Documentation Ancor2

31/05/2018

Table des matières

Informations générales

Dépôts git

Alexis https ://bitbucket.org/Slayug/ancortodemocrat.git

Augustin https ://gitlab.com/augustinvoima/ancor2.git

Données des expériences

https ://drive.google.com/drive/u/0/folders/14mp2w690RVP6yh3H2dmxkYIeB-LMa2E2

Documentation

https ://docs.google.com/document/d/1vJP4o1z3eCfZSxo88vZTlS0OpT54B1Zu-Nfq4naSlWc/edit ?usp=sharing

Dépendances

git submodule — reference-coreference-scorers

maven javax.xml.bind Parser XML

log4j Sortie text

nz.ac.waikato.cms.weka Classification

org.annolab.tt4j TreeTager : gestion des annotations *part-of-speech*

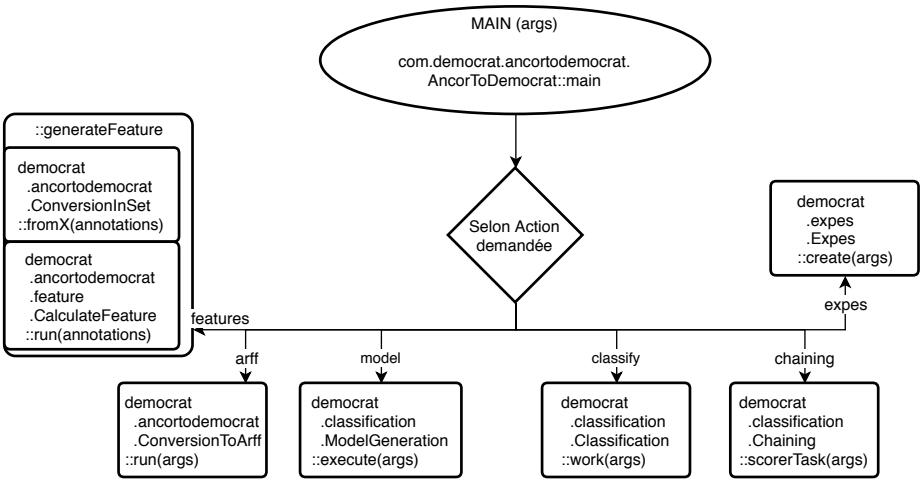
commons-cli Gestion des arguments en ligne de commande (Remplacement de l'ancienne gestion des arguments à faire)

commons-io Gestion des Fichiers

org.odftoolkit Gestion des fichiers OpenDocument (sortie des résultats pour les expériences, en cours de dvlp)

