Université Paris Ouest - Nanterre La Défense

Master Professionnel Spécialité
Documents Electroniques et Flux d'Informations

Mémoire de recherche

Apprentissage d'un modèle de résolution automatique de la coréférence à partir d'un corpus de français oral

Adèle Désoyer

25 juin 2014

Stage effectué au LATTICE (UMR 8094) et financé par le projet ANR ORFEO
Sous la direction de Frédéric Landragin et Isabelle Tellier

Table des matières

Chapitre 1

Introduction

1.1 Problématique de recherche

Depuis quelques années, l'analyse du discours est un domaine de recherche actif, en traitement automatique des langues et même plus largement en linguistique. Le développement d'applications telles que le résumé automatique, la navigation textuelle ou encore la tâche de question-réponse a effectivement démontré l'importance de considérer des structures intermédiaires entre le texte et la phrase, constituant des unités cohérentes utiles à l'interprétation d'un énoncé. Parmi ces unités, les chaînes de coréférence se définissent comme des ensembles d'expressions d'un texte référant à une même entité externe au discours, et constituent depuis les vingt dernières années un objet d'étude à part entière, au cœur de grandes campagnes d'évaluation telles que MUC (*Message Understanding Conference*), SemEval (*Semantic Evaluation*), CoNLL (*Computationnal Natural Language Learning*) ou ACE (*Automatic Content Extraction*).

Les systèmes de résolution automatique de la référence sont encore aujourd'hui assez rares, et même inexistants s'agissant du français. Mais avec le développement croissant depuis les vingt dernières années de corpus de référence servant la linguistique de corpus, des problématiques telles que celles-ci sont de plus en plus étudiées. L'enjeu de ce travail sera d'apprendre, à partir d'un corpus de français oral, un modèle de résolution automatique de la coréférence, dont les résultats nous permettront de dégager des caractéristiques de construction de la référence en français, et plus spécifiquement des phénomènes propres à l'oral.

1.1.1 Point de départ théorique

La notion d'anaphore décrit une procédure référentielle qui regroupe les phénomènes de renvoi à un antécédent du discours immédiat. Concrètement, il s'agit d'une relation asymétrique entre un antécédent et une expression anaphorique dans laquelle l'anaphorique ne peut être interprété qu'à partir de l'antécédent. Dans un énoncé tel que

a) « **Pierre** est venu. **Il** repartira demain »

le pronom *Il* ne peut être interprété que si l'on accède à l'énoncé précédent pour y repérer *Pierre*. La tâche peut s'avérer plus complexe lorsque plusieurs antécédents sont possibles pour une même anaphore, comme dans

b) « **Pierre** a croisé **Paul**. **Il** l'a tout de suite reconnu »

où il est difficile de déterminer quel référent entre *Pierre* et *Paul* est l'antécédent de *il*.

Bien que le pronom constitue la forme prototypique de l'anaphore, les formes qu'elle peut prendre sont plus diversifiées : ainsi, une anaphore peut être nominale définie (c), démonstrative (d) ou indéfinie(e) ou encore associative (*i.e.* il s'agit d'une forme de reprise pour laquelle l'anaphorique et l'antécédent sont liés sémantiquement sans faire référence à une même entité : elle relève d'un savoir partagé et se manifeste souvent par une relation de type métonymique (f)).

c) « Tu as invité **Pierre**, **le garçon** très charmant. »
d) « Tu as invité **Pierre**, **ce garçon** très charmant. »
e) « Tu as invité **Pierre**, **un garçon** très charmant. »
f) « **Le restaurant** est bon mais **le serveur** est très désagréable. »

Cette notion d'anaphore s'intègre en partie à celle de coréférence. Celle-ci est plus complexe puisqu'elle décrit la relation existant entre plusieurs expressions référant à une même entité. Contrairement à l'anaphore qui distinguait strictement ses deux parties, la relation de coréférence est symétrique (en effet, deux expressions coréférentes x et y peuvent être définies par *x est coréférent de y* aussi bien que par *y est coréférent de x*). Dans (Corblin, 1995), une chaîne de coréférence est définie comme « une suite d'expressions d'un texte entre lesquelles l'interprétation établit une identité de référence » dont les différentes expressions constituent des **maillons**. Cette unité discursive de chaîne de coréférence est souvent réduite à la description d'une suite de relations anaphoriques que l'on peut reconstituer par transitivité. Par exemple, la chaîne de coréférence [*Jean-Pierre Bel*$_1$*, président du Sénat*$_2$*, il*$_3$*, son*$_4$] présente en (g) pourrait se ramener à la suite de paires anaphoriques [[*Jean-Pierre Bel*$_1$, *président du Sénat*$_2$], [*Jean-Pierre Bel*$_1$, *il*$_3$],[*Jean-Pierre Bel*$_1$, *son*$_4$]].

g) « **Jean-Pierre Bel**$_1$, **président du Sénat**$_2$ depuis le 1er octobre 2011, annonce qu'**il**$_3$ quittera **son**$_4$ fauteuil du Palais du Luxembourg. »

1.1.2 Objectifs

L'objectif de ce travail est double : il s'agit principalement de construire un système de détection automatique de la coréférence à base d'apprentissage automatique, en s'inspirant des protocoles de travaux existants. A terme, un tel outil pourrait être utile à des analyses linguistiques approfondies de la problématique que constitue la coréférence, grâce à l'annotation automatique de grands volumes de données, en limitant les biais de l'annotation manuelle (erreurs d'inattention, subjectivité de l'annotateur, ...). Si ce n'est une annotation complète, les fonctionnalités d'un tel système pourraient au moins permettre une pré-annotation de corpus qui diminueraient les coûts financiers et humains d'une annotation manuelle.

D'un point de vue plus linguistique, l'objectif repose sur l'observation et l'analyse des procédés de référence dans le discours en français oral. Le corpus annoté dont nous disposons présente des spécificités qui seront particulièrement étudiées à travers les différents résultats d'expérimentations, ainsi que par les informations quantitatives des données le constituant (comme la longueur moyenne des chaînes de coréférence ou encore les critères discriminants de la séléction d'un antécédent).

Dans ce mémoire, nous commencerons par présenter un historique des systèmes de l'état de l'art répondant à notre problématique. Puis, à partir des méthodes implémentées dans ces systèmes, nous décrirons les données dont nous disposons pour l'apprentissage supervisé avant d'exposer la manière dont elles seront utilisées pour la génération d'un modèle de détection des chaînes de coréférence. Après la description de notre protocole

d'expérimentation et de ses paramètres, nous exposerons nos résultats d'évaluation en les comparant à ceux des systèmes existants.

1.2 Projet ANR Orfeo

Ce travail s'inscrit dans les recherches du projet ANR ORFEO (*Outils et Ressources pour le Français Ecrit et Oral*) porté par le Lattice et coordonné par Jeanne-Marie Debaisieux, depuis février 2013. Ce projet de 36 mois se donne pour objectif de constituer un Corpus d'Etude pour le Français Contemporain (*CECF*) d'environ 15M de mots, issu de la collecte de données écrites et orales représentatives des différents usages et genres du français contemporain, ainsi que de celle de corpus préexistants. Seront ajoutées à ces données différentes couches d'annotations, relevant de différents niveaux d'analyse linguistique (morphologique, syntaxique, discursif et prosodique dans le cas des données orales), afin d'obtenir un corpus richement annoté, exploitable et utile à de futurs travaux linguistiques. Ce sera à terme une plate-forme en ligne qui hébergera l'ensemble de ces données et en permettra l'accès. Un travail d'annotation comme celui-ci ne saurait se satisfaire d'analyses manuelles, tant le corpus est volumineux. Le projet Orfeo consacre donc également une grande partie à l'automatisation des processus, pour proposer aux utilisateurs de la plate-forme des outils de requête et d'analyse automatiques, grâce auxquels ils pourront développer les analyses de leur choix, et des corpus adaptés à leurs besoins. Un dernier aspect du projet, dans lequel s'inscrit ce mémoire, consiste à proposer des études pilotes dans différents domaines afin de rassembler au sein d'un même projet global différents travaux de recherche pour une collaboration inter-disciplinaire sur des données communes.

Chapitre 2

Etat de l'art des systèmes de résolution de la coréférence

La résolution automatique de la coréférence se trouve dans la majorité des systèmes ramenée à un problème de résolution anaphorique. Une chaîne de coréférence est en effet souvent réduite à un ensemble de mentions entretenant deux à deux des relations anaphoriques, et les détecter automatiquement revient à les reformer à partir des résultats de résolution anaphorique. Dans sa thèse (2007), Pascal Denis décrit un algorithme général en sept grandes étapes, que tous les travaux ayant précédé le sien ont suivi pour construire leur système dit *end-to-end*, c'est-à-dire qui part d'un texte brut sans aucune annotation pour y détecter les chaînes de coréférence.

2.1 Schéma général

Sans décrire les méthodes intrinsèques à chaque étape de l'algorithme global, qui sont propres à chacun des systèmes développés, voici une description du traitement d'un document brut D dont on veut retrouver l'ensemble des liens de coréference :

1. Identification des expressions référentielles contenues dans D, que l'on stocke dans la liste M (*i.e.* extraction de toutes les expressions nominales et pronominales dont le contenu est référentiel, en laissant de côté notamment les pronoms explétifs [1].). Un ensemble d'expressions anaphoriques A est initialisé à partir de M, c'est-à-dire que toutes les mentions de M sont pour l'instant considérées comme anaphoriques. Cette première étape peut faire appel à des ressources externes telles que des chunkers ou des extracteurs d'entités nommées, afin d'aiguiller l'extraction.
2. Caractérisation des mentions permettant d'extraire des informations susceptibles d'être pertinentes pour relier une mention à un antécédent (ces informations peuvent concerner les catégories grammaticales, les types d'entités nommées, etc.).
3. Filtrage des anaphoriques : parmi l'ensemble A précédemment généré, on ne conserve que les mentions m dont le fonctionnement est anaphorique.

   Parmi l'ensemble A, pour chaque mention m :

4. Génération des antécédents candidats (par défaut, l'ensemble des candidats C_j contient toutes les mentions précédant linéairement la mention courante C_j).
5. Filtrage de C_j selon un ensemble de contraintes prédéfinies (*e.g.* accord en genre et nombre, même tête nominale ...).

1. un pronom explétif est employé dans un énoncé sans véritable rôle grammatical, comme dans *il pleut* ou *il faut*

6. Attribution de score ou classement par degré de pertinence de chacun des candidats restants, en fonction là encore d'une série de contraintes.
7. Choix de l'antécédent en fonction des scores attribués à chaque candidat. Si l'ensemble A est finalement vide, l'anaphore n'est pas résolue et la mention courante était probablement non-anaphorique sans que le système n'ait su le détecter en phase 3.

2.2 Différentes méthodes de résolution

2.2.1 Méthodes à base de règles

Au niveau symbolique, les premières approches des années 70 sont plutôt développées pour la résolution des anaphores pronominales et font appel à des analyseurs syntaxiques pour l'annotation des catégories grammaticales en vue de la détection des pronoms et de leurs antécédents, et à la vérification de rôle syntaxique. Ces outils fournissent de solides indices sur lesquels se baser pour la résolution d'une anaphore, à partir desquels on peut construire différentes règles. Nous pourrions par exemple imaginer une règle spécifiant qu'un antécédent et son référent pronominal ne peuvent jamais apparaître au sein de la même phrase simple dans une même position syntaxique. A partir de cette règle, nous pouvons déduire que dans un énoncé tel que « Pierre le réconforte », les deux expressions *Pierre* et *le* ne peuvent référer à une même entité. D'autres analyseurs de type morphologique peuvent également être intégrés pour contraindre entre autres les correspondances de genre et nombre entre un antécédent et son anaphore. Certains systèmes plus récents comme celui de (Lappin and Leass, 1994) ajoutent à ces analyses un traitement du degré de saillance des référents pour un appariement plus pertinent. Par saillance, Catherine Schnedecker (2012) entend définir « le statut de *centralité* de certains référents dans la conscience de l'énonciateur » tout en concédant que la notion est controversée dans l'univers linguistique. Certaines approches soutiennent néanmoins qu'elle pourrait jouer un rôle intéressant dans le calcul du degré d'accessibilité du référent [2].

