Gregory Grefenstette, Pasi Tapanainen, 1994

# ‘What is a word, What is a sentence? Problems of Tokenization’

Karolina Suchecka, M2 LTTAC, 06/03/2019

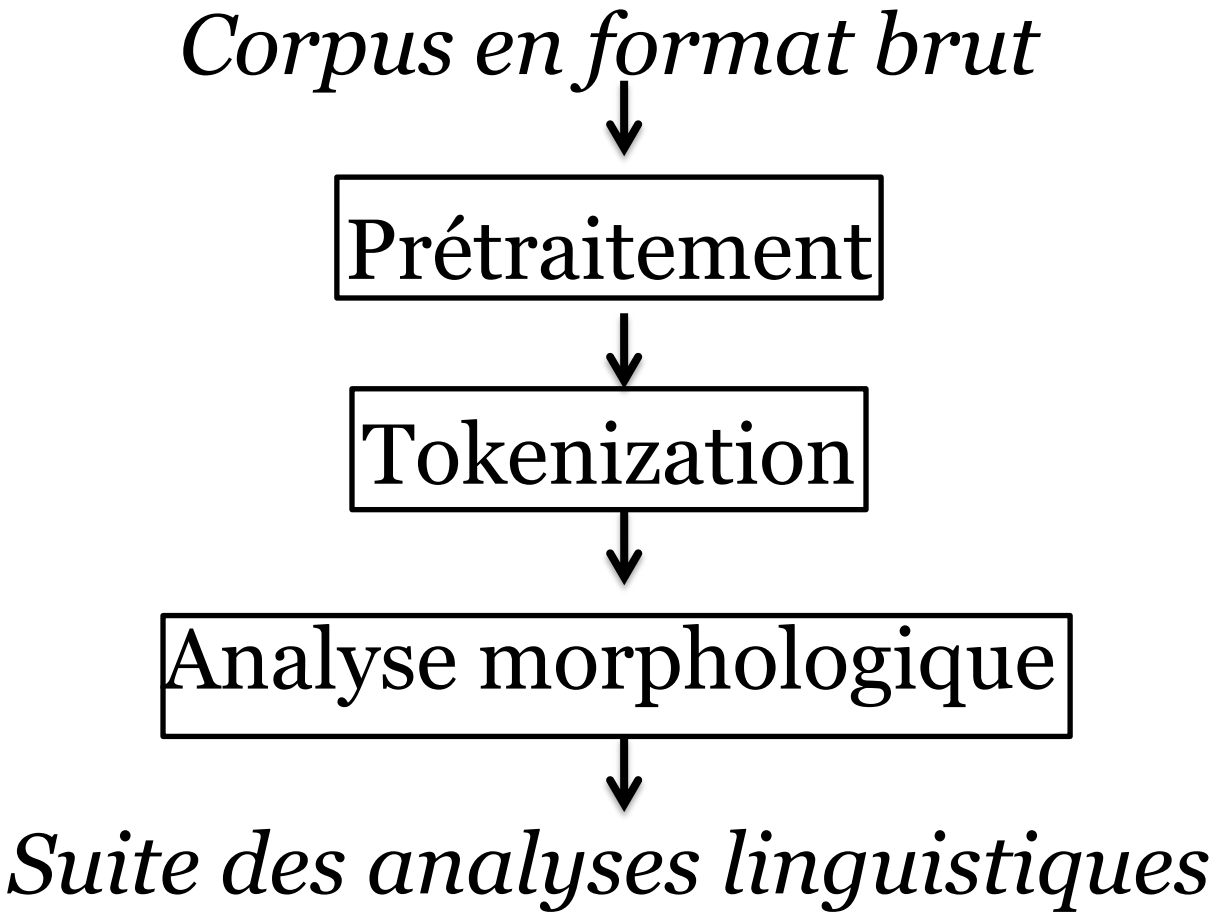
**Objectif de l'article :** détailler des choix qui doivent être faits afin d'optimiser l'étape de la tokenization du corpus.

**Tokenization:** l'action d'isoler les unités rassemblant aux mots du texte. Elle peut résulter en deux types de tokens :

- Structures reconnaissables, comme la ponctuation, les nombres et les dates.
- Structures qui vont ensuite faire objet d'une analyse morphologique (= mots).

