Les règles larges peuvent ainsi détecter et corriger plus de fautes. Ces deux jeux sont donc complémentaires : nous obtenons de meilleurs scores en associant les deux. Enfin, la prise en compte de la fréquence d'un mot pour sa normalisation reste primordiale. Cela est particulièrement visible dans les figures 5, 6, 7 et 8 dans lesquelles nos scores se dégradent dès lors que λ_r ou λ_l sont au-dessus de 0,0006 environ, et ce d'autant plus que λ_r ou λ_l sont élevés.

5 Conclusion

La présence de fautes ou de formes non standard peuvent mettre à mal l'analyse d'un texte si cette analyse n'est pas adaptée aux textes bruités. Par exemple, pour un outil d'extraction d'informations, il sera plus difficile de détecter les motifs intéressants si le texte à traiter est dégradé. Le travail présenté ici vise à participer à l'amélioration des performances d'un tel outil en proposant une étape intermédiaire de prétraitement, dont l'objectif est de normaliser, au moins partiellement, des données textuelles bruitées. Pour ce faire, nous avons mis en place et évalué un système de normalisation fonctionnant à l'aide de règles de correction induites par analogie. L'objectif de ce système est de proposer un ou plusieurs candidats de normalisation pondérés pour tous les mots inconnus de nos lexiques présents dans un texte.

Les résultats obtenus étant satisfaisants, nous voudrions tout d'abord intégrer ce premier système à notre chaîne de traitement complète, qui inclut notamment des étapes de détection des néologismes et des emprunts, afin d'évaluer ses performances et son apport de manière plus globale. Par ailleurs, nous nous sommes pour l'instant limités aux fautes lexicales. À terme, nous aimerions étendre ce travail aux fautes grammaticales en prenant en compte le contexte dans lequel apparaissent ces fautes, par exemple au moyen de modèles de langage. D'autre part, ce système n'a pour l'instant été évalué que sur le français. L'apprentissage de nos règles de correction étant faite automatiquement, cette normalisation pourrait parfaitement fonctionner pour d'autres langues, pour peu qu'il existe une base de fautes corrigées pour cette langue. Cette base pourrait parfaitement, par exemple, être extraite de la Wikipedia correspondante, de la même façon qu'a été construit le corpus WiCoPaCo pour le français ou comme le propose Zesch (2012) pour l'anglais. Enfin, l'apprentissage de nos règles de correction est actuellement effectuée à partir d'un corpus de fautes annotées. Il n'existe actuellement que très peu de corpus de ce type. C'est pourquoi nous aimerions à terme pouvoir apprendre nos règles de façon non-supervisée, à partir d'un corpus brut légèrement bruité. Cela nous permettrait d'obtenir un système qui ne dépendrait d'aucune ressource particulière mis à part d'un échantillon de la langue et d'un lexique de cette langue.

Références

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 ACL'10*, pages 770–779, Uppsala, Suède.

BLAIR, C. R. (1960). A program for correcting spelling errors. *Information and Control*, 3(1):60–67.

BOYD, A. (2009). Pronunciation modeling in spelling correction for writers of English as a foreign language. *In Proceedings of the HLT-NAACL'09 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 ACL'00*, Hong Kong.

CARLSON, A. et FETTE, I. (2007). Memory-based context-sensitive spelling correction at web scale. *In Proceedings of ICMLA'07*, pages 166–171, Cincinnati, Ohio.

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

GUIMIER DE NEEF, E., DEBEURME, A. et PARK, J. (2007). TiLT correcteur de SMS : évaluation et bilan qualitatif. *In Actes de TALN'07*, pages 123–132, Toulouse, France.

HAN, B. et BALDWIN, T. (2011). Lexical normalisation of short text messages : makn sens a #twitter. *In Proceedings of JACL-HTL'11*, pages 368–378, Portland, États-Unis.

HATHOUT, N. (2010). Morphonette : a morphological network of French. *Proceedings of CoRR*, abs/1005.3902.

KERNIGHAN, M. D., CHURCH, K. W. et GALE, W. A. (1990). A Spelling Correction Program Based on a Noisy Channel Model. *In Proceedings of 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'08*, pages 128–138, Avignon, France.