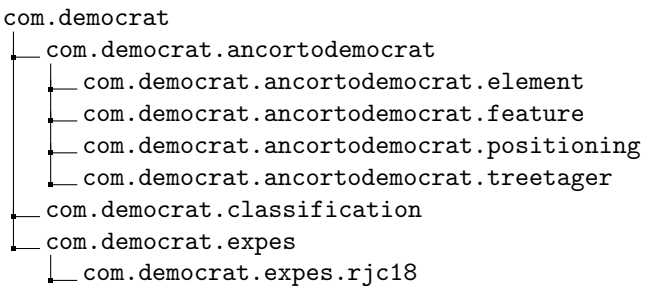
Processus

Appel des fonctionnalités



Architecture du projet

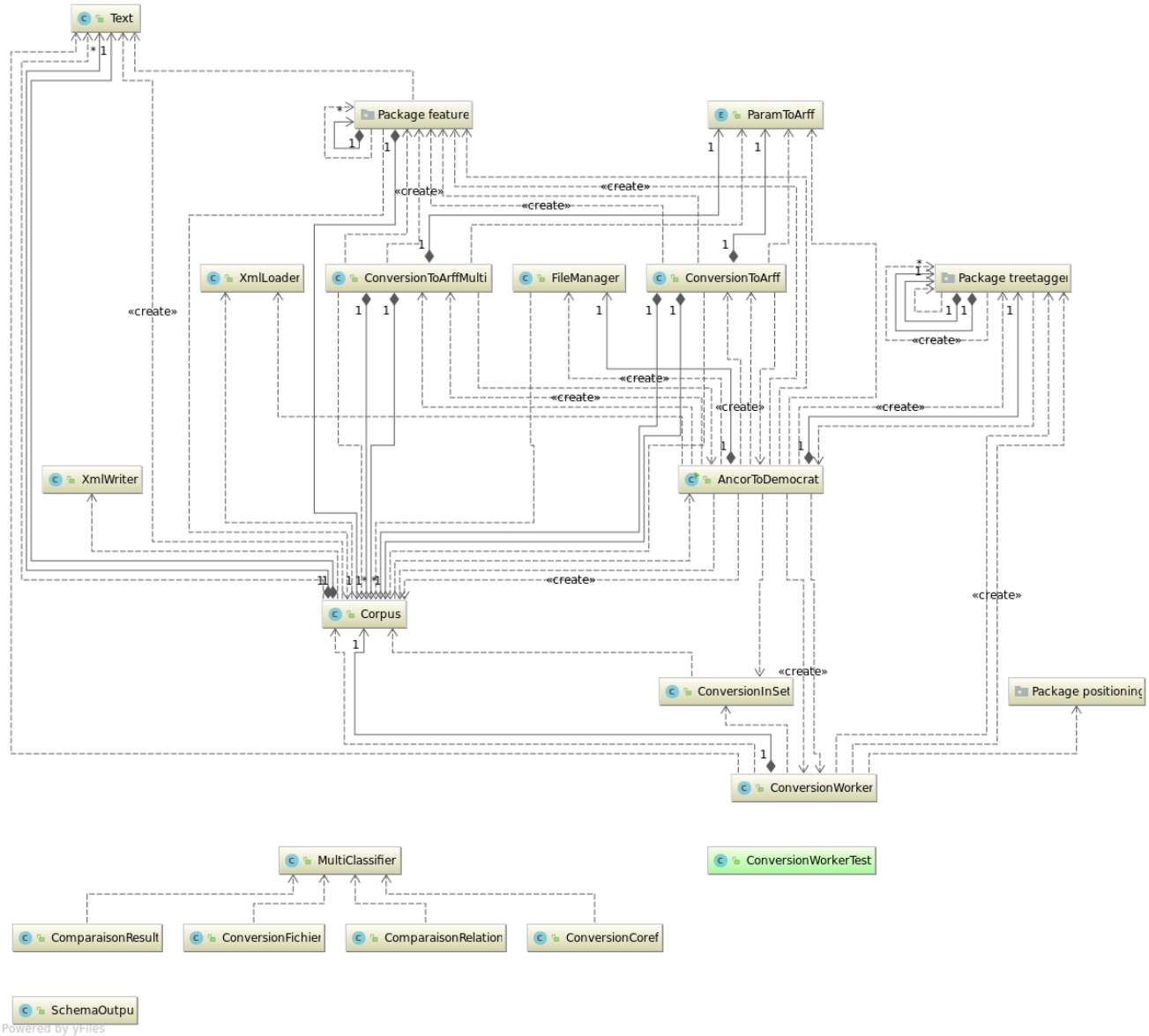
Packages du projet



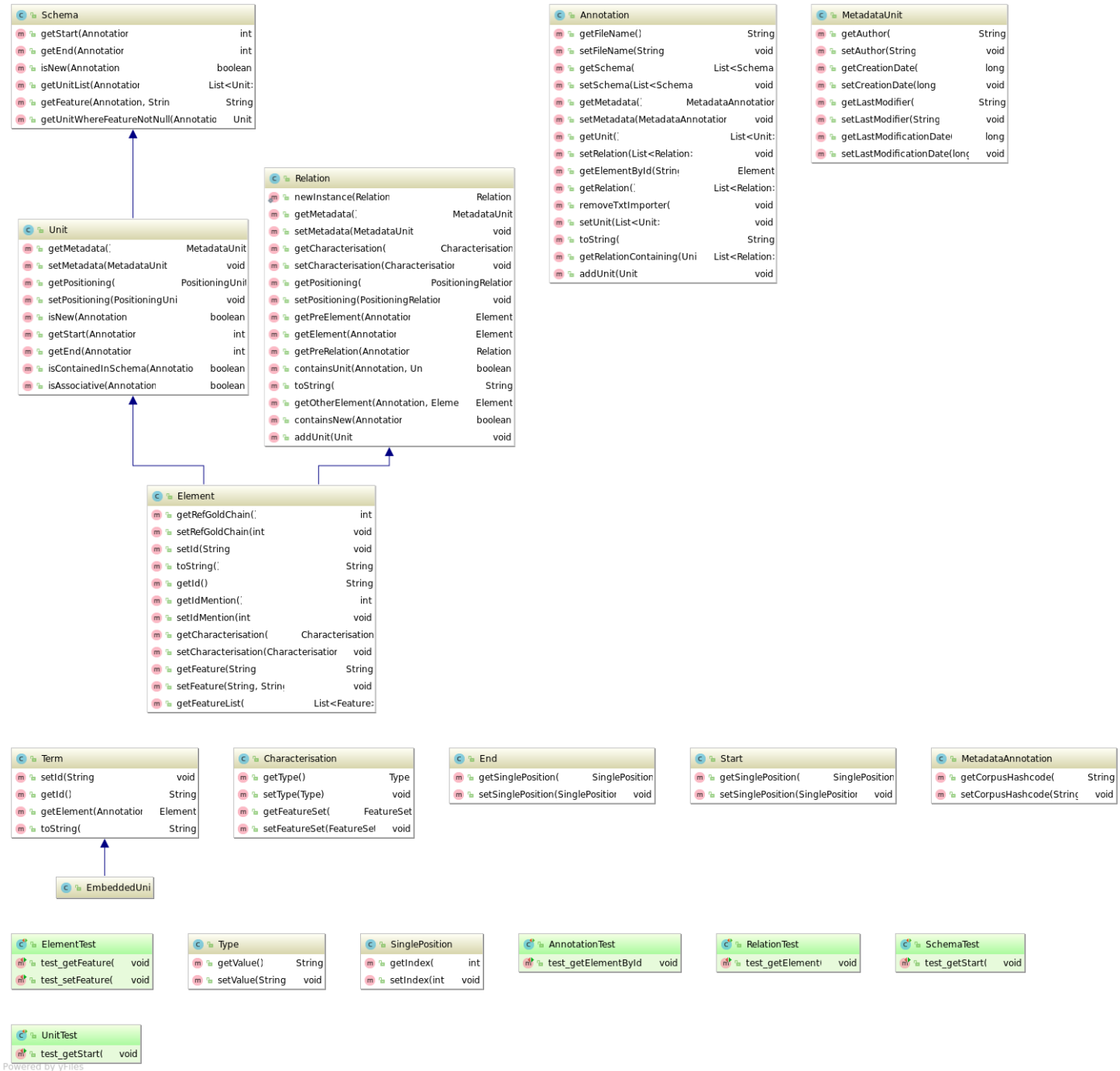
com.democrat.ancortodemocrat

sous-packages

