Révélation des proximités de mots dans un texte littéraire

Introduction

La langue française, plus que toute autre et surtout plus que l'anglais, ne supporte pas les répétitions (de mot, canonique ou non). Répéter un mot à trop court intervalle est lourd et cette lourdeur doit être corrigée (un « intervalle » est la distance entre deux occurrences lémmatiquement identiques, comme « viendraient » et « venu », dont le lèmme — ou « canon » — est « venir »).

Avant de pouvoir être *corrigées*, ces proximités fautives doivent être *signalées* avec le plus de pertinence possible. C'est l'objet de l'application **Proximity**.

Il est nécessaire de trouver une formule susceptible de permettre de signaler les proximités avec le plus de justesse possible.

Ah si Flaubert avait raison !

Gustave Flaubert a établi qu'un même mot (ou lèmme) ne devait pas se répéter avant qu'une page ne soit passée. En dessous de cette **Distance Minimale** (ici de 1500 caractères ou de 250 mots [1]), deux mots seraient « en proximité » et le texte deviendrait lourd.

> [1] *On préfèrera compter en nombre de caractères, le nombre de mots pouvant être bien trop aléatoire.*

Malheureusement, à l'usage, il devient vite évident que cette formule est quelque peu expéditive et ne tient aucun compte des types de mots ou des aspects subjectifs de la lecture, non plus que de la longueur du texte.

Il s'agit donc ici de **déterminer la formule réelle pour calculer cette DM** (distance minimale) de sécurité propre à chaque mot et la rendre visuellement dans le texte affiché.

Nota Bene terminologique

Dans la suite, lorsque nous parlerons de « mot », il faudra entendre « lèmme », sauf contre-indication. Quand nous parlerons de « deux mêmes mots en proximité », il peut s'agir de « venu » et « viendrai » qui ont le même lèmme (canon) « venir », même si ces deux mots ne sont pas strictement identiques.

Résultat visuel à obtenir

Le but de cette formule et de celles qui seront nécessaires à son fonctionnement doit permettre de produire un texte où les mots *en proximité* seront « taggués », c'est-à-dire mis en exergue en couleur.

Cette couleur devra varier en fonction de la **dangerosité de la proximité**. La « dangerosité » de proximité entre deux mots est proportionnelle à sa proximité. Plus la proximité (`Px`) est grande (ie plus la *Distance observée* — `Do` — est faible) et plus la dangerosité est forte => la couleur se rapproche du rouge.

> *Rapport entre proximité et distance observée :*
>
> ```
> Px = 1 / Do
> ```

Quatre couleurs seront adoptées selon la dangerosité de la proximité :

- rouge lorsque `Do < DM / 4` (formule provisoire ne tenant pas compte des autres facteurs),
- orange lorsque `Do < DM / 2` (idem),
- green lorsque `Do <= 3 * DM / 4` (idem),
- bleu lorsque `Do > 3 * DM / 4` (idem).

> *Noter qu'un même mot peut être en proximité double, avec un mot avant ET un mot après. Ces deux proximités peuvent être différentes (une rouge et une bleu).*

Le mot devra pouvoir être survolé par la souris, et faire apparaitre une info-bulle qui contiendra des informations objectives (distance observée entre les deux mots, distance minimale prise en référence, etc.).

Distance Minimale commune ou particulière

Tous les mots possèdent une DM par défaut de **1500 caractères** (DM « flaubertienne »).

Un dictionnaire définit cependant des DM propres à certains mots particuliers de la langue. Ces mots sont fixes et ne dépendent en aucun compte du texte. Ils sont assujettis à des distances minimales absolues. À commencer par les mots qui supportent une proximité plus petite sans apporter de lourdeur au texte. Cette table, que nous appellerons « Dictionnaire des Distances Minimales », est définie ainsi

```
const DICO_DIST_MINI = {
    'pour': 300
  , 'mais': 250
  ,
}

const DIST_MINI_DEFAULT = 1500
```