Ces précédents systèmes, bien qu'offrant des résultats relativement satisfaisants, dépendent beaucoup trop des ressources linguistiques, ainsi que d'heuristiques d'attribution de poids de saillance tout à fait discutables (une des plus remises en cause concerne notamment la récence de l'antécédent, selon laquelle un antécédent aurait un poids plus élevé par rapport à une anaphore s'il en est plus proche). D'autres approches symboliques se veulent moins gourmandes en connaissances linguistiques, comme c'est le cas de la plus reconnue, l'approche *knowledge-poor* décrite dans (Mitkov, 2002) qui, pour seul prétraitement, nécessite une analyse morphosyntaxique et un découpage en chunks simples [3]. A partir de ces données prétraitées, le système commence par repérer les expressions pronominales référentielles, puis, pour chacune d'entre elles, un algorithme combinant une dizaine de règles (parmi lesquelles l'accord en genre et nombre, la saillance, la distance référentielle entre un antécédent et sa reprise ou la répétition lexicale qui privilégie l'antécédent du texte le plus souvent mentionné) attribue à chaque antécédent un score puis sélectionne comme référent l'antécédent dont le score est le plus élevé. Ce type d'approche est intéressant à connaître, mais le précédent problème de dépendance aux heuristiques persiste, puisque ces méthodes reposent entièrement sur des règles mises en place par leurs auteurs, qui les pensent en fonction de leur point de vue linguistique sur le problème que constitue la résolution anaphorique.

2. Inspiré de la théorie de l'accessibilité, ce calcul permet d'attribuer un poids plus élevé à un antécédent au fort degré d'informativité (un SN complexe et descriptif sera par exemple plus informatif qu'un SN simple) et au fort degré de rigidité (*i.e.* si l'identification du référent est précise, comme par exemple un SN défini ou une entité nommée).
3. Un chunk se définit comme la plus petite séquence d'unités linguistiques dirigée par une tête forte.

Pour pallier ces biais théoriques, une des solutions est l'intégration de méthodes statistiques qui ne se basent que sur les données pour créer un modèle. Sans y recourir intégralement, le système hybride présenté dans (Longo, 2013) allie la performance des modèles statistiques à la connaissance des systèmes symboliques, en intégrant deux modules distincts :

1. Un module de segmentation statistique qui découpe le texte brut en sous-segments thématiques ;
2. Un module linguistique de repérage des marqueurs linguistiques de cohésion.

Avant la présentation de cette thèse, d'autres systèmes avaient su construire des modèles intégralement basés sur des méthodes d'apprentissage automatique, mais aucun néanmoins pour la résolution de la coréférence en français. Ce constat découle essentiellement du fait que jusqu'à maintenant, aucun corpus de français annoté en coréférence n'était disponible, or c'est tout ce dont a besoin un algorithme d'apprentissage pour construire un modèle.

2.2.2 Méthodes par apprentissage automatique

Lorsqu'une tâche est trop complexe pour être remplie par des moyens algorithmiques « classiques », les méthodes d'apprentissage automatique permettent à un système d'adapter ses analyses aux données qui lui sont fournies. Les travaux de recherche sur la résolution automatique de la coréférence se sont rapidement intéressés à ce type d'approche, qui ne dépend pas de la combinaison d'heuristiques, mais de calculs statistiques basés sur l'existant. Parmi ces méthodes, deux catégories se distinguent selon le mode d'apprentissage qu'elles emploient et la disponibilité d'exemples annotés : l'apprentissage supervisé (pour lequel les classes des données fournies sont connues) et l'apprentissage non-supervisé (pour lequel les classes ne sont pas connues).

2.2.2.1 Apprentissage supervisé

Cette méthode d'apprentissage est utile lorsqu'on dispose d'exemples annotés dont les classes sont pré-déterminées par un expert (il s'agit très souvent de données annotées manuellement), à partir desquels le système apprend un modèle de classement. Grâce à ce modèle, une phase de test consiste à fournir de nouvelles données dont on ne connaît pas la classe, et à prédire son étiquette.

Depuis les années 90 et le développement d'un certain nombre de corpus annotés en coréférence (majoritairement en anglais), l'apprentissage automatique pour la tâche de résolution de la coréférence est devenue classique. Différents types d'approches s'opposent dans la façon d'intégrer ces algorithmes d'apprentissage, parmi lesquels :

– les modèles *mention-pair* ou *pairwise* se décomposent en deux grandes phases dont la première est une classification binaire comparant une anaphore à des antécédents potentiels situés dans les phrases précédentes. Concrètement, les exemples fournis au programme sont des paires de mentions (une anaphore et un antécédent potentiel) pour lesquelles l'objectif est de déterminer si elles sont coréférentes ou pas. Une anaphore ne pouvant avoir qu'un unique antécédent, une deuxième phase doit déterminer, parmi les paires de mentions classées comme coréférentes, quel est le véritable antécédent d'une anaphore parmi tous ceux possibles : différentes approches ont été implémentées par différents systèmes, parmi lesquelles celles de (Soon et al., 2001) et (Ng and Cardie, 2002b), régulièrement utilisées comme système de référence à partir desquels les nouveaux systèmes comparent leur performance. Pour les premiers, l'antécédent sélectionné parmi un ensemble pour une anaphore donnée est celui qui en est le plus proche. Il s'agit d'un regroupement dit *Closest-First* qui, pour chaque anaphore, parcourt l'ensemble du texte vers la gauche, jusqu'à trouver un antécédent ou atteindre le début du texte. Les seconds proposent une alternative

à cette approche, dit regroupement *Best-First*, qui sélectionne comme antécédent celui ayant le plus haut score de « probabilité coréférentielle » parmi l'ensemble des précédentes mentions coréférentes.
L'inconvénient de ce type de méthode est la considération binaire du problème de la coréférence, qui ne prend pas en compte l'aspect dépendant des différents maillons d'une même chaîne de coréférence.

- les modèles *twin-candidate*, proposés dans (Yang et al., 2003) considèrent également le problème comme une tâche de classification, mais dont les instances sont cette fois composées de trois éléments (x, y_i, y_j) où x est une anaphore et y_i et y_j deux antécédents candidats (y_i étant le plus proche de x en terme de distance). L'objectif du modèle est d'établir des critères de comparaison des deux antécédents pour cette anaphore, et de classer l'instance en FIRST si le bon antécédent est y_i et en SECOND si le bon antécédent est y_j. Cette classification alternative est intéressante puisqu'elle ne considère plus la résolution de la coréférence comme l'addition de résolutions anaphoriques, mais prend en compte l'aspect « concurrentiel » des différents antécédents possibles pour une anaphore.
- les modèles *mention-ranking* tels que celui décrit dans (Denis, 2007), envisagent non plus d'étiqueter chaque paire de mentions mais de classer l'ensemble des antécédents possibles pour une anaphore donnée selon un processus itératif qui compare successivement cette anaphore à deux antécédents potentiels : à chaque itération, on conserve le meilleur candidat, puis on forme une nouvelle paire de candidats avec ce *gagnant* et un nouveau candidat. L'itération s'arrête lorsqu'il n'y a plus de candidat possible. Une alternative à cette méthode propose de comparer simultanément tous les antécédents possibles pour une anaphore donnée.
- Les modèles *entity-mention* déterminent quant à eux la probabilité qu'une expression réfère à une entité ou à une classe d'entités précédemment considérées comme coréferentes (*i.e.* un candidat est comparé à un unique antécédent ou à un cluster contenant toutes les références à une même entité).

Ces trois types de modèles n'obtiennent pas de résultats significativement meilleurs que les systèmes à base de règles, mais présentent l'avantage d'être moins dépendants des ressources. Néanmoins, les modèles générés automatiquement sont fortement liés au corpus d'exemples sur lequel ils ont été appris, et risquent de ne pas être exhaustifs quant à la multitude d'expressions référentielles qu'il est possible de rencontrer en testant le modèle sur de nouvelles données.

2.2.2.2 Apprentissage non supervisé

C'est pour contourner la dépendance à de grands volumes d'exemples annotés que les méthodes d'apprentissage non-supervisé ont vu le jour. Ce qui est fourni au programme d'apprentissage est le même format de données que précédemment (des instances décrites par une série d'attributs) pour lesquelles cette fois la classe n'est pas connue. Concernant la tâche de résolution de la coréférence, peu de systèmes basés sur de l'apprentissage non supervisé ont su démontrer une amélioration significative par rapport aux systèmes précédents. l'un d'entre eux toutefois est parvenu à faire ses preuves en démontrant des performances comparables à celles des systèmes supervisés : il s'agit du système de (Haghighi and Klein, 2007), cité par (Longo, 2013), qui s'appuie sur une série de traits linguistiques pour construire différents clusters, dont chacun correspond en fait à une chaîne de coréférence. L'avantage est que cette méthode permet de ne pas préciser par avance le nombre de clusters souhaités, laissant libre le système de retourner autant de chaînes qu'il en a détecté.

Chapitre 3

Présentation du corpus ANCOR

Dans le cadre de notre travail, nous souhaitons utiliser des méthodes à base d'apprentissage supervisé pour notre système de résolution de la coréférence, et pour ce faire, nous devons disposer d'un corpus annoté en coréférence, suffisamment volumineux pour que les données qui le constituent soient représentatives du phénomène et pertinentes pour l'apprentissage automatique. C'est en réponse à ce besoin que nous travaillons sur le corpus ANCOR, à ce jour l'un des seuls corpus du français annoté en coréférence.

3.1 Description du projet

Le projet ANCOR [1] (Anaphore et coréférence dans les corpus oraux) étudie les formes de reprise dans le discours, en annotant les phénomènes anaphoriques et de coréférence sur différents corpus oraux recueillis par la région centre (ESLO_ANCOR, ESLO_CO2, OTG, Acceuil_UBS), et retranscrits via Transcriber [2]. Plutôt que de se limiter à décrire les anaphores nominales, son objet d'étude s'élargit aux phénomènes d'anaphore associative et d'anaphore pronominale, et propose au total plus de 116 000 mentions et plus de 50 000 relations entre mentions répartis tel qu'illustré dans le tableau suivant :

Corpus	ESLO_ANCOR	ESLO_CO2	OTG	Acceuil_UBS	TOTAL
Nb. mentions	97,939	8,798	7,462	1,872	116,071
Nb. de relations	44,597	3,513	2,572	655	51,337

Table 3.1 – Répartition des données dans le corpus ANCOR

Notons ici que le sous-corpus ESLO_CO2 présentait au début de ce travail d'importants problèmes d'encodage, dûs à des enregistrements de fichiers successifs sous différents formats. Plus précisément, tous les diacritiques des fichiers étaient remplacés par un unique caractère, ?, rendant impossible un nettoyage automatique qui chercherait à remplacer chaque caractère mal encodé par son équivalent bien encodé. Après avoir contacté les administrateurs du projet à ce sujet, nous disposons aujourd'hui des mêmes données corrigées, mais les traitements pour adapter leur format aux algorithmes de calcul du modèle d'apprentissage automatique étant assez conséquents, nous les avons exploités en l'état.