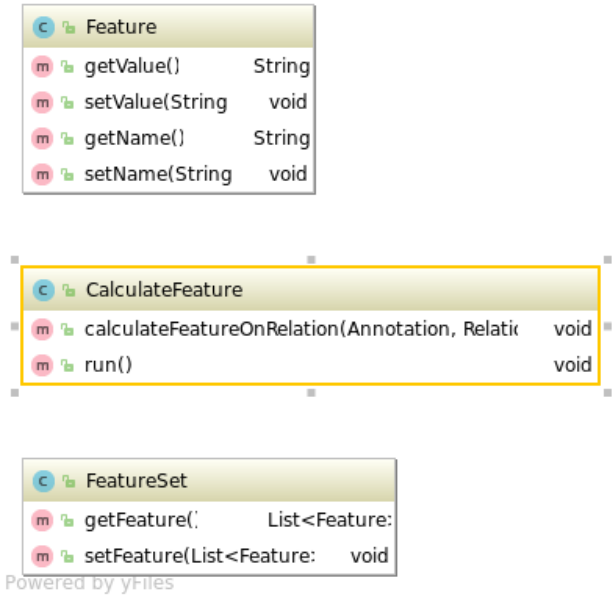
```
com.democrat.ancortodemocrat
├── com.democrat.ancortodemocrat.element
├── com.democrat.ancortodemocrat.feature
├── com.democrat.ancortodemocrat.positioning
└── com.democrat.ancortodemocrat.treetager
```