→ La tokenization est souvent considérée comme une étape préliminaire de moindre importance. En réalité, elle pose des problèmes qui ne devraient pas être négligés si l'on veut optimiser l'analyse linguistique du corpus.

**Schéma des transformations qui précèdent l'analyse linguistique :**



# Plan de l'article :

1. Nettoyage du texte lors de la phase du prétraitement.
2. Optimisation de la tokenization des abréviations.
3. Quelques remarques concernant l'analyse morphologique.

***Corpus de référence :*** Brown Corpus (Francis and Kucera, 1982) d'un million des mots dont la tokenization a été corrigée manuellement.

# Prétraitement : suppression de balises

- Remarque : l'article considère les corpus nativement numériques plutôt que ceux obtenus à partir de la numérisation des scans (océrisation) → nécessite un nettoyage spécifique (espaces, balises de mise en forme, caractères spéciaux), mais qui reste moins important que pour les OCR.
- Proposition d'une méthode permettant d'éliminer le code SGML (Standard Generalized Markup Language, un langage de description des balises) à l'aide d'un script Flex → remplacement des caractères à l'aide des expressions régulières :

  `"<"[^\n<>]+">"` : supprime les balises et leur contenu (sauf si celles-ci contiennent des sauts de ligne ou des chevrons)

# Prétraitement : reconstitution des mots séparés par la césure

- Comme la césure résulte uniquement de la mise en page, elle doit obligatoirement être supprimée afin de pouvoir reconstituer les mots et de les tokenizer proprement.
- Proposition de nettoyer les césures avec les expressions régulières:

  `[a-z]-[\t]*\n[\t]*` : reconnaît une chaîne des caractères suivie d'un tiret, éventuellement d'une ou des tabulations, puis d'un saut de ligne et des autres tabulations optionnelles. Elle retient ensuite uniquement les signes alphanumériques afin de reconstituer le mot.

**Problème :** le tiret à la fin de ligne n'est pas obligatoirement une césure, mais peut faire partie d'un mot composé. 4,9 % des mots n'ont pas été reconstitués correctement → proposition de les corriger dans une étape ultérieure ou de les considérer comme bruit.

# Rôle de la tokenization

- Après la phase de prétraitement, on obtient une série de chaînes de caractères qui sera considérée comme texte et dont les éléments auront une classe syntaxique précise, p. e. *chien* pourra être étiqueté comme « NOM SINGULIER ».
- Pour que l'étiquetage soit possible, le texte doit être divisé en unités de traitement qui appartiendront à une catégorie spécifique.
- L'approche considérée dans l'article : ces unités de traitement sont des phrases.

# Qu'est-ce qu'une phrase ?

- Une phrase finit avec un signe de ponctuation. Certains de ces signes sont univalents, comme le point d'interrogation ou le point d'exclamation, mais d'autres, et notamment le point, sont ambigus :
  - Point final ?
  - Abréviation ?
  - Les deux ?
  - Partie d'une structure alphanumérique, comme une date (02.02.18), une chiffre (123,456.78) ou une numérotation (T-1-AB.1.2.) ?
- Ce problème est loin d'être insignifiant : une phrase sur 14 du corpus contient un point qui n'est pas final.
- Selon les auteurs, une partie de ces problèmes peut être résolue grâce aux expressions régulières.

# Traitement des chiffres

- Problème de l'usage : 123,456.78 en anglais sera traduit par 123 456,78 en français.
  - Regex pour l'anglais : `(0-9)+[,])*[0-9]([.][0-9]+)?`
  - Regex pour le français : `(0-9)+[°])*[0-9]([,][0-9]+)?`

    L'utilisation de ces expressions régulières risque de générer des erreurs, mais l'occurrence de ce type des chiffres n'est pas fréquente.
- Proposition d'intégrer trois regex au tokenizateur pour permettre la reconnaissance des entités numériques :
  - Dates : `[0-9]+(\/[0-9]+)+`
  - Pourcentage : `([+\-]?[0-9]+()?[0-9]*%]`
  - Numéros, dollars : `[\[0-9]+,?)?+(\.[0-9]+|[0-9])*`

    La reconnaissance de ces entités permet d'écarter une partie des ambiguïtés liées aux caractères spéciaux, puisqu'ils seront désormais inclus dans le token.

