TALN 2010, Montréal, 19–23 juillet 2010

Text-it / Voice-it
Une application mobile de normalisation des SMS

Richard Beaufort Kévin Macé Cédrick Fairon
CENTAL, Université catholique de Louvain, 1348 Louvain-la-Neuve, Belgique
{richard.beaufort,kevin.mace,cedrick.fairon}@uclouvain.be

Résumé. Cet article présente *Text-it / Voice-it*, une application de normalisation des SMS pour téléphone mobile. L'application permet d'envoyer et de recevoir des SMS normalisés, et offre le choix entre un résultat textuel (*Text-it*) et vocal (*Voice-it*).

Abstract. This paper presents *Text-it / Voice-it*, an application that makes it possible to normalize text messages directly from mobile phones. The application allows the user to both send and receive normalized text messages, and gives the choice between a textual (*Text-it*) and a vocal (*Voice-it*) result.

Mots-clés : SMS, normalisation, application, plugin, serveur.

Keywords: Text messages, normalization, application, plugin, server.

1 Introduction

Depuis la fin des années '90, les fournisseurs de téléphonie mobile proposent à leurs utilisateurs de communiquer en s'envoyant des messages écrits : les *textos* ou *SMS* (du nom du service lui-même, *Short Message Service*). Comme le constatent Fairon *et al.* (2006), les SMS s'écartent significativement des conventions orthographiques. Si bon nombre de ces écarts peuvent certainement être qualifiés de *fautes* d'orthographe, la majorité, pourtant, relève plutôt de la *stratégie de codage*, voire du cumul de stratégies. Les utilisateurs expérimentés, par exemple, n'hésitent pas à combiner jeux et transcriptions phonétiques (*'demain'* → *'2m1'*), squelettes consonantiques (*'toujours'* → *'tjrs'*), ou séparateurs incorrects, manquants et abusifs (*'j esper'* pour *'j'espère'*) au sein d'une même séquence (*'j'croib1kcv'* pour *'je crois bien que ça va'*). Quelles qu'en soient les raisons, ces écarts par rapport aux conventions orthographiques compliquent la gestion automatique des SMS par des applications telles que la synthèse de la parole (à destination, par exemple, des personnes malvoyantes ou des automobilistes) et la recherche d'informations (par exemple, l'extraction automatique de rendez-vous à ajouter à l'agenda).

Dans un tel contexte, on peut dès lors faire l'hypothèse que la normalisation des SMS, c'est-à-dire la réécriture des SMS en une orthographe plus conventionnelle, deviendra, dans un avenir proche, une étape à part entière du processus de communication par SMS.

Text-it / Voice-it a été conçu dans cette perspective : l'application permet d'ajouter la normalisation des SMS aux fonctionnalités déjà présentes sur le téléphone portable. Le système se divise en deux parties : un serveur et un plugin. Le serveur, invisible pour les utilisateurs, réalise la normalisation. Le plugin, installé sur le téléphone mobile, donne accès à un certain nombre d'options de normalisation et commu-

nique avec le serveur de normalisation. L'application permet aussi bien d'envoyer que de recevoir des SMS normalisés, et offre le choix entre deux résultats : le texte normalisé (*Text-it*) ou la synthèse vocale correspondante (*Voice-it*). L'algorithme de normalisation a été présenté dans le détail dans Beaufort *et al.* (2010) auquel nous renvoyons le lecteur.

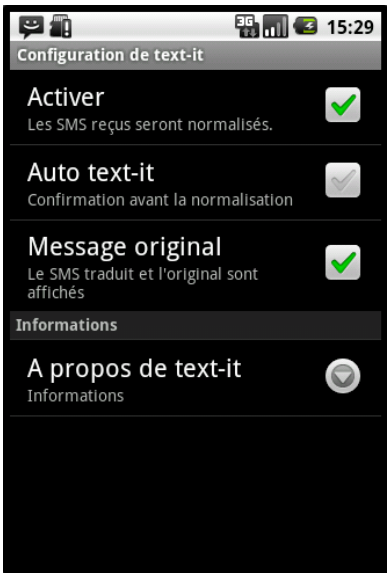
Cet article s'organise comme suit. Afin de faciliter la compréhension du fonctionnement du système dans son ensemble, nous commençons par présenter en section 2 le plugin installé sur le mobile, avant de présenter le serveur en section 3. La démonstration proposée dans le cadre de la conférence est ensuite brièvement décrite en section 4.