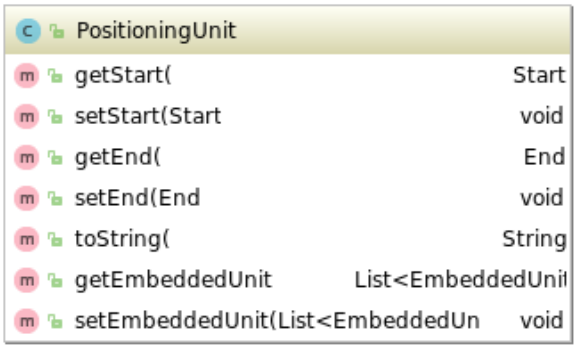
com.democrat.ancortodemocrat.element

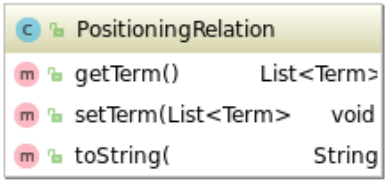


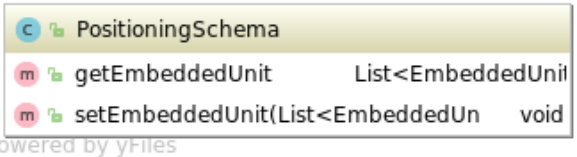
com.democrat.ancortodemocrat.feature



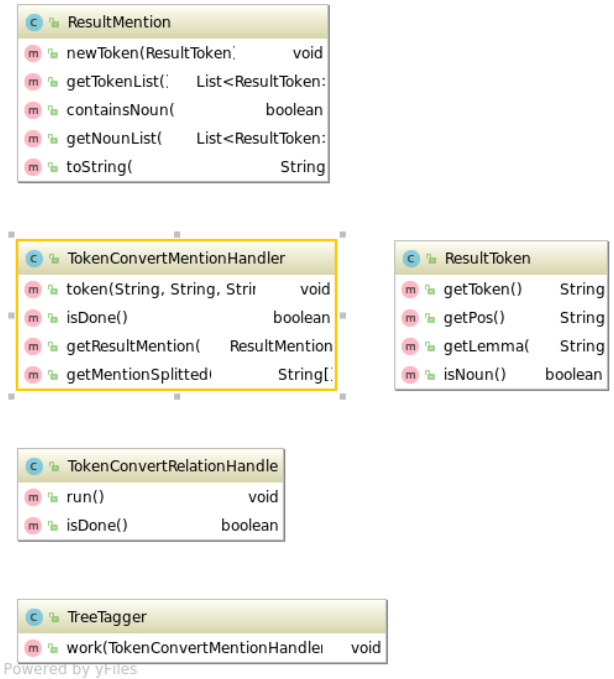
com.democrat.ancortodemocrat.positioning