1. http ://tln.li.univ-tours.fr/Tln_Ancor.html
2. http ://trans.sourceforge.net/en/presentation.php

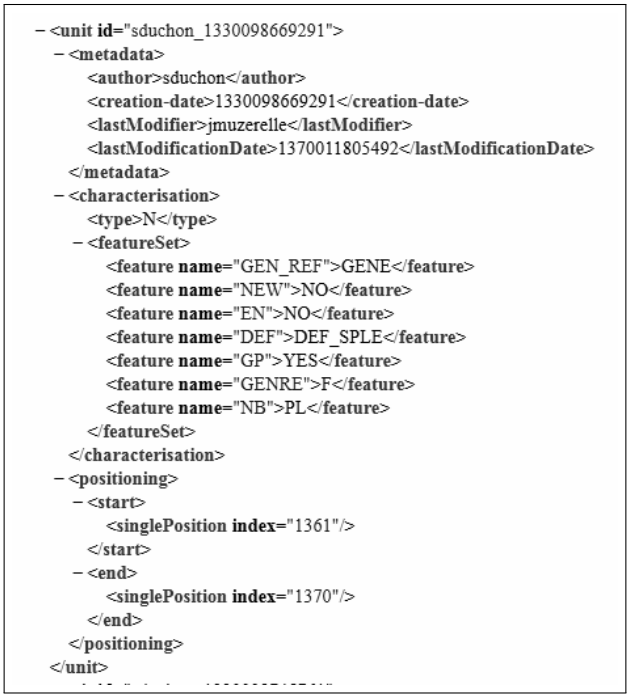


FIGURE 3.1 – Format d'annotation des unités dans ANCOR

3.2 Description des données

3.2.1 Format d'annotation

L'annotation du corpus a été faite via la plate-forme logicielle Glozz [3] qui permet de créer et d'utiliser son propre format d'annotation. Le format de sortie de l'outil est celui d'une annotation déportée, avec d'une part un fichier au format **.ac*, qui contient le corpus même (*i.e.* une transcription d'enregistrements), d'autre part, un fichier au format **.aa*, qui contient les annotations apportées au corpus correspondant. Les fichiers d'annotations comprennent différents types d'informations, parmi lesquels deux principaux nous intéressent :

1. L'annotation des unités (*i.e.* un maillon de chaîne de coréférence) telle qu'illustrée en 3.1 qui contient à la fois des informations linguistiques (*e.g. genre, nombre, type d'expression, etc*), et des informations concernant la position de l'unité, permettant de délimiter la chaîne de caractères dont il est question dans le corpus correspondant.
2. L'annotation des relations (*i.e.* une relation binaire entre deux unités) telle qu'illustrée en 3.2, dont les deux éléments (*<term>*) renvoient chacun à une unité précédemment annotée.

La figure 3.3, illustrant le format **.ac*, nous fait remarquer la présence de balises xml au sein du fichier corpus : il s'agit en fait d'éléments au format **.trs* de Transcriber, logiciel utilisé pour la retranscription des enregistrements du projet. C'est ce type de fichier qui a été fourni à Glozz pour être annoté, ce qui fait émerger en sortie quelques incohérences d'annotation du fait que les balises xml sont considérées comme du texte brut. C'est notamment le cas de l'annotation des paragraphes, que Glozz repère en se basant sur les retours à la ligne. Or dans un fichier **.trs*, les éléments xml sont indentés et chaque nouvelle ligne fait l'objet d'un nouveau paragraphe dans le fichier d'annotation.

3. http ://www.glozz.org/

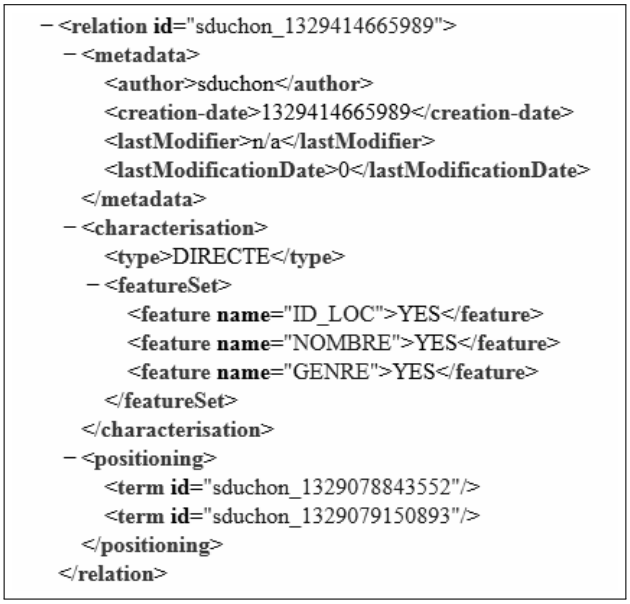


FIGURE 3.2 – Format d'annotation des relations dans ANCOR

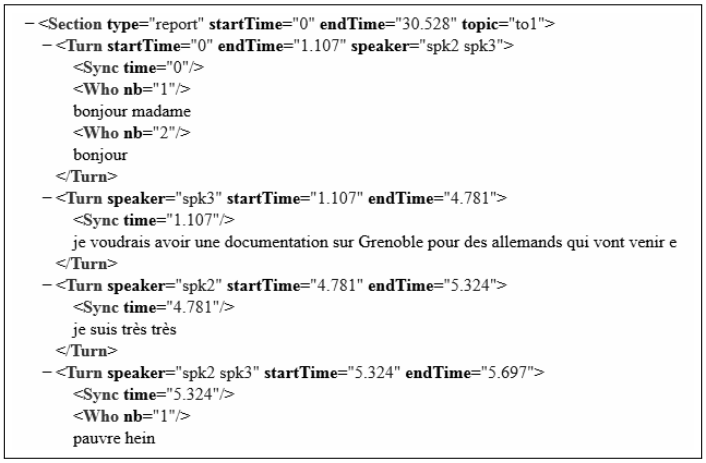


FIGURE 3.3 – Format du corpus dans ANCOR

Une autre remarque quant à ce corpus tient au choix des annotateurs de relier chaque mention considérée comme faisant partie d'une chaîne à la première mention de la chaîne, et non à la précédente. Le fait est remarquable puisque tous les systèmes de l'état de l'art étudiés au cours de ce travail ont fait le choix contraire de relier chaque maillon d'une chaîne au précédent, et non au premier. Toutes ces observations devront être prises en compte lors de l'adaptation du format des données à notre problématique, afin d'éviter d'inclure des biais dans la phase d'apprentissage du modèle de résolution de la coréférence.

3.2.2 Types de métadonnées

Dans le projet ANCOR, les relations unissant deux expressions référentielles sont distinguées selon leur type, c'est-à-dire la nature de la reprise. La reprise fidèle (dite *directe* dans ANCOR) se définit comme une relation dans laquelle les deux groupes nominaux réfèrent à la même entité du discours, et qui ont la même tête nominale (*La voiture rouge(...)* & *Cette belle voiture(...)*). La reprise infidèle (dite *indirecte* dans ANCOR) se distingue de la précédente par le fait que les deux mentions qu'elle relie ont des têtes nominales distinctes, qui entrent néanmoins souvent dans une relation sémantique telle que la synonymie, l'hyponymie ou l'hyperonymie (*Le cabriolet(...)* & *La voiture(...)*). La relation la plus représentée dans le corpus est l'anaphore pronominale (simplement dite *anaphore* dans ANCOR), c'est-à-dire la reprise d'un groupe nominal par un pronom (*La voiture(...)* & *Elle(...)*). Les deux dernières relations représentées relèvent de l'anaphore associative, c'est-à-dire que les deux expressions en relation réfèrent à des entités distinctes, qui rentrent dans une relation méronymique de type tout/partie. Parmi ces phénomènes, l'annotation d'ANCOR distingue l'anaphore associative nominale, reliant deux syntagmes nominaux (*La voiture(...)* & *La portière(...)*) et l'anaphore associative pronominale, reliant un syntagme nominal à un pronom (*Les classeurs(...)* & *L'un d'entre eux(...)*). Les unités annotées se voient elles aussi attribuer des informations concernant

DIRECTE	INDIRECTE	ANAPHORE	ASSOCIATIVE	
			NOMINALE	PRONOMINALE
40%	7%	42%	10%	1%

Table 3.2 – Répartition des données par type de relation

leur type, c'est-à-dire soit nominal, soit pronominal puisque ce sont les deux formes de syntagmes auxquels se sont limitées les annotations d'ANCOR. Le choix de ne s'intéresser qu'à ces deux formes découle du constat de difficulté éprouvée par les annotateurs quant au choix de considérer ou de ne pas considérer une expression du discours comme une expression référentielle. Bien que l'annotation soit limitée par ce choix (par exemple, nous n'aurons pas d'informations concernant les syntagmes adverbiaux tels que *demain*, qui auraient pu constituer une coréférence temporelle), il en garantit davantage de fiabilité en restreignant les critères. Pour chacune des unités, qu'elle soit nominale ou pronominale, une série de traits linguistiques fournit des informations la décrivant :

1. Genre
2. Nombre
3. Type d'entité nommée
4. Définitude (Si l'expression est un syntagme nominal, s'agit-il d'un défini, indéfini, démonstratif ou explétif ?)
5. Inclusion dans un groupe prépositionnel
6. Nouvelle entité du discours
7. Caractère générique ou spécifique de la coréférence (*e.g.* l'expression *« Une voiture »* sera générique dans *« Une voiture, c'est pratique »* mais spécifique dans *« Une voiture arrive à gauche »*)

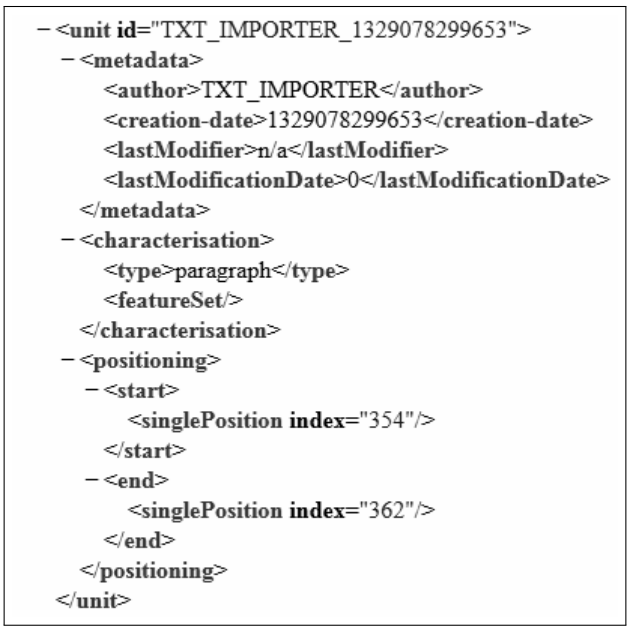


FIGURE 3.4 – Format d'annotation des paragraphes dans ANCOR

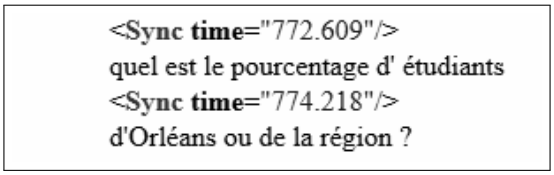


FIGURE 3.5 – Illustration de cas où un schéma est nécessaire

La plupart de ces informations seront reprises en tant que trait pour le corpus d'apprentissage du modèle (cf. section 4.1), et nous reviendrons alors plus précisément sur la définition de chacune.