# Traitement des abréviations

- Brown corpus : 4819 abréviations non-uniques et 323 uniques. Elles contiennent uniquement des lettres et finissent par un point. Dans ce cas, si on considère que chaque point est final, on a raison uniquement dans 90 % des cas.
- Une série d'expérimentations pour améliorer ce résultat : reconnaissance de la structure des abréviations.

# Première approche : regex uniquement

- Trois types d'abréviations :
  - Lettre + point → *A., B., C.* → `[A-Za-z]\.`
  - Lettre(+point+lettre)+ → *i.e., U.S, m.p.h.* →

    `[A-Za-z]\.([A-Za-z0-9]\.)+`
  - Lettres + point → *Mr., St., Assn.* →

    `[A-Z][bcdfghj-np-tvxz]+\.`
    - Résultat : 3876 sur 3939 abréviations reconnues correctement. 63 erreurs où il s'agit d'un point final et 103 ambiguïtés où l'abréviation est également un point final.

# Deuxième approche : regex et filtrage dans le corpus

- Le corpus lui-même est utilisé pour identifier les abréviations : on considère comme abréviation (i) chaque suite de lettres qui finit par un point et est suivie d'un virgule, d'une lettre minuscule ou d'une chiffre et (ii) chaque chaîne qui commence par une majuscule et finit par un point.
  - Cette approche permet de reconnaître 138 de 323 abréviations uniques du corpus, mais introduit 54 reconnaissances fautives (*chili, continous, every, everyone,* etc.) quand les mots finissent une phrase alors que la phrase suivante commence par une chiffre.
- En utilisant le corpus pour filtrer, les résultats s'améliorent : 986 des phrases ne sont toujours pas reconnues, mais le taux de reconnaissance atteint quand même 97,9 %.

# Troisième approche : intégration d'un lexique dépourvu des abréviations

- Tous les mots suivis d'un point sont traités par un analyseur morphologique qui décide s'il s'agit d'une abréviation ou d'un point final.
- Reconnaissance en 6 étapes :

  1. Si le mot est suivi d'une minuscule, d'un virgule ou d'un point-virgule, c'est une abréviation.
  2. Si le mot est en minuscules et que le même mot est connu du lexique, ce n'est pas une abréviation. Sinon, c'est une abréviation.
  3. Si le mot commence par une majuscule et apparaît ailleurs dans le corpus comme une abréviation connue, c'est également une abréviation.
  4. S'il commence par une majuscule et apparaît autre part dans le corpus suivi d'un espace blanc, ce n'est pas une abréviation (probablement un nom propre).
  5. Si le mot commence par une majuscule et apparaît seulement une fois ou deux, il vaut mieux le considérer comme un mot finissant la phrase.
  6. Sinon, c'est une abréviation.

- Résultat : 99,7 % des phrases segmentées correctement.

# Quatrième approche : intégration de quelques abréviations connues au lexique

- Ajout des abréviations fréquentes au lexique précédent :
  - Titres (*Mr., Mrs., Dr., Sen.*)
  - Mois (*Jan., Feb., Mar.*)
  - États (*Ala., Calif., Penna.*)
  - Autres (*etc., fig., no., Co., Ltd., Corp.*)
- Trois conditions :
  1. Si le mot est suivi d'une minuscule, d'un virgule ou d'un point-virgule, c'est une abréviation.
  2. Si le mot est connu du lexique en tant qu'abréviation, c'est une abréviation.
  3. Sinon, ce n'est pas une abréviation.
- Résultat : seulement 53 erreurs sur 51 240 candidats (contre 155 pour l'approche précédente).

# Autres travaux sur la reconnaissance des limites de la phrase

- Palmer and Hearst (1994) : utilisent les réseaux neuronaux → 98,5 % de reconnaissances correctes après seulement une minute d'entraînement. L'approche ne prend pas en compte les majuscules, elle peut donc être adaptée à d'autres langues, comme l'Allemand, ou aux textes écrits entièrement en majuscules.
- Avant : Riley, 1989 et Müller *et al*, 1980.

# Conclusion

- La tokenization doit être considérée comme une série de filtres modulaires par lesquels le texte passe progressivement.
- Dans la phase du prétraitement, il est important de reconstituer les mots coupés par la césure et d'éliminer le balisage.
- L'analyse de la structure des chaînes de caractère permet de résoudre certaines ambiguïtés avant la confrontation du corpus aux lexiques.

# Merci de votre attention