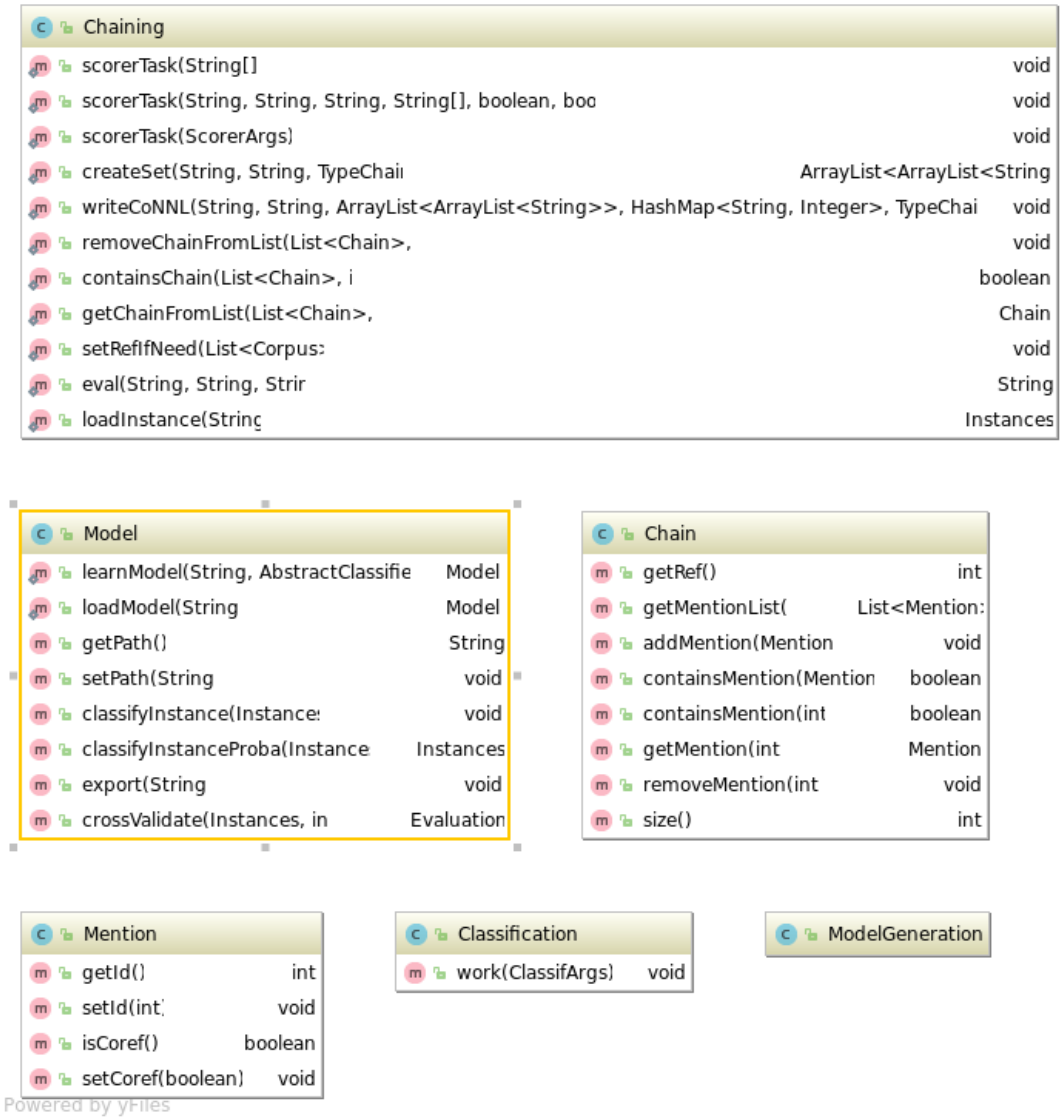




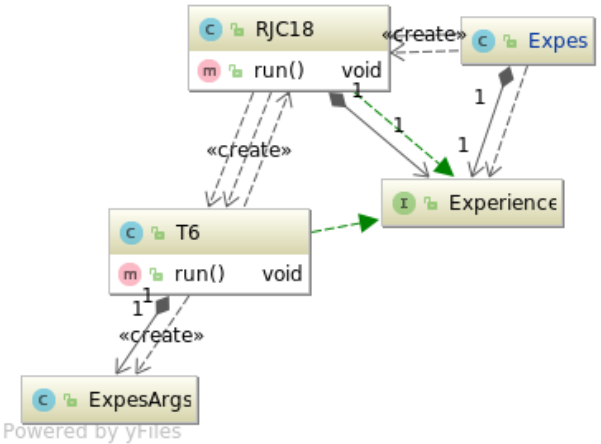
com.democrat.ancortodemocrat.treetagger



com.democrat.classification



com.democrat.expes



Entrées - Sorties

Calcul des features

Entrée : Corpus source : chemin/vers/Ancor

```
chemin/vers/corpus/Ancor
├── aa_fichiers
│   └── *.aa
└── ac_fichiers
    └── *.ac
```

Sortie : Corpus avec nouveaux features calculés : chemin/vers/features

```
chemin/vers/features
├── aa_fichiers
│   └── *.aa
└── ac_fichiers
    └── *.ac
```

Etapes du calcul des features

```
com.democrat.AncorToDemocrat::generateFeature(corpus, output){
    Pour chaque annotation du corpus:
        ConversionInSet.toSetFrom{Chain/FirstMention}(annotation) // Genère le feature REF
    CalculateFeature::new(corpus,output){
        Pour chaque annotation du corpus:
        calculateNewFeature(annotation)
        calculateFeatureOnRelation(annotation)
        corpus.export(output)
    }
}
```

Génération des arff

Entrée : Corpus contenant tous les features : chemin/vers/features

Sortie : nomfichier

- nomfichier(date).arff →Contient toutes les instances
- nomfichier(date).idff →Contient les identifiants de chaque instance et des mentions qui les composent