3.3 Adaptation du format

3.3.1 Prétraitement des données

Certaines annotations du corpus ne sont pas directement liées à notre problématique, et dans un souci d'allègement des fichiers en vue de futurs traitements parfois assez lourds, nous décidons ici de les retirer de nos données. C'est le cas notamment des éléments *<unit>* définissant les paragraphes. Nous avons précédemment évoqué le problème que posait la génération automatique de ces éléments, qui s'appuie sur le saut de ligne pour en créer un nouveau. La figure 3.4 illustre la description de ce type d'unité dans un fichier au format **.aa*, et nous y observons que les indices de position ne sont en effet pas cohérents avec la taille d'un paragraphe (en l'occurrence, si cette unité représentait un paragraphe, celui-ci ne compterait que 12 caractères).
Par ailleurs, ANCOR annote des éléments qui sont des *<schema>*, utiles dans les cas où un syntagme nominal se retrouve scindé en plusieurs parties, du fait d'un chevauchement de parole ou d'un changement de locuteur (cf. figure 3.5). Le manuel d'annotation précise dans ce cas que doit être créé un élément *<schema>*, au sein duquel sont listées les unités

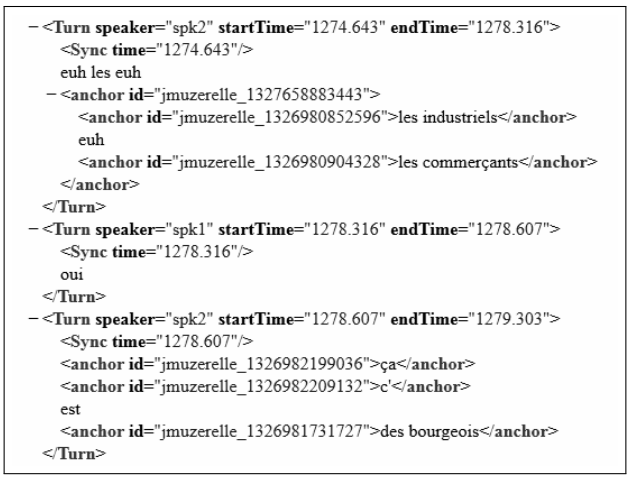


FIGURE 3.6 – Illustration de l'annotation des ancres

qui le construisent. Un dernier prétraitement nous permet finalement de laisser de côté les métadonnées générées automatiquement par Glozz lors de l'annotation. Ces données sont observables sur les précédentes illustrations du corpus, puisqu'il s'agit des élements *<metadata>*, dont les éléments fils sont des données non linguistiques qui ne nous seront pas utiles par la suite. Comme les deux précédents, nous les retirons de nos données.

3.3.2 Concaténation des fichiers

Le traitement suivant permet de rassembler en un unique fichier les données concernant le corpus et celles concernant ses annotations. Nous disposons en effet à ce stade de 459 fichiers au format **.ac* et pour chacun d'entre eux d'un fichier apparié au format **.aa*, soit un total de 918 fichiers. Il nous a semblé pertinent ici de rassembler les informations de chacune des paires de fichiers au sein d'un seul et même fichier, afin de faciliter les traitements suivants en évitant de parcourir divers répertoires en recherchant les fichiers souhaités. Ainsi, pour chaque paire, nous générons un nouveau fichier xml, dont la racine *<ANCOR>* a deux éléments fils : un *<Trans>* sous lequel sont intégrées les données du fichier corpus, puis un *<annotation>* sous lequel sont intégrées les données du fichier d'annotation.

3.3.3 Création d'unités ancrées

Les fichiers à ce stade du travail pourraient être exploités tels quels pour nos analyses, mais une dernière difficulté liée à l'identification des unités nous amène finalement à une dernière étape de prétraitement. Nous évoquions précédemment la manière dont sont délimitées les expressions, c'est-à-dire par des nœuds xml définissant les indices de début et de fin de l'unité en question. L'inconvénient de ce système est qu'il est nécessaire de considérer simultanément le contenu du fichier comme un arbre xml et comme une chaîne de caractères : en parcourant l'arbre, nous récupérons les indices de position, grâce auxquels nous recherchons dans la chaîne de caractères la sous-chaîne correspondant à ces coordonnées. En imaginant des traitements assez lourds, il est difficilement envisageable de devoir effectuer ce transfert à chaque itération. Nous envisageons donc d'insérer les expressions définies par des éléments *<unit>* au sein de nouveaux éléments *<anchor>*, pour chacun desquels est créé un identifiant unique dont la valeur est récupérée automatiquement à partir de l'identifiant de l'élément *<unit>* correspondant. Ces nouveaux éléments sont construits sur le principe des ancres, qui constituent un point de repère unique dans une masse de données volumineuse, en permettant de relier divers éléments

sur la base d'un identifiant commun.

La tâche est plus compliquée qu'il n'y paraît du fait de considérer le début du fichier xml (*i.e.* ce qui correspond au corpus) en tant que chaîne de caractères, à laquelle sont ajoutées itérativement des balises xml encadrant les unités analysées. La difficulté majeure de ce traitement tient à cette double lecture, qui nécessite un recalcul itératif des indices de position des unités : considérons par exemple un corpus composé de x caractères, et deux unités y et z telles que la première débute à l'indice *y1* et se termine à l'indice *y2*, et la seconde commence à l'indice *z1* et se termine à l'indice *z2*. A la première itération, les indices de l'unité y correspondent effectivement à sa position dans le corpus, et la création de l'élément *<anchor>* se fait proprement. En revanche, à la seconde itération, on ne peut plus se baser sur les indices *z1* et *z2*, puisque la taille du corpus a précédemment été augmentée par l'ajout des balises *<anchor>*. En prenant cela en compte, chaque fin d'itération doit opérer un calcul sur les indices de position, en augmentant itérativement les chiffres des tailles de chaînes de caractères ajoutées à l'itération précédente.

Ce dernier traitement a fait émerger de certains fichiers des annotations atypiques qui par exemple incluaient dans une unité référentielle une portion de balise xml, voire une balise complète. Certains de ces phénomènes, lorsqu'ils étaient récurrents, ont pu être corrigés automatiquement, mais lorsqu'ils représentaient un cas isolé difficile à prendre en compte, il a été décidé de laisser de côté le fichier en question. C'est pour cette raison que le corpus final se compose, après tous les prétraitements, de 452 fichiers dont les tailles varient entre 3Ko et 3Mo, pour un total de 113Mo de données.

Chapitre 4

Résolution de la coréférence comme tâche de classification

Un certain nombre de systèmes de l'état de l'art ont ramené le problème de la résolution de la coréférence à un problème de classification binaire, où chaque instance présentée au modèle correspond à une paire de mentions, pour laquelle la classe est soit « coréférente », soit « non-coréférente ». On parle de modèle *mention-pair*, qui compare deux à deux chaque mention du corpus initial, et reconstitue finalement les chaînes de coréférence en se basant sur leur propriété transitive. En effet, en considérant que 1, 2 et 3 sont trois mentions d'un texte, et que les paires [1,2] et [2,3] sont classées comme coréférentes, il est possible de déduire que la paire [1,3] l'est aussi, et de reconstituer la chaîne complète [1, 2, 3] par union des sous-ensembles de paires. La classification que nous entreprenons dans ce travail est donc binaire, c'est-à-dire que chaque instance est classée soit dans une classe, soit dans l'autre : en l'occurrence, soit dans la classe des paires de mentions coréférentes (COREF), soit dans celle des mentions non-coréférentes (NOT_COREF). Nous ferons désormais référence à ces classes en appelant la classe des positifs celle dont les instances sont coréférentes, et la classe des négatifs celle dont les instances ne le sont pas.

4.1 Définition de l'ensemble de traits

La pertinence d'un modèle automatique repose sur les données qui lui sont fournies pour son apprentissage, et particulièrement sur les traits linguistiques (dits *features*) décrivant le phénomène qu'il doit pouvoir résoudre. La résolution de la coréférence est un phénomène discursif complexe, et un système destiné à le résoudre automatiquement peut également le devenir. Néanmoins, (Bengtson and Roth, 2008) soulignent qu'un simple modèle de classification binaire est capable d'obtenir d'aussi bons résultats qu'un système plus complexe de ranking, grâce à la pertinence de son ensemble de features.

En observant les ensembles de traits des différents systèmes de l'état de l'art, nous remarquons qu'ils évoluent en reprenant les traits des premiers systèmes, précisément celui de (Soon et al., 2001), composé de 12 traits, qui va devenir l'ensemble de base à partir duquel tous les suivants s'appuieront pour développer le leur (cf. tableau 4.1). C'est également de celui-ci que nous nous inspirerons pour constituer notre propre ensemble, ainsi que sur celui de (Ng and Cardie, 2002b), qui ajoute aux 12 précédents traits 41 nouveaux, et de ceux de travaux plus récents qui reprennent eux-mêmes les deux précédents. Avant d'entrer dans les détails de chacun des traits linguistiques, il est important d'en distinguer deux grandes catégories nécessaires à décrire nos données : les traits relationnels, qui décrivent la relation unissant deux mentions d'une paire, et les traits non-relationnels, dé-

	Traits	Valeurs possibles
1	Distance en nombre de phrases entre i et j	nombre entier
2	i est-il un pronom ?	vrai ou faux
3	j est-il un pronom ?	vrai ou faux
4	Les chaînes de caractères de i et j sont-elles égales ?	vrai ou faux
5	j est-il un SN défini ?	vrai ou faux
6	j est-il un SN démonstratif ?	vrai ou faux
7	i et j s'accordent-ils en nombre ?	vrai ou faux
8	i et j s'accordent-ils en genre ?	vrai ou faux
9	i et j appartiennent-ils à la même classe sémantique ?	vrai ou faux
10	i et j sont-ils tous deux des noms propres ?	vrai ou faux
11	i et j sont-ils alias l'un de l'autre ?	vrai ou faux
12	i et j sont-ils au sein d'une structure appositive ?	vrai ou faux

TABLE 4.1 – Ensemble de features de (Soon et al., 2001) entre un antécédent i et une anaphore j

crivant simplement une mention. En plus de cette distinction, les valeurs de chaque trait peuvent être de différente nature, ainsi certains peuvent être des chaînes de caractères, quand d'autres sont des nombres réels ou encore des booléens. Dans les descriptions de traits suivantes, nous considérons une mention m_1 comme un antécédent, et une mention m_2 comme une anaphore dont le système doit vérifier qu'elle coréfère ou non à m_1.

4.1.1 Traits non-relationnels

Ce type de traits est destiné à décrire chacune des deux mentions d'une paire, indépendamment l'une de l'autre. Ainsi, les informations y ayant trait apparaîtront toujours en double, puisqu'une première sera définie pour m_1, et une seconde pour m_2.

4.1.1.1 Informations grammaticales

Les traits 2, 3, 5, 6, 7 et 8 de l'ensemble de (Soon et al., 2001) sont des informations grammaticales de type non-relationnel. Nous les reprenons pour notre propre ensemble, en les adaptant aux possibilités offertes par l'annotation du corpus ANCOR. Ainsi, voilà comment ce type d'informations est intégré à notre ensemble de traits (les valeurs possibles étant celles proposées par ANCOR dans son guide d'annotation - cf. (Antoine et al., 2013)) :

- Type d'expression de m_1 = {NOM, PRONOM} ;
- Type d'expression de m_2 = {NOM, PRONOM} ;
- Degré de définitude de m_1 = {INDÉFINI, DÉFINI DÉMONSTRATIF, DÉFINI NON DÉMONSTRATIF, EXPLÉTIF} ;
- Degré de définitude de m_2 = {INDÉFINI, DÉFINI DÉMONSTRATIF, DÉFINI NON DÉMONSTRATIF, EXPLÉTIF} ;
- Genre de m_1 = {MASCULIN, FÉMININ} ;
- Genre de m_2 = {MASCULIN, FÉMININ} ;
- Nombre de m_1 = {SINGULIER, PLURIEL} ;
- Nombre de m_2 = {SINGULIER, PLURIEL}.

4.1.1.2 Informations contextuelles

Pascal Denis (Denis, 2007) décrit dans sa thèse un ensemble de traits dans lequel il prend en compte le contexte de chacune des deux mentions. Plus précisément, il s'intéresse aux catégories syntaxiques des tokens précédents et suivants de ces deux mentions, en

argumentant que ce type d'informations pourrait aider à retrouver le rôle grammatical de la mention analysée. Par exemple, si une mention est suivie d'un verbe conjugué, cette mention correspondra la plupart du temps à un sujet. Ces informations contextuelles auraient été intéressantes à intégrer à notre ensemble de traits, mais il est assez compliqué d'y accéder, puisque nous ne disposons pas d'étiqueteur POS spécialisé dans l'annotation de données orales. A défaut de leur catégorie syntaxique, nous adaptons le problème en récupérant les deux tokens à gauche et à droite de chaque mention. Dans le cas où une mention débute un tour de parole, nous choisissons d'annoter son token précédent par ^, et dans le cas où une mention termine un tour de parole, l'annotation de son token suivant est $.

- Token précédant m_1 = STRING ;
- Token précédant m_2 = STRING ;
- Token suivant m_1 = STRING ;
- Token suivant m_2 = STRING.