On peut de cette manière définir toute DM absolue d'un mot (ie quel que soit le texte) par :

```
Dm('pour') = DICO_DIST_MINI['pour'] || DIST_MINI_DEFAULT
```

Variance appliquée à la DM

Cependant, ces formules simples ne tiennent pas compte de la réalité d'un texte. Elles doivent être *adaptées* suivant le contexte, en sachant que :

1. plus un mot sera rare dans un texte, et plus sa DM devra augmenter selon un facteur `Kdm` (dont la valeur — qui doit être expérimentée — sera définie en constante)
2. mais à un seuil d'occurrences, la DM deviendra nulle (aucune problème de proximité signalé),
3. la différence levenstein entre deux mots influe également sur la DM.

1. influence de la rareté du mot

> *La rareté d'un mot est propre à un texte, pas à la langue. Ainsi, le mot « maison », fréquent dans la langue, pourra devenir un mot extrêmement « rare » dans un texte de 10 000 mots qui ne le contiendrait qu'une seule fois.*

La *rareté* d'un mot (`Ra`) est inversement proportionnel à la fréquence (`Fr`) du mot. Si un mot à une fréquence de 10%, ça rareté sera de `100/10`.

```
Soit un texte de 10 000
Qui contient 200 fois le mot "maison"
=>
La fréquence du mot est de 2 %
Sa rareté est de 50
```

2. Influence du nombre d'occurrences du mot

Si le nombre d'occurrences du mot est inférieure à 4, sa DM passe à 0 (zéro) => aucune proximité possible, même si les mots sont côte à côte.

```
if occurrence(mot) < 4 then DM = 0
```

3. Influence de la valeur de levenstein

La valeur — ou distance — de levenstein entre les deux mots doit également influencer la DM. Plus les deux mots sont distants au niveau levenstein (ie « différents »), plus la distance peut être réduite.

> *Pour rappel, deux mots identiques ont une distance de levenstein de 0 (zéro). Tandis que « niche » et « chien » ont une distance de 5 (5 opérations nécessaires pour passer de l'un à l'autre).*

Exemple : quand « venir » et « venir » sont assujettis à la DM par défaut de 1500 caractères, les mots « venu » et « viendrai » doivent supporter une DM plus petite (par exemple 1300 caractères).

Si l'on considère le facteur `Klev` (qui sera calculé par expérimentation et défini en constante), on peut considérer que :

```
Dr = DM / (distanceLevenstein(motA,motB) + 1) * Klev
```

(`Dr` : Distance minimale rectifiée, `DM` : Distance minimale pour le mot étudiée)

Variance de la dangerosité

La *dangerosité* définie tout à l'heure varie aussi selon la fréquence et la différence de levenstein suivant les règles suivantes :

- plus la fréquence du mot est grande et plus son indice de dangerosité augmente.

- plus la distance de levenstein s'éloigne de 0 et plus son indice de dangerosité diminue.

Formule et méthode finale

Le résultat de cette réflexion est une méthode qui doit avoir en entrée un texte quelconque de longueur quelconque et qui fournit en sortie une liste des proximités trouvées avec les informations suivantes tenant compte de toutes les conditions énumérées ici :

```
[ // liste des proximités

  // Première proximité
  {
      proximiteId: 1
    , motA : {instance du premier mot en proximité}
    , motB : {instance du second mot en proximité}
    , couleur: <couleur en fonction de dangerosité>
    , Do: <distance observée>
    , Dr: <distance minimale rectifiée>
    , DM: <distance minimale absolue suivant le mot> (pour rappel)
  }

  // Deuxième proximité
 ,{
      proximiteId: 2
    , motA : {instance premier mot}
    , motB : {instance second mot}
    , couleur: <couleur en fonction de dangerosité>
    , Do: <distance observée>
    , ...
  }

  // etc. pour chaque proximité
]
```

Cette liste doit permettre de « tagguer » les mots du texte et de fournir une info-bulle qui indiquera les valeurs exactes de proximité du mot lorsque l'on survolera le mot à la souris.

Enjoy! 😁 😅 😂 😤 😡 🤕 🤠