Un segmenteur de texte en phrases guidé par l'utilisateur

Thomas Heitz*

*Université Paris-Sud XI, 91405 Orsay CEDEX
heitz@lri.fr,
http://www.lri.fr/∼heitz

Résumé. Ce programme effectue une segmentation en phrases d'un texte. Contrairement aux procédures classiques, nous n'utilisons pas d'annotations préliminaires et tirons parti d'un apprentissage guidé par l'utilisateur.

La segmentation en phrases entièrement automatisée et avec une importante proportion des corpus annotés en phrases manuellement est déjà très efficace. De même, la segmentation en phrases à l'aide de dictionnaires et de règles syntaxiques spécifiquement adaptées à un corpus donné est aussi relativement efficace.

Ce qui nous intéresse ici est donc la segmentation d'un corpus en phrases sans aucune segmentation initiale et avec l'aide de l'utilisateur pour diriger les traitements et notamment l'apprentissage. Ce que nous appelons apprentissage guidé. Le but est de minimiser le temps consacré par l'utilisateur à annoter des fins de phrases. C'est pourquoi nous utilisons au maximum les connaissances générales de l'écriture du langage naturel et nous présentons à l'utilisateur les seuls cas les plus ambigus.

Le but est d'annoter le mot précédent et suivant de chaque point suivi d'un espace afin de déterminer si la phrase doit être terminée sur ce point ou non.

L'idée qui est utilisée dans ce segmenteur est la suivante. Le mot précédent le point peut être une abréviation et dans ce cas il est fort probable que le point ne soit pas une fin de phrase. Le mot suivant le point peut être un mot toujours capitalisé, c'est-à-dire commençant par une majuscule dans tout le texte, et dans ce cas il est fort probable que le point ne soit pas une fin de phrase.

Les **annotations utilisées** pour classer les mots précédents et suivants les points suivis d'un espace sont les annotations *certain* et *impossible* qui correspondent aux mots que l'utilisateur considère comme étant (respectivement n'étant pas) certainement une abréviation terminée par un point ou un mot toujours capitalisé. L'annotation *possible* correspond aux éléments indéterminés qui deviendront *certain* ou *impossible* ultérieurement.

La procédure globale de segmentation se déroule selon les étapes suivantes :

① Établissement de **statistiques** sur les abréviations probables et les mots capitalisés probables sur le corpus complet. Notamment le nombre d'occurrences avec et sans point final et avec et sans majuscule initiale.

② **Annotation automatique** sur un extrait du corpus des abréviations à l'aide de listes de mots communs, d'abréviations et de règles syntaxiques. L'utilisateur peut choisir d'avoir des résultats plus précis sur les annotations *certain* et *impossible* mais obtiendra en contrepartie une plus grande quantité d'annotations *possible*. L'utilisateur peut ensuite classer les abréviations restées *possible* en *certain* et *impossible*.

③ Application d'un **apprentissage** par arbre de décision sur l'extrait annoté pour obtenir les instances mal classées par l'apprentissage qui correspondent presque toujours à des incohérences d'annotation. L'utilisateur peut corriger la classe des abréviations, ajouter ou retirer des descripteurs et recommencer cette étape pour améliorer l'apprentissage.

④ Idem que l'étape 2 mais avec les mots capitalisés qui suivent un point.

⑤ Idem que l'étape 3 mais avec les mots capitalisés qui suivent un point.

⑥ **Traitement récursif** des abréviations et mots capitalisés restés annotés *possible*. L'idée utilisée ici est que l'on peut définir la certitude ou l'impossibilité d'un mot toujours capitalisé suivant un point en fonction de la certitude ou de l'impossibilité de l'abréviation qui précède ce point. Et vice versa.

⑦ Application automatique des étapes précédentes sur le corpus complet avec les choix de l'utilisateur ainsi que les arbres de décision appris. Le résultat est un corpus segmenté sur les fins de phrases ainsi que des listes d'abréviations et de mots capitalisés nouveaux.

Nous définissons l'**apprentissage guidé par l'utilisateur** comme un apprentissage auquel l'utilisateur prend part, au milieu de la procédure, lorsque les informations dont dispose le programme seul ne suffisent pas à prendre une décision.

Le fonctionnement de l'apprentissage est rendu explicite à l'utilisateur lorsqu'il est nécessaire de lui poser une question. Ainsi, l'utilisateur peut influencer le déroulement de l'apprentissage et comprendre pourquoi l'apprentissage ne fonctionne pas.

Cependant, il est important de ne proposer qu'un nombre très limité de cas à traiter, en l'occurrence les plus ambigus. Ces cas sont souvent déterminants pour éviter les contresens dans le traitement du langage naturel. Par exemple, le cas d'un mot capitalisé qui commence presque toujours une phrase tel *"Moreover"* qui signifie *"De plus"* ou d'une abréviation qui termine presque toujours une fin de phrase tel *"etc"*.

Lors de l'apprentissage aux étapes 3 et 5 sur l'extrait du corpus annoté, l'utilisateur peut visualiser les erreurs d'apprentissage et connaître leur origine. Ces erreurs d'apprentissage sont des incohérences dans l'annotation. En pratique, elles correspondent à des erreurs d'annotation. Ceci permet à l'utilisateur de modifier en conséquence les listes de mots, les règles syntaxiques, la précision d'application des statistiques et enfin les descripteurs utilisés pour l'apprentissage.

Les instances qui ne peuvent pas être classées à l'aide des statistiques ou de l'apprentissage sont celles qui sont les plus ambiguës et donc les plus importantes à montrer à l'utilisateur dans le but d'éviter des contresens dangereux dans les étapes suivantes de l'extraction des connaissances. C'est le but de l'apprentissage guidé par l'utilisateur.

À noter que nous mettons à disposition notre outil[1] développé dans le cadre de ces expérimentations sous forme d'un module avec interface graphique pour l'environnement d'ingénierie du langage naturel Gate[2] qui contient déjà un grand nombre de modules.

Summary

This program carry out a sentence segmentation of a text. Contrary to classic procedures, we don't use preliminary annotations and we make the most of a user-guided learning.

[1] Segmenteur en phrases, Java, logiciel libre, http ://www.lri.fr/∼heitz/

[2] Gate, General Architecture for Text Engineering, Java, logiciel libre, http ://gate.ac.uk/