Il sera moins évident lors de l'apprentissage de généraliser les contextes comme il aurait été possible de le faire avec les étiquettes POS, grâce auxquelles le modèle aurait appris des patrons syntaxiques récurrents. Nous observerons lors de la phase d'expérimentation la manière dont ces quatre traits influencent l'apprentissage.

4.1.1.3 Informations énonciatives

Ce type d'information relève principalement de la prise en charge énonciative des expressions référentielles. Certains systèmes de l'état de l'art comme ceux de (Ng and Cardie, 2002b), (Stoyanov et al., 2010) ou encore de (Raghunathan et al., 2010) intègrent par exemple des traits vérifiant si une mention apparaît entre guillemets. Ces derniers expliquent ce choix par une simple heuristique de détection automatique de locuteur, selon laquelle « *I* » et « *she* » feraient référence à la même personne dans l'énoncé *« [I] voted my conscience », [she] said.*

Nous avons la chance de disposer d'un corpus oral, qui contient entre autres des annotations sur les locuteurs, qu'il nous est possible de récupérer pour chaque mention analysée. Ceci nous permettra entre autres de distinguer les emplois des pronoms de première personne, dont il sera plus facile de résoudre la coréférence en en connaissant le locuteur. Les deux nouveaux traits intégrés dans l'ensemble auront pour valeur une chaîne de caractères du type *spk1*, *spk2*, décrivant chacun des locuteurs présents dans la conversation. L'ensemble du corpus nous permettait de générer un ensemble de valeurs possibles à partir de ces annotations, mais le modèle ne serait pas réutilisable sur un corpus dont les noms de locuteurs n'apparaîtraient pas dans cet ensemble.

- Locuteur de m_1 = STRING ;
- Locuteur de m_2 = STRING.

Le corpus annoté dont nous disposons fournit une autre information qu'il est difficile de catégoriser, mais qui semble relever de l'énonciation : c'est ce que Denis (2007) décrit comme « le statut discursif de l'expression ». Dans sa thèse, celui-ci développe un système de classification dont la tâche est de déterminer si une mention m d'un document introduit ou non une nouvelle entité dans le discours (à l'issue de cette tâche, chaque mention sera annotée de la valeur de sa classe, soit NEW si l'entité référentielle est nouvelle dans le discours, soit OLD si une autre mention l'avait déjà introduite auparavant). Cette classification constitue dans ses différents systèmes une étape préliminaire, destinée à faciliter la tâche du système global, qui n'aurait à rechercher un antécédent que pour les expressions classées en OLD : en effet, si une expression est classée en NEW, c'est que celle-ci constitue le premier maillon d'une nouvelle chaîne, et donc qu'aucune expression précédente ne réfère à la même entité, il n'est donc pas utile de lui rechercher un antécédent. Cette information est intégrée au corpus ANCOR sous la forme d'un attribut NEW,

Valeur de traits	Description	Définition
PERS	PERSONNE	Personne réelle ou fictive et animal
FONC	FONCTION	Fonction politique, militaire, administrative, religieuse, ...
LOC	LIEU	Géonyme, région administrative, axe de circulation, adresse, construction humaine ...
ORG	ORGANISATION	Organisation politique, éducative, commerciale, géo-administrative ...
PROD	PRODUCTION HUMAINE	Classe très vague : moyen de transport, œuvre artistique, film ...
TIME	DATE ET HEURE	Date relative ou absolue, heure. Les durées sont dans la classe AMOUNT
AMOUNT	MONTANT	Age, durée, température, longueur, aire, volume, poids, vitesse, valeur monétaire, ...
EVENT	ÉVÈNEMENT	exemple : *la fête nationale*
NO	ne correspond pas à une EN	

TABLE 4.2 – Description des types d'entités nommées dans le corpus ANCOR

qui pour chaque expression peut prendre la valeur YES ou NO. Il nous est alors simple de récupérer ce trait pour notre ensemble :

- m_1 introduit une nouvelle entité = TRUE or FALSE ;
- m_2 introduit une nouvelle entité = TRUE or FALSE.

4.1.1.4 Informations sémantiques

Il s'agit certainement du type d'information parmi les plus compliqués à récupérer automatiquement, puisqu'il nécessite l'intégration de ressources externes telles que des dictionnaires, thésaurus, ontologies ou autres systèmes de repérage automatique d'entités nommées. L'ensemble de traits des précurseurs (Soon et al., 2001) intégrait déjà des informations de ce type en associant leurs données à celles de WordNet [1]. Cette ressource permet entre autres de répondre au trait 9 du tableau 4.1, vérifiant que deux mentions d'une paire appartiennent ou non à une même classe sémantique.

Nous ne disposons pas de ressources telles que celle-ci pour le français, et aucune annotation de ce type n'apparaît dans le corpus ANCOR. Néanmoins, celui-ci intègre une autre annotation sémantique que des travaux comme ceux de (Bengtson and Roth, 2008) ou de (Recasens, 2010) ont introduit dans leur ensemble de traits : celle concernant les entités nommées. Selon (Recasens, 2010), cette information est particulièrement pertinente dans le cas de l'analyse d'articles de presse, contenant beaucoup d'entités de type *person* et *organization*, susceptibles d'appartenir à des chaînes de coréférence. L'information nous paraît également pertinente car elle pourrait constituer un facteur décisif de classement en négatif lorsque les deux mentions d'une paire ne sont pas de même type : en effet, une expression de type *localisation* ne pourra jamais être coréférente à une expression de type *person*. L'annotation du corpus ANCOR est particulièrement riche concernant le typage des entités nommées, puisqu'il en distingue 8 qui constitueront l'ensemble des valeurs possibles des deux nouveaux traits intégrés à notre ensemble [2] :

1. Wordnet est une base de données lexicale de l'anglais présentant les relations sémantiques entre termes sous formes d'ensembles de synonymes (http ://wordnet.princeton.edu/).

2. Mis à part la classe *event*, l'ensemble des classes reprend la classification de la campagne d'évaluation ESTER2 (http ://www.afcp-parole.org/camp_eval_systemes_transcription/).

- Type d'entité nommée de m_1 = {PERS, FONC, LOC, ORG, PROD, TIME, AMOUNT, EVENT, NO} ;
- Type d'entité nommée de m_2 = {PERS, FONC, LOC, ORG, PROD, TIME, AMOUNT, EVENT, NO}.

Pour plus de précision, le tableau 4.2 décrit la signification des abréviations des classes précédentes telle que décrite dans le guide d'annotation d'ANCOR (Antoine et al., 2013).

4.1.2 Traits relationnels

Les traits relationnels sont utiles à comparer les deux mentions d'une paire, à partir des précédents traits non-relationnels de chacune d'entre elles, ainsi que par différents calculs de distance.

4.1.2.1 Distances lexicales

Cette première catégorie s'intéresse à la comparaison des formes linguistiques de m_1 et m_2, à différents degrés :

- Identité totale = TRUE or FALSE ;
- Identité partielle ;
  - Sous-forme = TRUE or FALSE ;
  - Taux d'inclusion = RÉEL ;
  - Taux de tokens communs = RÉEL.

Le premier de ces traits compare strictement les deux chaînes de caractères, et prend la valeur TRUE uniquement si elles sont exactement identiques. S'agissant des égalités partielles, nous décomposons l'information en plusieurs parties afin de rendre compte au mieux de la distance lexicale séparant les deux chaînes. Le trait *Sous-forme* prend la valeur TRUE si la plus petite des deux mentions est complètement incluse dans la plus grande (la comparaison est stricte, c'est-à-dire que la grande chaîne doit contenir la chaîne exacte de la petite, sans inclusion d'éléments entre les tokens qui la composent) ; l'information sur le taux d'inclusion découle de celle-ci puisqu'elle compare le nombre de mots de la plus petite mention au nombre de mots qu'elle partage avec la seconde mention (l'information est plus précise car même si les tokens de la petite chaîne sont dispersés dans la grande, la valeur restera 1). Le dernier trait, concernant le taux de tokens communs, s'obtient par le rapport de l'intersection des deux ensembles de tokens formés par m_1 et m_2 et de leur union.

Nous illustrons ces précédents calculs grâce à l'exemple suivant :

$$\begin{array}{rcl} m_1 & = & \text{« } \textit{la grande ville} \text{ »} \\ m_2 & = & \text{« } \textit{la ville} \text{ »} \end{array}$$

$$\begin{array}{rcl} \text{ID\_FORM} & = & \text{FALSE} \\ \text{ID\_SUBFORM} & = & \text{FALSE} \\ \text{INCL\_RATE} & = & 1 \\ \text{COM\_RATE} & = & \frac{2}{3} \end{array}$$

Le trait ID_SUBFORM est interprétable à partir du résultat du calcul de INCL_RATE, uniquement si sa valeur est inférieure à 1, auquel cas la valeur de ID_SUBFORM est forcément FALSE ; en revanche, le cas contraire où la valeur de INCL_RATE est 1 n'induit pas que celle de ID_SUBFORM soit TRUE (comme l'illustre l'exemple précédent).

4.1.2.2 Distances grammaticales

Ces traits s'obtiennent en comparant les valeurs des informations grammaticales de chacune des mentions d'une paire. Les valeurs de ces nouveaux traits sont toutes booléennes, puisque l'égalité est vraie si les valeurs des deux mentions sont identiques, fausse sinon :

- Identité du type d'entité nommée
- Identité du degré de définitude
- Identité du type d'expression
- Identité du genre
- Identité du nombre

} TRUE or FALSE.

4.1.2.3 Distances spatiales

L'ensemble de référence de (Soon et al., 2001) propose en terme de distance spatiale un attribut concernant le nombre de phrases séparant les deux mentions d'une paire (premier attribut du tableau 4.1). Les travaux suivants de (Recasens and Hovy, 1997), (Ng and Cardie, 2002b), (Denis, 2007), (Bengtson and Roth, 2008), (Recasens, 2010) et (Stoyanov et al., 2010) le reprennent et lui ajoutent un trait du même type qui calcule la distance entre deux mentions en terme de nombre de paragraphes. Ces mêmes travaux, mis à part celui de (Ng and Cardie, 2002b), intègrent également à leur ensemble un trait décrivant la distance entre deux expressions en nombre de mentions, ainsi que pour certains (hormis (Denis, 2007)), un autre décrivant la distance en nombre de mots.

Les données dont nous disposons permettent d'intégrer à notre ensemble d'attributs certaines de ces informations, comme la distance en nombre de mentions et celle en nombre de mots. Il est en revanche plus compliqué de calculer celles s'appuyant sur des unités telles que la phrase ou le paragraphe, qui sont propres à l'écrit. Le format d'annotation de Transcriber structure en revanche ses données sous forme de tours de paroles, grâce auxquels il nous est possible de calculer la distance séparant deux mentions d'une paire en terme de nombre de tours de parole. A tous ces attributs, nous ajoutons également un trait décrivant la distance séparant deux mentions en nombre de caractères, pour obtenir un sous-ensemble de quatre traits déterminant divers degrés de distance entre mentions :

- Distance en nombre de tours de parole = ENTIER ;
- Distance en nombre de mentions = ENTIER ;
- Distance en nombre de mots = ENTIER ;
- Distance en nombre de caractères = ENTIER.

Une autre information spatiale décrite dans les travaux de (Recasens and Hovy, 1997), (Ng and Cardie, 2002b), (Denis, 2007), (Recasens, 2010) et (Stoyanov et al., 2010) s'intéresse à vérifier si une mention est incluse dans une autre mention plus grande : en d'autres termes, la mention analysée fait-elle partie d'un syntagme nominal complexe ? Ces travaux intègrent alors à leurs ensembles d'attributs deux traits supplémentaires répondant à cette question pour chacune des deux mentions d'une paire. Recasens, dans ses travaux de (1997) et (2010), décrit un trait similaire vérifiant si deux mentions d'une paire sont incluses l'une dans l'autre, plus précisément si m_2, la plus à droite des deux selon l'ordre linéaire du texte, est incluse dans la précédente m_1. C'est cet attribut que nous choisissons d'intégrer à notre ensemble, puisqu'il semble pouvoir jouer un rôle déterminant pour la classification d'une paire de mentions : en effet si m_2 fait partie de m_1 (*e.g.* « *[la voiture de [mon frère]]* »), m_2 apporte des informations complémentaires à m_1, mais les deux ne pourront jamais référer à une même entité.