Etapes génération Arff

```
com.democrat.ancortodemocrat.ConversionToArff::work(){
   sortInstance(){
       Pour chaque corpus:
         Pour chaque annotation du corpus:
            Pour chaque relation de l'annotation:
               positiveRelationSelected.put(relation)
               generateNegativeRelation(corpus,annotation, relation){
                  Pour chaque relation négative à générer:
                     newrelation = deux mentions au hasard qui ont
                                   un champ REF différent
                     negativeRelationSelected.put(newrelation)
             }
   }
   selectInstance(){
       restriction nb_pos et nb_neg en fonction
          du nombre d'instances dans le corpus
          et en conservant le ratio
       mélange aléatoire de la liste des instances positives // à modifier
       mélange aléatoire de la liste des instances négatives
   }
   writeInstance()
}
```

Génération du model

Entrée : Corpus d'apprentissage et de test
- train.arff
- test.arff

Sortie : Model

Etapes génération du model

```
com.democrat.classification.ModelGeneration::execute( classifierclass, args ){
    AbstractClassifier.runClassifier(classifierclass.newInstance(),args);
}
```

Classification

Entrée :
- Model
- Fichier arff à classifier

Sortie : nomfichier.arff →Instances classifiées

Etapes de la classification

```
com.democrat.classification.Classification::work(args){
    instances = ArffLoader::getDataSet();
    model = Model.loadModel(args.model);
    instances = model.classifyInstancesProba(instances);
    ArffSaver::setInstances(instances);
}
```

Chaînage

Entrée : corpus gold et system
- Gold.arff
- System.arff

Sortie :
- *nomfichier*_conll_to_ancor.csv →Table de correspondance des identifiants de mentions conll / Ancor
- *nomfichier*_GOLD.conll →Chaines Gold au format conll
- *nomfichier*_SYSTEM.conll →Chaines System au format conll

Facultatif :
- *nomfichier*_LOE_GOLD.csv →List of Edges Gold (ensemble des relations en première mention pour Gephi)
- *nomfichier*_LOE_SYSTEM.csv →List of Edges System (ensemble des relations en première mention pour Gephi)
- *nomfichier*_LOM.csv →List of Mentions (ensemble des mentions (Ancor_ID, Conll_ID, Chain_ID, Num_Antecedents_Before_BestFirst)

Etapes du Chaînage

```
com.democrat.classification.Chaining::scorerTask(args){
    goldChains = createSet(goldArff)
    // idem pour system

    pour chaque mention
        ecrire équivalence id_Ancor / id_Conll dans mention_str_to_int

    writeCoNNL(goldOut, mentions_str_to_int, goldChaine);
    // idem pour system

}

createSet(fichier){
    instances = loadInstances(fichier);

    boucle: Ajout de chaque mention dans un ensemble

    antécédents_possibles = HashMap<mention, HashMap<antécédent, proba> >
    Pour chaque instance coref:
        ajout de (instance.antecedent,proba) pour instance.element
        dans antécédents_possibles

    renvoyer constructChains(antécédents_possibles, mentions){

        chaines = ArrayList<HashMap<String,Integer>> //  liste de chaines
        // chaque mention connaît son nombre d'antécédents possibles avant best-first

        pour chaque mention parmi les clés de antécédents_possibles:
            antécédent = antécédent max par proba dans antécédents_possibles.get(mention);
```

```
            On enlève mention et antécédent du dictionnaire mentions pour ne laisser que
            les singletons

            boucle sur chaine: on place mention et antécédent dans la chaîne qui contient
            l'un des deux

            Si aucune chaîne ne contient mention ou antécédent, on crée une nouvelle
            chaîne qui les contient

            Si mention et antécédent sont placés dans deux chaînes, on les fusionne.

        boucle: on crée et ajoute une chaine par mention restante dans le dictionnaire
        mentions (ce sont les singletons)

        renvoyer chaines;
    }
}
```

Refactoring

Refactoring de la classe principale sous-traiter à des sous-classes principales pour chaque fonctionnalité

Gestion des arguments utiliser la librairie *commons-cli* pour avoir une gestion des arguments plus simple et maintenable, et locale à chaque fonctionnalité (sous-classes principales.

Refactoring des duplicatats de code Présents à beaucoup d'endroits

ConversionToArff* Suppression des classes actuellement inutiles

Multi-Threading Certains traitements sont réalisés dans des threads qui n'exploitent pas forcément le multi-threading.
→Suppression de ces threads ou optimisation ?