2 Le plugin

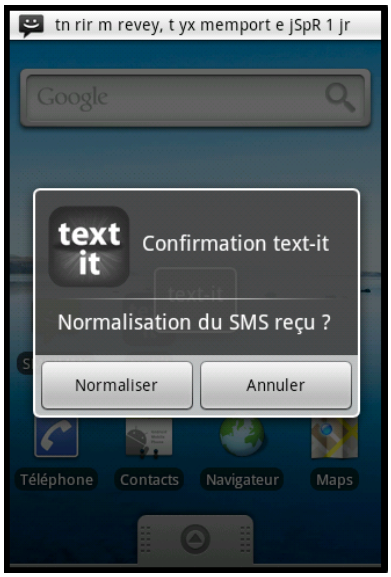
Le plugin est encore en cours de développement et n'est actuellement finalisé que pour les smartphones utilisant le système d'exploitation Android. L'objectif est cependant de le décliner pour le plus grand nombre de plateformes possible. A terme, le plugin sera téléchargeable sur des sites web spécialisés dans la distribution d'applications pour téléphones mobiles, tels que l'*Apple Store* et l'*Android Market*.

Une fois installé, le plugin autorise deux modes de fonctionnement : en réception et en émission. Le mode *réception* permet à l'utilisateur de recevoir des SMS normalisés. Il peut être configuré au travers d'une interface (figure 1*a*) qui donne accès à un certain nombre d'options :

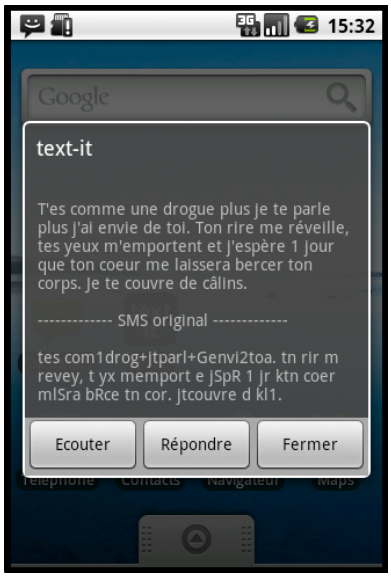
– *Activer* : cochée, cette option assure que le plugin se déclenche à la réception d'un nouvel SMS.
– *Auto text-it* : par défaut, le plugin demande confirmation avant de normaliser un SMS reçu (figure 1*b*). Lorsque cette option est cochée, la normalisation est réalisée d'office et affichée à l'écran.
– *Message original* : cette option cochée, le SMS original est affiché en plus du SMS normalisé (figure 1*c*).



(a) Panneau de configuration



(b) Demande de confirmation



(c) Affichage du SMS normalisé

FIG. 1 – Quelques interfaces du plugin.

Actuellement, nous employons la bibliothèque de synthèse de la parole fournie par *Eyes-free* (`http://code.google.com/p/eyes-free/`). La vocalisation d'un SMS est donc réalisée directement sur le téléphone, à partir du SMS normalisé, et est accessible à l'utilisateur via un bouton de l'interface d'affichage (figure 1*c*). A terme, par contre, nous prévoyons d'utiliser le système de synthèse présenté dans Beaufort *et al.* (2010), afin d'améliorer la détection et la phonétisation des unités linguistiques (URL, numéros de téléphone, etc.) présentes dans le message à synthétiser. La synthèse sera alors réalisée sur le serveur, ce qui préservera la batterie du téléphone.

Le mode *émission* est en cours de développement. Il permettra d'envoyer des SMS normalisés. Dans l'ensemble, la fonctionnalité devrait fortement ressembler à l'envoi d'un SMS standard, si ce n'est que l'utilisateur devra spécifier le type de message : textuel (*Text-it*) ou vocal (*Voice-it*). Une fois le texte rédigé, le plugin l'enverra au serveur pour normalisation, et attendra le résultat pour l'envoyer au destinataire. Un message textuel (*Text-it*) sera bien sûr envoyé sous la forme d'un SMS. Le format d'envoi d'un message vocal (*Voice-it*), par contre, reste encore à définir. Il pourrait s'agir d'un MMS ou d'un dépôt sur la boîte vocale du destinataire.

3 Le serveur

Le plugin communique avec l'application installée sur le serveur au travers d'un service web constitué de deux parties : un client et l'application en elle-même. Lorsqu'un plugin fait une requête de normalisation, un client spécifique est créé, soumet une requête à l'application, en attend la réponse et la renvoie au plugin. Pour gérer l'ensemble des interactions possibles entre les clients et l'application, nous avons utilisé des « pipes nommés ». Sous Linux, un pipe nommé fonctionne comme un pipe d'entrée/sortie classique, si ce n'est qu'il porte un nom qui l'identifie de manière univoque. Ceci offre plusieurs avantages :

1. Plusieurs processus peuvent accéder au même pipe nommé (écriture/lecture multiple).
2. Les pipes nommés se gèrent comme des fichiers standard. Les écritures simultanées sont donc traitées de manière séquentielle, ce qui évite les risques de mélange de données.
3. Un pipe nommé peut être ouvert en mode bloquant : tout processus qui y accède en lecture est dès lors bloqué tant qu'aucune information n'y est écrite, ce qui réduit la consommation CPU.
4. Un processus peut utiliser plusieurs pipes nommés, en entrée et/ou en sortie.