- Inclusion = TRUE or FALSE.

4.1.2.4 Distances contextuelles

Nous nous intéressons dans cette catégorie au calcul des attributs vérifiant la correspondance entre les contextes des deux mentions d'une paire : un premier attribut vérifie l'égalité ou non des tokens à gauche de m_1 et m_2 ; un second fait de même sur les deux tokens à droite. De même que pour les attributs non-relationnels dont les résultats de ces

Soit la chaîne [1,2,3,4,5,6] où chaque chiffre est un maillon					
Annotation des relations dans le corpus (maillon, premier maillon)	[1,2]	[1,3]	[1,4]	[1,5]	[1,6]
	DEF	DEF	DEF_SPLE	DEF_DEM	INDEF
Annotation des relations telles que nous les considérons (maillon, maillon précédent)	[1,2]	[2,3]	[3,4]	[4,5]	[5,6]

FIGURE 4.1 – Exemple illustrant les différentes possibilités de relier les maillons d'une chaîne de coréférence

calculs sont issus, nous pouvons d'ores et déjà augurer que ces informations ne constitueront pas pour le futur modèle d'apprentissage des facteurs suffisamment homogènes sur lesquels s'appuyer. Ce sera lors de la phase d'expérimentations que nous pourrons évaluer l'influence de ces deux traits sur le modèle :

- Identité du token précédent = TRUE or FALSE ;
- Identité du token suivant =TRUE or FALSE.

4.1.2.5 Distances énonciatives

De la même façon que pour les précédents, les traits de cette catégorie sont calculés à partir des traits non-relationnels de cette même catégorie, en l'occurrence, il s'agit du locuteur de chacune des mentions d'une paire, et de leur statut discursif respectif. Les deux nouveaux attributs sont donc ceux dont le calcul vérifie ou non l'identité du locuteur et l'identité de leur statut :

- Identité du locuteur = TRUE or FALSE ;
- Identité du statut discrusif = TRUE or FALSE.

4.1.3 Conclusion

A l'issue des recherches sur les ensembles de traits des différents travaux de l'état de l'art, corrélés avec les annotations disponibles à partir du corpus ANCOR, nous avons pu constituer un ensemble de 36 traits, décrits dans le tableau 4.3. Quelques remarques sont à faire concernant les traits que nous n'avons pu récupérer parmi les ensembles des systèmes précédemment cités. En plus des informations sémantiques telles que la classe WordNet, d'autres informations complexes sont également inaccessibles sans prétraitement des données : c'est notamment le cas de l'attribut que Recasens (1997 & 2010) qualifie de « classique pour décrire la coréférence », qui vérifie l'identité des têtes lexicales des deux mentions d'une paire lorsqu'il s'agit de syntagmes nominaux. Ce trait, repris également dans les travaux de (Bengtson and Roth, 2008) et (Stoyanov et al., 2010), est en effet plus pertinent que la comparaison stricte des chaînes de caractères des deux mentions, puisque l'élément important à comparer est la tête du syntagme, et non ses modifieurs. L'annotation d'ANCOR permet en fait de récupérer cette information, grâce à l'annotation du type de relation existant entre deux mentions (cf. section 3.2.2), qui, si la valeur est DIRECTE, signifie que la reprise se fait par une expression de même tête lexicale. Rappelons que l'annotation d'ANCOR choisit de relier chaque mention au premier maillon de la chaîne de coréférence concernée, or par la suite, nous choisissons d'adapter la description de ces relations au format de celles des systèmes de l'état de l'art, c'est-à-dire en reliant chaque maillon à son précédent (Nous détaillerons la manière dont nous procédons à cette transformation en section 4.2.1.1). Transformer ainsi la représentation du corpus initial nous fait perdre l'information pertinente sur le type de relation entre

	Traits	Définition	Valeurs possibles
1	m_1_TYPE	catégorie syntaxique de m_1	{N, PR, UNK, NULL}
2	m_2_TYPE	catégorie syntaxique de m_2	{N, PR, UNK, NULL}
3	m_2_DEF	détermination de m_1	{INDEF, EXPL, DEF_SPLE, DEF_DEM, , UNK}
4	m_2_TYPE	détermination de m_2	{INDEF, EXPL, DEF_SPLE, DEF_DEM, , UNK}
5	m_1_GENRE	genre de m_1	{M, F, UNK, NULL}
6	m_2_GENRE	genre de m_2	{M, F, UNK, NULL}
7	m_1_NOMBRE	nombre de m_1	{SG, PL, UNK, NULL}
8	m_2_NOMBRE	nombre de m_2	{SG, PL, UNK, NULL}
9	m_1_PREVIOUS	token précédent m_1	STRING
10	m_2_PREVIOUS	token précédent m_2	STRING
11	m_1_NEXT	token suivant m_1	STRING
12	m_2_NEXT	token suivant m_2	STRING
13	m_1_SPK	Locuteur de m_1	STRING
14	m_2_SPK	Locuteur de m_2	STRING
15	m_1_NEW	Nouvelle entité introduite par m_1	{YES, NO, UNK, NULL}
16	m_2_NEW	Nouvelle entité introduite par m_2	{YES, NO, UNK, NULL}
17	m_1_EN	Type d'entité de m_1	{PERS, FONC, LOC, ORG, PROD, TIME, AMOUNT, EVENT, NO, UNK, NULL}
18	m_2_EN	Type d'entité de m_2	{PERS, FONC, LOC, ORG, PROD, TIME, AMOUNT, EVENT, NO, UNK, NULL}
19	ID_FORM	Identité des formes de m_1 et m_2	{YES, NO, NA}
20	ID_SUBFORM	Identité d'une sous-forme	{YES, NO, NA}
21	INCL_RATE	Taux d'inclusion des tokens	REAL
22	COM_RATE	Taux de tokens communs	REAL
23	ID_DEF	Identité de détermination	{YES, NO, NA}
24	ID_TYPE	Identité de type	{YES, NO, NA}
25	ID_EN	Identité d'entité nommée	{YES, NO, NA}
26	ID_GENRE	Identité de genre	{YES, NO, NA}
27	ID_NOMBRE	Identité de nombre	{YES, NO, NA}
28	DISTANCE_MENTION	Distance en nombre de mention	REAL
29	DISTANCE_TURN	Distance en nombre de tours de parole	REAL
30	DISTANCE_WORD	Distance en nombre de mots	REAL
31	DISTANCE_CHAR	Distance en nombre de caractères	REAL
32	EMBEDDED	Enchâssement de m_2 dans m_1	{YES, NO, NA}
33	ID_PREVIOUS	Identité du token précédent	{YES, NO, NA}
34	ID_NEXT	Identité du token suivant	{YES, NO, NA}
35	ID_SPK	Identité du token précédent	{YES, NO, NA}
36	ID_NEW	Identité du token suivant	{YES, NO, NA}

TABLE 4.3 – Ensemble de traits linguistiques

```
@RELATION coreference
@ATTRIBUTE INCL_SLOT real
@ATTRIBUTE M2_NEW {YES,NO,UNK,NULL}
@ATTRIBUTE ID_SUBFORM {YES,NO,NA}
@ATTRIBUTE ID_PREVIOUS {YES,NO,NA}
[...]
@ATTRIBUTE ID_NEXT {YES,NO,NA}
@ATTRIBUTE class {COREF, NOT_COREF}
```

FIGURE 4.2 – Illustration du format des attributs dans un fichier **.arff*

deux mentions, puisque les paires que nous analysons ne sont pas directement décrites dans ce corpus : l'exemple de la figure 4.1 démontre qu'il n'est possible de connaître le type de relation que de la première paire d'une chaîne, qui est la même quelle que soit la façon d'envisager les relations ; pour les suivantes en revanche, il n'est pas possible de retrouver le type de relation à partir des annotations du corpus.

Une dernière remarque sur cet ensemble de traits concerne les valeurs que peuvent prendre certains d'entre eux. C'est ce que le tableau 4.3 décrit dans sa dernière colonne, dans laquelle certaines valeurs sont à expliquer : pour les attributs dont la valeur fait partie d'un ensemble restreint de valeurs possibles, l'annotation d'ANCOR ajoute à chaque fois deux valeurs, UNK et NULL. La première est utilisée dans le cas où l'annotateur n'a su se décider sur l'information à renseigner, et la seconde dans le cas où l'expression annotée est « un artefact lié à certaines disfluences ». Un exemple concret du recours à cette valeur est celui de l'expression *« Pierre et Marie Curie »*, décomposée dans l'annotation en trois unités que sont *Pierre*, *Marie Curie* et *Curie*. La dernière de ces unités aura pour valeur de type le susdit NULL, car même si le patronyme est commun aux deux personnes, les comparaisons d'accord en genre et nombre à d'autres unités référentielles seront plus pertinentes par rapport aux prénoms. Quant aux attributs dont la valeur est booléenne, s'ajoute aux traditionnels YES et NO une troisième valeur, NA, qui intervient dans le cas où les informations à comparer pour obtenir cette valeur ne sont pas comparables : si l'on souhaite par exemple comparer les deux types d'entités nommées de m_1 et m_2, mais que l'une des deux n'est pas une entité nommée, la comparaison ne peut se faire, et la valeur NA est alors pertinente.

4.2 Génération de l'ensemble d'apprentissage

Rappelons que notre objectif est la génération d'un modèle de résolution de la coréférence, par apprentissage supervisé. Il s'agit donc de fournir à un algorithme d'apprentissage un ensemble de données dont la classe est connue, pour l'aider à établir des critères de classification qui seront réutilisés lors de la phase de test sur des données non annotées dont il faudra retrouver la classe. Nous disposons à ce stade d'un corpus annoté en unités référentielles et relations anaphoriques, ainsi que d'un ensemble de traits linguistiques, à partir desquels il nous faut construire un ensemble de données d'apprentissage adapté au format d'entrée des algorithmes qui calculent les modèles.

4.2.1 Vectorisation des données

Dans ce travail, nous utilisons l'API Java de Weka[3], qui implémente divers algorithmes d'apprentissage supervisé et non supervisé, pour construire nos modèles. Un très grand nombre de classes et méthodes permettent de générer des classifieurs à partir de

3. http ://www.cs.waikato.ac.nz/ml/weka/

```
@DATA
1.0 NO YES NO LOC DEF_SPLE NA LOC 3 N NO UNK YES YES UNK YES NA UNK N YES 9 NO 30
124 NO 1.0 DEF_SPLE UNK YES COREF
0.0 NO NO NO LOC DEF_SPLE NA LOC 0 PR YES UNK NO NO UNK YES NA F N YES 4 YES 15
68 NO 0.0 DEF_SPLE SG NO NOT_COREF
0.0 NO NO NO LOC DEF_SPLE NA LOC 2 N YES UNK NO NO UNK YES NA F N YES 8 YES 29
121 NO 0.0 DEF_SPLE SG YES NOT_COREF
0.0 YES NO NO LOC DEF_DEM YES LOC 0 N YES SG NO NO F NO YES F N YES 1 NO 2 9 NO
0.0 DEF_SPLE SG NO COREF
1.0 NO YES NO LOC DEF_SPLE YES LOC 1 N NO SG YES NO F YES YES F N YES 3 NO 0 0 NO
0.16666667 DEF_SPLE SG YES COREF
0.0 NO NO NO LOC DEF_SPLE NA LOC 1 N NO SG YES NO F YES NA UNK N YES 1 NO 0 0 NO
0.0 DEF_SPLE UNK YES NOT_COREF
0.0 NO NO NO LOC DEF_SPLE NA LOC 1 N NO SG YES NO F YES NA UNK N YES 1 NO 0 0 NO
0.0 DEF_SPLE UNK YES NOT_COREF
0.0 NO NO NO LOC DEF_SPLE YES LOC 2 N YES SG NO NO F YES YES F PR YES 2 YES 4 7
NO 0.0 DEF_SPLE SG NO COREF
0.0 NO NO NO PERS DEF_SPLE YES LOC 0 PR YES SG NO NO F YES YES F PR NO 1 YES 0 0
NO 0.0 DEF_SPLE SG NO NOT_COREF
0.0 NO NO NO LOC DEF_SPLE YES LOC 0 PR YES SG NO NO F YES YES F PR YES 2 YES 8 36
NO 0.0 DEF_SPLE SG NO COREF
```

