Projet TranspArt

A faire en Java, et en binôme

1 Objectif

Les résultats d'un article scientifique A ont été présentés à une conférence lors d'un exposé que nous allons appeler T [1] Nous supposons que l'article A a été découpé en pages qui vous sont fournies (un fichier Ai.txt pour chaque page a_i, en supposant que le nombre n de pages est connu). Nous supposons également que les transparents de la présentation T ont été transcrits sous forme texte et vous sont fournis (un fichier Tj.txt pour chaque transparent t_j, en supposant que le nombre m de transparents est connu). Le but est d'associer les transparents aux pages de l'article auxquelles ils correspondent (des cas particuliers, où un transparent ne correspond à aucune page, ou une page n'est décrite par aucun transparent sont envisageables).

2 Travail à faire

Plusieurs variantes sont envisageables, comme suit :

- **One-Vs-ZeroOne :** associe tout transparent à exactement une page ; une page ne peut pas être associée à plusieurs transparents.
- **ZeroOne-Vs-ZeroOne :** associe un maximum de transparents à exactement une page chacun; une page ne peut pas être associée à plusieurs transparents.
- **ZeroOne-Vs-Many :** associe un maximum de transparents à exactement une page chacun, mais une page peut être associée à plusieurs transparents.

Pour chaque variante, il est nécessaire de :

1. créer le dictionnaire *Dico* des mots significatifs de l'article, c'est-à-dire des mots qui se trouvent dans l'article et qui sont propres au domaine de recherche. Pour cela, un fichier *DicoDeMonDomaine.txt* vous est donné, contenant une suite de mots séparés par des espaces (un seul espace entre deux mots consécutifs de la même ligne) ou des fins de ligne.
2. définir une ou plusieurs notions de similarité entre n'importe quelle page a de l'article et n'importe quel transparent t de la présentation.
3. définir une condition nécessaire et suffisantes pour qu'une page a et un transparent t soient considérés assez proches pour que l'association entre les deux soit envisageable.
4. définir le problème correspondant à la variante comme un problème de graphes
5. écrire l'algorithme correspondant, et le programmer en Java

Il est demandé d'implémenter toutes les variantes énoncées de la manière la plus efficace possible (du point de vue algorithmique), en suivant les points ci-dessus (évidemment, certains d'entre eux sont communs à toutes les variantes, et ne seront réalisés qu'une fois).

Votre implémentation utilisera, entre autres, des algorithmes classiques de théorie des graphes. Dans ce cas, elle doit suivre de près la présentation des algorithmes faite dans le cours. Chaque classe doit être dûment commentée, ligne par ligne, en français, en faisant référence aux lignes des algorithmes du cours. La doc Java doit permetre de savoir en un coup d'oeil dans quelle méthode est effectuée chaque opération importante des algorithmes vus en cours. Les commentaires à l'intérieur des fichiers sources doivent permettre de comprendre comment chaque opération est implémentée, ligne par ligne encore.

[1] T comme talk

Pour les algorithmes du cours comme pour ceux que vous écrirez vous-mêmes, un lecteur ne doit jamais être perdu, il doit comprendre ce que vous faites et comment vous le faites. La notation prendra fortement en compte cet aspect de votre programme.

3 Rendu de TP

Les programmes doivent fonctionner parfaitement sur les machines du CIE, sous Linux. Un rapport d'au maximum 10 pages sera fourni, indiquant (entre autres) :

- les commandes de compilation et exécution en mode console
- les définitions des notions de similarité (plusieurs notions sont à envisager), des conditions nécessaires et suffisantes (plusieurs conditions sont à tester) et des problèmes énoncés (cf. points 2, 3 et 4 ci-dessus).
- les algorithmes utilisés (avec leurs descriptions en pseudo-code, pas en Java)
- les complexités de ces algorithmes et la complexité globale de votre programme
- des jeux de données, comparant notamment les différentes notions de similarité que vous aurez proposées.

Les fichiers du programme, ainsi que les jeux de données seront mis dans une archive de nom Nom1Nom2.**zip** (où Nom1 et Nom2 sont les noms des deux étudiants du binôme). L'archive sera déposée sur Madoc, dans l'espace dédié à votre groupe, au plus tard le jour de votre dernier TP, à 23h59 (plus précisément, le 2 décembre 2015 pour le groupe de TP1, et le 4 décembre 2015 pour le groupe de TP2).

4 Important

- La toute dernière séance d'1h20 est dédiée à une démo de vos programmes (10min par programme).
- Aucun dépôt en retard ne sera autorisé par Madoc.