L'architecture présentée en figure 2 profite directement de ces avantages :

1. L'application possède un seul pipe d'entrée, où tous les clients écrivent. Elle traite séquentiellement les requêtes dans l'ordre d'arrivée, et se met en mode d'attente (pipe bloquant) dès que le pipe est vide. Une requête est un nom de fichier contenant un SMS. L'application lit donc le nom dans le pipe, ouvre le fichier correspondant et traite le SMS qu'il contient.
2. L'application possède un pipe d'erreur commun, qui lui permet de signaler à tous les clients actifs qu'un problème qui les concerne tous s'est produit (par exemple, une impossibilité de se charger).
3. Tout client crée deux pipes qui lui sont propres, et les ouvre en lecture. L'application, de son côté, ouvre les deux mêmes pipes en écriture. Le premier pipe est un pipe de sortie qui permet d'écrire le nom du fichier-résultat produit par le traitement. Le second pipe est un pipe d'erreur qui permet à l'application d'indiquer si une erreur est survenue lors du traitement du fichier déposé par le client.
4. Lorsque le client a reçu les résultats attendus, il supprime les pipes qu'il a créés et renvoie le résultat (texte ou son) vers le plugin (et donc le téléphone) qui l'a appelé.

5. Lorsque l'application reçoit une demande d'arrêt, elle supprime ses pipes d'entrée et d'erreur avant de s'arrêter.

Par souci de robustesse et afin de sécuriser le service, nous avons développé un module de surveillance, dont le principe est illustré en figure 3. Ce module réalise trois opérations :

1. Il vérifie la présence de l'application en mémoire vive (flèches pointillées). Si l'application a disparu, le système de surveillance la relance.
2. Il vérifie que l'application fonctionne correctement (flèches pleines). Le test est réalisé en fournissant à l'application un fichier dont le temps de traitement est connu. Si l'application ne répond pas dans les temps requis, un embouteillage est en cours et l'application ainsi que tous les clients associés sont interrompus. L'application est ensuite relancée.
3. Lorsque l'application est relancée 3 fois sans succès, le module de surveillance redémarre la machine, se relance lui-même et initialise l'application en mémoire.

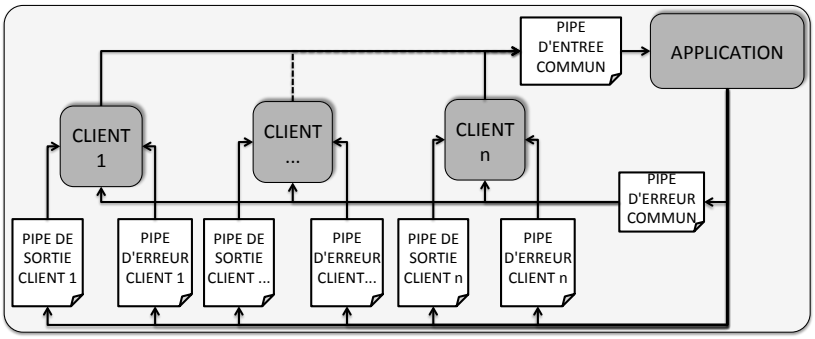


FIG. 2 – Architecture du serveur.

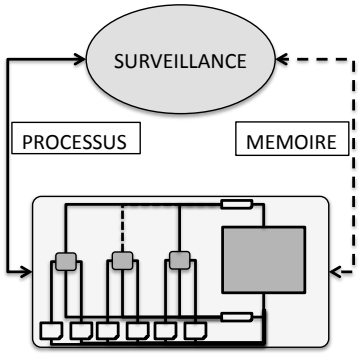


FIG. 3 – Module de surveillance.

4 La démonstration

Dans le cadre de la session de démonstration, nous proposons de tester l'application *Text-it / Voice-it* en mode *réception*. Les participants qui le souhaitent auront l'occasion d'envoyer leurs propres SMS vers un smartphone équipé du plugin.

Si une connexion SSH est disponible sur le lieu de la conférence, il sera également possible d'observer le traitement réalisé sur le serveur.

Références

BEAUFORT R., ROEKHAUT S., COUGNON L.-A. & FAIRON C. (2010). Une approche hybride traduction/correction pour la normalisation des SMS. In *Actes de la Conférence sur le Traitement Automatique des Langues (TALN'10)*. A paraître.

FAIRON C., KLEIN J. R. & PAUMIER S. (2006). *Le langage SMS : étude d'un corpus informatisé à partir de l'enquête Faites don de vos SMS à la science*. Presses Universitaires de Louvain. 136 pages.