FIGURE 4.3 – Illustration du format des instances dans un fichier **.arff*

données fournies, puis de les réutiliser sur de nouvelles données. Weka exige en entrée de ses programmes un format de données particulier, appelé *Attribute-Relation File Format* (**.arff*), qui décrit à la fois tous les attributs nécessaires à la description des données, puis les données elles-mêmes, structuré comme suit :

1. Une première partie du fichier (**.arff*) énumère l'ensemble des traits linguistiques utiles à décrire chaque instance d'apprentissage. Pour chacun d'entre eux, on crée une nouvelle ligne dans le fichier, qui doit débuter par le mot-clé *@attribute*, suivi du nom de l'attribut puis des différentes valeurs qu'il peut prendre. La figure 4.2 illustre ce format de représentation, et on y observe que la dernière ligne renseignée correspond à un attribut particulier : la classe.
2. La suite du fichier correspond aux instances du corpus d'apprentissage, c'est-à-dire des exemples de données dont la classes est connue. Dans le format **.arff*, une ligne correspond à une instance, et chacune d'entre elles est décrite par autant de valeurs qu'il y a d'attributs : plus précisément, pour chaque attribut, une valeur est associée à l'instance, et chacune de ces valeurs est séparée de la suivante par une virgule. La dernière valeur renseignée est celle de la classe à laquelle appartient l'instance, en rouge sur la figure 4.3.

Pour parvenir à un tel format, il nous faut d'abord rechercher des instances, c'est-à-dire des paires d'expressions pour lesquelles il faut récupérer les valeurs de chaque attribut, puis les structurer en format (**.arff*) avant de les fournir à Weka.

4.2.1.1 Recherche des paires de mentions

Les éléments *<relation>* du corpus ANCOR correspondent à des paires de mentions coréférentes, mais comme l'illustre la figure 4.1, c'est une autre représentation des paires que nous recherchons, à la fois pour rester fidèle à celle décrite dans les systèmes de l'état de l'art, et pour éviter les biais lors de l'apprentissage, notamment liés aux traits de distance, qui nous paraissent moins pertinents pour décrire une paire reliant un maillon

```
  Soit deux chaînes de coréférence au format ANCOR :
chaîne1 = [[1,4], [1,5], [1,7], [1,10]]
chaîne2 = [[2,3], [2,6], [2,8], [2,9]]

Initialisation d'un tableau <clé - liste de valeurs>, TAB

Pour chaque chaîne :
     Pour chaque paire :
          m1 = premier maillon
          m2 = reprise
          Si m1 n'est pas une clé de TAB :
               On ajoute à TAB un élément de clé m1 en initialisant
               sa liste de valeurs à [m1, m2]
          Si m1 est déjà une clé de TAB :
                On ajoute m2 à sa liste de valeurs

En fin d'itération, la table TAB est celle-ci :
TAB = [1=[1,4,5,7,10], 2=[2,3,6,8,9]]

Initialisation d'une liste de listes listePaires
Pour chaque liste de valeurs val de TAB :
      Pour chaque index i de val :
          Création d'une liste à deux éléments [val[i], val[i+1]];

listePaires = [[1,4], [4,5], [5,7], [7,10], [2,3], [3,6], [6,8], [8,9]]
```

FIGURE 4.4 – Exemple de conversion de format de représentation de relations
(Dans le tableau *TAB*, nous faisons le choix de conserver comme premier élément de la liste de valeurs celle qui constitue la clé, afin de faciliter le calcul suivant en opérant une concaténation itérative sur chaque élément et son suivant pour constituer les paires)

de chaîne à son premier maillon que pour décrire celle reliant deux maillons successifs. Ainsi, nous adaptons les données extraites du corpus ANCOR, pour constituer les paires de mentions positives dont nous avons besoin : la figure 4.4 décrit par un exemple la manière dont nous procédons à cette transposition.

A ces instances positives, doivent s'ajouter des négatives pour que le corpus d'apprentissage soit représentatif du phénomène. C'est en section 4.2.2.2 que nous décrirons la manière dont nous opérons cette sélection.

4.2.1.2 Récupération des valeurs de trait

Afin de compléter les données d'apprentissage, chacune des paires de mentions générées doit être définie selon les traits linguistiques décrits dans le tableau 4.3. Certaines informations sont directement extraites à partir des métadonnées de chacune des deux mentions (c'est le cas des traits 1, 2, 3, 4, 5, 6, 7, 8, 15, 16, 17 et 18), comme l'illustre la figure 3.1 (p. 11) décrivant l'annotation d'une unité dans le corpus ANCOR. De ces annotations, certaines comparaisons de valeurs sont simples à récupérer, pour ainsi compléter les traits 23, 24, 25, 26, 27 et 36.

Concernant les traits contextuels 9, 10, 11 et 12 décrivant les tokens précédent et suivant de chaque mention, nous opérons un parcours automatique de l'arbre xml associé à une requête Xpath récupérant les nœuds textuels avant et après l'unité souhaitée, puis une tokenisation du résultat de cette requête agit comme un filtre pour ne conserver que les deux tokens directement voisins. Les traits 33 et 34 sont ensuite obtenus par comparaison de ces valeurs entre les deux mentions d'une paire.

L'ensemble de features compte un autre nœud textuel, qui est celui décrivant le locuteur d'une mention. Pour le récupérer, il suffit d'accéder, via une requête Xpath, à l'élément *<Turn>* parent de la mention dont on souhaite récupérer l'information. Comme illustré en figure 3.3 (p.12), cet élément parent contient un attribut *speaker* dont la valeur est celle que nous récupérons pour les traits 13 et 14. L'opération peut s'avérer plus complexe lorsqu'il y a un chevauchement de parole, puisque l'annotation de Transcriber décrit alors sous un même tour de parole l'intervention de chacun des deux locuteurs concernés, dont les deux identifiants constituent la valeur de l'attribut *speaker* de ce tour (c'est le cas du premier *<Turn>* de la figure 3.3). Lorsque ces attributs sont récupérés pour chacune des deux mentions d'une paire, elles sont comparées pour calculer la valeur du 35^{e} trait du tableau.

Les derniers attributs non résolus sont ceux relevant de la distance spatiale entre les deux mentions d'une paire [m_1, m_2]. Celles concernant le nombre de caractères et de mots séparant les deux sont récupérées grâce à une requête Xpath recherchant tous les nœuds textuels suivant m_1 et précédant m_2. Le résultat de cette requête est uns chaîne de caractères, dont la longueur correspond à la valeur du nombre de caractères séparant m_1 de m_2. Puis de la même manière que pour les attributs contextuels, une tokenisation de cette chaîne de caractères nous permet d'accéder au nombre de mots les séparant. Les deux autres attributs de distance, en nombre de tours de parole et en nombre de mentions, s'obtiennent après une pré-annotation des éléments xml utiles au calcul, à savoir *<Turn>* et *<anchor>* : pour chaque type d'élément respectif, un compteur est initialisé à 1 puis incrémenté à chaque nouvel élément de même type auquel on ajoute un nouvel attribut *nb* dont la valeur est celle du compteur. Ainsi, l'attribut du premier élément sera 1, celui du deuxième sera 2, etc., jusqu'au dernier élément dont la valeur de *nb* sera égale au nombre total d'éléments dans le fichier (cf. Annexe A.5). C'est à partir de ces annotations que nous calculons, d'une part la distance en nombre de mentions, en soustrayant les valeurs

d'attribut *nb* de m_2 et de m_1, d'autre part la distance en nombre de tours de parole en opérant de la même façon sur les éléments parents *<Turn>* de chacune des mentions.

Avec ces différentes méthodes de récupération de valeurs d'attributs, nous disposons de toutes les données nécessaires à la création d'un corpus d'apprentissage. Avant de le générer au format **.arff*, nous passons par une représentation intermédiaire au format xml, plus facile à gérer du fait de sa forte structuration (cf. Annexe A.8). Ainsi, chacun des 452 fichiers du corpus (hormis un dont les données seront celles de test) est parcouru pour générer en sortie un fichier xml décrivant les instances positives et négatives qu'il contient.

4.2.1.3 Adaptation du format

Nous disposons à ce stade de 451 fichiers xml décrivant des paires de mentions coréférentes et, qu'il nous faut convertir au format **.arff* pour les fournir aux méthodes de l'API Weka. Après avoir créé un nouveau fichier et y avoir intégré automatiquement un en-tête décrivant la liste des attributs telle qu'illustrée en figure 4.2, l'opération consiste à concaténer à la suite de l'en-tête les données de chacun des fichiers, en transposant leur format xml vers le format tabulaire décrit en figure 4.3. Globalement, pour chaque instance est créée une nouvelle ligne, et pour chacun de ses attributs est inscrite sa valeur, suivi d'une virgule, jusqu'à la valeur de la classe qui signale la fin du traitement et le passage à l'instance suivante. La taille de ce corpus d'apprentissage peut être assez considérable en fonction du nombre d'instances renseignées, qui varie selon l'échantillonnage initial des données que nous décrivons dans la section suivante.

4.2.2 Echantillonnage des données

En classification supervisée, un échantillon correspond à un ensemble de données d'entraînement pour une classe donnée, et l'échantillonnage permet de sélectionner parmi un grand ensemble de données celles qui serviront à l'apprentissage du modèle. Dans notre cas, il s'agit non pas de sélectionner des données mais plutôt d'en générer, car nous ne disposons pas à ce stade d'instances négatives à intégrer aux corpus d'apprentissage.

4.2.2.1 Equilibre des classes

En ramenant la résolution de la coréférence à une tâche de classification binaire, se pose une question concernant les instances du corpus d'apprentissage : en effet à partir d'un texte annoté en unités référentielles, quels sont les moyens de choisir les paires de mentions à fournir à l'algorithme de calcul ? Faut-il sélectionner toutes les paires possibles (en récupérant, pour chaque mention, toutes les précédentes) ou seulement une sous-partie ?

Pour rappel, nous considérons dans ce travail qu'une paire de mentions est coréférente uniquement si ces deux mentions constituent deux maillons successifs d'une même chaîne. Ce parti pris est également celui décrit dans (Soon et al., 2001), alors que Marta Recasens, dans sa thèse (2010), considère que deux maillons d'une même chaîne coréfèrent, même s'ils ne sont pas linéairement successifs. Suivant notre approche, si l'on décide de générer toutes les paires possibles pour un texte donné, il y aura inévitablement beaucoup plus d'instances négatives que de positives, et fournir de telles données à un algorithme d'apprentissage favoriserait la classe sur-représentée des négatifs. Nous devons donc trouver une autre stratégie pour équilibrer au mieux les échantillons représentant chacune des deux classes.

Soit S une suite d'expressions référentielles dans un texte

$$S = [A_1, b, c, d, A_2, e, A_3]$$

Soit A la chaîne de coréférence composée des maillons A_1, A_2 et A_3
A partir de l'instance positive $[A_1, A_2]$, sont générées trois instances négatives

$$[b, A_2], [c, A_2] \text{ et } [d, A_2]$$

FIGURE 4.5 – Exemple de génération d'instances négatives

4.2.2.2 Génération de l'ensemble des négatifs

Les instances positives sont récupérées de la manière décrite en section 4.2.1.1, et la question qui se pose à présent est de savoir comment leur ajouter des instances négatives pour constituer un ensemble d'apprentissage pertinent. (Soon et al., 2001), dont la méthode est reprise par (Uryupina, 2004), propose de créer des paires négatives en récupérant pour chaque paire positive $[m_1, m_2]$, toutes les mentions x_i situées entre ses deux maillons, pour construire les paires $[x_i, m_2]$ (cf. illustration en 4.5). Adapté à nos données, ce procédé présente le principal inconvénient d'être coûteux en temps, particulièrement lorsque deux mentions d'une paire positive sont séparées par un grand nombre de mentions (plusieurs instances positives de notre corpus sont par exemple composées de deux mentions séparées par 50 autres). Sachant que pour chaque paire, un temps est déjà nécessaire au calcul de chacun des attributs, envisager de multiplier ce temps par autant de paires présentes dans les 451 fichiers nous incite à réfléchir à une solution alternative.