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

LAVALLÉE, J.-F. et LANGLAIS, P. (2011). Moranapho : un système multilingue d'analyse morphologique fondé sur l'analogie formelle. *TAL*, 52(2):17–44.

LAVALLÉE, J. F. et LANGLAIS, P. (2009). Unsupervised morphological analysis by formal analogy. *In Multilingual Information Access Evaluation Vol. I : Text Retrieval Experiments, Proceedings of CLEF*, pages 618–625, Corfu, Greece.

LEPAGE, Y. (1998). Solving analogies on words : An algorithm. *In Proceedings of CoLing-ACL'98*, pages 728–735, Montreal, Quebec, Canada.

LEPAGE, Y. (2000). Languages of analogical strings. *In Proceedings of CoLing*, pages 488–494, Saarbrücken, Germany.

LEPAGE, Y. et DENOUAL, E. (2005). Purest ever example-based machine translation : Detailed presentation and assessment. *Machine Translation*, 19(3-4):251–282.

LEVENSHTEIN, V. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10(8):707.

LI, M., ZHANG, Y., ZHU, M. et ZHOU, M. (2006). Exploring distributional similarity based models for query spelling correction. *In Proceedings of ACL-CoLing'06*, pages 1025–1032, Syndey, Australie.

MAX, A. et WISNIEWSKI, G. (2010). Mining naturally-occurring corrections and paraphrases from wikipedia's revision history. *In Proceedings of LREC'10*, Valletta, Malta.

MITTON, R. (1996). *English Spelling and the computer*. London :Longman.

MITTON, R. (2010). Fifty years of spellchecking. *Writting Systems Research*, 2(1):1–7.

MOREAU, F., CLAVEAU, V. et SÉBILLOT, P. (2007). Automatic morphological query expansion using analogy-based machine learning. *In Proceedings of ECIR'07*, Rome, Italie.

OFLAZER, K. (1996). Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22(1):73–89.

PARK, Y. A. et LEVY, R. (2011). Automated whole sentence grammar correction using a noisy channel model. *In Proceedings of ACL-HTL'11*, pages 934–944, Portland, Oregon, USA.

SAGOT, B. (2010). The L*efff*, a freely available and large-coverage morphological and syntactic lexicon for French. *In Proceedings of LREC'10*, La Valette, Malta.

SAGOT, B. et BOULLIER, P. (2008). SxPipe 2 : architecture pour le traitement pré-syntaxique de corpus bruts. *TAL*, 49(2):155–188.

SAGOT, B., NOUVEL, D., MOUILLERON, V. et BARANES, M. (2013). Extension dynamique de lexiques morphologiques pour le français à partir d'un flux textuel. *In Actes de TALN'13*, Les Sables d'Olonne, France.

SEDDAH, D., SAGOT, B., CANDITO, M., MOUILLERON, V. et COMBET, V. (2012). The French Social Media Bank : a Treebank of Noisy User Generated Content. *In Proceedings of CoLing'12*, Mumbai, Inde.

STROPPA, N. et YVON, F. (2005). An analogical learner for morphological analysis. *In Proceedings of CoNLL'05*, pages 120–127, Stroudsburg, PA, USA.

STROPPA, N. et YVON, F. (2006). Du quatrième de proportion comme principe inductif : une proposition et son application à l'apprentissage de la morphologie. *TAL*, 47(1):33–59.

SUIGNARD, P. et KERROUA, S. (2013). Utilisation de contextes pour la correction automatique ou semi-automatique de réclamations clients. *In Actes de TALN'13*, Les Sables d'Olonne, France.

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

VÉRONIS, J. (1988). Computerized correction of phonographic errors. *Computers and the Humanities*, 22(1):43–56.

YVON, F. (2011). `spellChecker` : un système de correction automatique fondé sur des automates probabilistes. Livrable du Projet TRACE (ANR-09-CORD-023). Accessible à l'URL http ://anrtrace.limsi.fr/dev/Anr_trace_-_lot3.pdf.

ZESCH, T. (2012). Detecting malapropisms using measures of contextual fitness. *TAL*, 53(3):11–31.