En s'inspirant de la précédente méthode, nous constituons trois ensembles d'apprentissage qui se distinguent par leurs répartitions d'instances positives et négatives. Techniquement, sont récupérées pour chaque instance positive respectivement une, deux et trois instances négatives, au lieu de toutes, et ce de façon aléatoire. Voici finalement la distribution de chacun de ces ensembles :

1. *small_ trainingSet* : 43170 instances coréférentes et 30935 non coréférentes ;
2. *medium_ trainingSet* : 43516 instances coréférentes et 55599 non coréférentes ;
3. *big_ trainingSet* : 43434 instances coréférentes et 75790 non coréférentes.

Chapitre 5

Expérimentations

Cette section détaille le protocole mis en place pour rechercher le meilleur modèle de résolution de la coréférence possible. Parmi les algorithmes proposés par Weka, aucun ne semble à priori meilleur qu'un autre pour cette tâche, et ce sont différents tests qui nous permettront d'évaluer la performance de chacun. De la même manière, différentes représentations des données seront testées, pour finalement comparer les résultats obtenus par différentes combinaisons de paramètres sur de nouvelles données. Ce que nous construisons ici sont des modèles de classification, et non des systèmes globaux de détection automatique de chaînes de coréférence. Il ne s'agit donc pas de rechercher, à partir d'un texte brut, les expressions référentielles puis de chercher à classifier chaque paire qu'elles forment. Nous supposons disposer de données préalablement annotées en unités référentielles, à partir desquelles nous opérons une classification.

5.1 Définition de l'ensemble de test

Afin d'évaluer l'impact des relations entre données de test et d'apprentissage en classification supervisée, nous construisons différentes versions de l'ensemble de données à tester, selon différentes méthodes. Toutes sont générées à partir des données d'un même fichier, celles qui n'ont pas été intégrées à la phase d'apprentissage : un premier ensemble de test est celui que Marta Recasens décrit dans sa thèse (2010) comme l'ensemble « représentatif », puisqu'il contient toutes les paires possibles d'un texte. Elle ajoute à celui-ci un second ensemble qu'elle qualifie d'« équilibré », puisqu'il est construit en sous-représentant la classe des négatifs de la même manière que lors de la création des ensembles d'apprentissage (dans sa thèse, il s'agit de récupérer 5 mentions entre deux mentions coréférentes). Nous l'imitons dans ce choix, bien que les résultats des modèles sur cet ensemble seront à nuancer du fait que nous considérons dans ce cas connaître à l'avance la classe de chaque instance. Enfin, un dernier ensemble de test s'inspire des travaux de (Denis, 2007) qui introduit la notion de fenêtre de recherche en récupérant, pour former ses paires, toutes les mentions situées dans les 10 phrases précédant une mention. En s'inspirant de cette méthode, nous construisons ce nouvel ensemble en récupérant, pour une mention donnée, les 20 précédentes mentions (en envisageant d'ores et déjà observer une baisse du rappel, du fait que certaines paires coréférentes ne seront pas représentées dans cet ensemble).

Voici donc les trois ensembles de données qui serviront à tester les modèles :

1. *testSet_ all* : 75 instances coréférentes et 6805 non coréférentes
2. *testSet_ balanced* : 75 instances coréférentes et 106 non coréférentes
3. *testSet_ window20* : 75 instances coréférentes et 1389 non coréférentes

5.2 Variation des paramètres

En plus de l'échantillonnage de données, d'autres facteurs sont à faire varier pour espérer obtenir l'association de paramètres qui proposera les meilleurs résultats de classification. Ces facteurs peuvent relever d'une part de la représentation de données, comme c'est le cas de l'échantillonnage, mais également celui de l'ensemble d'attributs qui les décrit, d'autre part de la méthode de calcul du modèle.

5.2.1 Algorithmes d'apprentissage

5.2.1.1 Arbre de décision

Ce type de classifieur apprend à partir de données d'apprentissage un « arbre » dont les nœuds représentent des critères, portant chacun sur un unique attribut de l'ensemble, et où chaque feuille correspond à une classe. L'initialisation de l'algorithme se fait sur un arbre vide (dont l'unique nœud est la racine), puis une itération débute, qui vérifie si le nœud courant est une feuille ou non : si c'est le cas, une classe lui est associée ; sinon, le programme sélectionne un test et crée un sous-arbre dont les branches sont les différentes réponses de ce test. L'algorithme est répété sur le même schéma, et ce jusqu'à ce que toutes les feuilles soient des classes. La sélection du nœud racine se fait par comparaison des mesures d'entropie des classes pour chacun des attributs : on recherche celui dont l'entropie est la plus faible pour débuter l'arbre (*i.e.* celui qui répartit le mieux les classes).

Parmi les programmes proposés par Weka en terme d'arbre de décision, c'est l'algorithme C4.5 (dit J48 dans Weka) que nous utilisons pour nos tests. C'était également l'un des choix de précédents travaux tels que ceux de (Soon et al., 2001) ou (Ng and Cardie, 2002b).

5.2.1.2 Naive Bayes

Il s'agit cette fois d'un modèle représentant des variables aléatoires, fondé sur le théorème des probabilités conditionnelles de Bayes. Adapté à notre problématique, cet algorithme permet de calculer la probabilité pour une instance d'appartenir à la classe des positifs, et celle d'appartenir à celle des négatifs, en connaissant ses attributs. Chacune de ces probabilités est calculée par approximation naïve, et la plus élevée des deux déterminera la classe de l'instance considérée. Nous intégrons ce nouvel algorithme à nos expérimentations pour tester son efficacité sur nos données, de la même manière que dans les travaux de (Kobdani and Schütze, 2010).

5.2.1.3 SVM

Abréviation de *Support Vector Machine*, ou parfois *Séparateur à Vaste Marge* en français, le concept de cette méthode est de séparer l'espace dans lequel sont représentées les données en deux parties pour en délimiter deux sous-espaces. Si différents hyperplans [1] peuvent partager ce plan en deux, on sélectionne celui qui a la meilleure marge (*i.e.* celui à équidistance des points les plus proches des deux ensembles à séparer). Afin de simplifier et d'accélérer les calculs, intervient l'astuce du noyau : ici, nous utilisons le noyau par défaut intégré dans Weka, et l'astuce est que les calculs du SVM s'opèrent sur ce noyau, qui ne dépend pas des dimensions décrivant les données (qui peuvent donc être en très grand nombre sans influencer le temps de calcul). Un SVM est en fait un hyperplan, appelé ainsi du fait qu'il est déterminé par des vecteurs supports, qui sont les plus proches de la frontière entre deux classes.

1. Dans un espace à n dimensions, un hyperplan est un objet de dimension n-1 qui sépare donc l'espace en deux.

	Traits	Définition	Valeurs possibles
1	ID_FORM	Identité des formes de m_1 et m_2	{YES, NO, NA}
2	ID_SUBFORM	Identité d'une sous-forme	{YES, NO, NA}
3	INCL_RATE	Taux d'inclusion des tokens	REAL
4	COM_RATE	Taux de tokens communs	REAL
5	DISTANCE_MENTION	Distance en nombre de mention	REAL
6	DISTANCE_TURN	Distance en nombre de tours de parole	REAL
7	DISTANCE_WORD	Distance en nombre de mots	REAL
8	DISTANCE_CHAR	Distance en nombre de caractères	REAL
9	EMBEDDED	Enchâssement de m_2 dans m_1	{YES, NO, NA}
10	ID_PREVIOUS	Identité du token précédent	{YES, NO, NA}
11	ID_NEXT	Identité du token suivant	{YES, NO, NA}
12	ID_SPK	Identité du locuteur	{YES, NO, NA}

TABLE 5.1 – Sous-ensemble d'attributs détectables automatiquement

5.2.2 Ensemble de traits

La sélection d'attribut est une étape importante en classification automatique, puisqu'elle permet de distinguer les facteurs pertinents de ceux qui ne le sont pas, ainsi que d'éviter les redondances pour faciliter l'apprentissage. L'ensemble de features idéal est celui dont la taille est minimale mais suffisante pour décrire le concept analysé, le problème est que trouver un tel sous-ensembles à partir de l'ensemble initial. C'est une tâche complexe qui peut être automatisée via des programmes de comparaison de résultats de différents sous-ensemble, ou qui peut être guidée manuellement notamment par la lecture d'arbres de décision. A l'issue d'une première série d'expériences, plusieurs arbres pourront être analysés afin de déduire la pertinence de certains attributs.

Faire varier l'ensemble de trait initial nous est également utile dans la perspective d'un système *end-to-end* qui prendrait en entrée un texte brut. Dans ce cas en effet, aucune métadonnée n'existe pour orienter la recherche des attributs, il s'agit de pré-traiter les données en les soumettant à divers outils de TAL pour retrouver les valeurs de chacun des traits. Or l'ensemble de traits que nous avons constitué contient certaines informations qu'il sera difficile, voire impossible, de récupérer. Nous envisageons donc de construire différents modèles appris sur des données dont les attributs peuvent être retrouvés automatiquement, sans recourir à des outils externes tels qu'un détecteur d'entités nommées susceptible d'intégrer des erreurs dans les données. C'est donc un sous-ensemble de 12 attributs que nous conservons pour ce test (Table 5.1).

Notons avant de poursuivre que les attributs de type string ne servent finalement pas à décrire les données qui seront utilisées dans nos tests. Le fait est que les algorithmes de Weka ne prennent pas en compte ce type d'attribut, mais l'API propose en revanche de vectoriser ces chaînes de caractères en sélectionnant chaque token de la chaîne pour en faire un nouvel attribut dont la valeur est un entier, pour ensuite décrire chaque instance en fonction des tokens qui la composent. Ce procédé augmenterait considérablement la taille des ensembles de données (il y aurait autant d'attributs en plus que de tokens différents dans le corpus), et augmenterait le temps de calcul du modèle. De plus, mis à part les pronoms qui constituent un ensemble fini d'éléments, il semble difficile d'imaginer que les chaînes de caractères puissent fournir des indices suffisamment homogènes pour

Référence \ Système	C	N
C	CC	CN
N	NC	NN

C = COREF
N = NOT_COREF

COREF			NOT_COREF		
$precision_c$	=	$\frac{CC}{CC+NC}$	$precision_n$	=	$\frac{CN}{CC+NC}$
$rappel_c$	=	$\frac{CC}{CC+CN}$	$precision_n$	=	$\frac{CN\ N}{CC+CN}$
$F-mesure_c$	=	$2\ \frac{precision_p.rappel_p}{precision_p+rappel_p}$	$F-mesure_n$	=	$2\ \frac{precision_n.rappel_n}{precision_n+rappel_n}$

FIGURE 5.1 – Illustration d'une matrice de confusion et des mesures associées à sa lecture

guider l'algorithme dans ses choix de classification.

5.3 Métriques d'évaluation

5.3.1 Métriques classiques

Une tâche telle que la classification supervisée est traditionnellement évaluée selon des mesures classiques de précision et de rappel, pour lesquelles chaque classe a une valeur. Pour une classe donnée, la précision représente la proportion de réponses pertinentes parmi toutes celles retrouvées par le système, tandis que le rapport représente la proportion de réponses retournées par le système parmi toutes celles qui auraient du être retournées. Une autre métrique, la F-mesure, se calcule par la moyenne harmonique des deux précédentes. Dans le vocabulaire de fouille de données, les résultats sont définis d'une part du point de vue du système (positif ou négatif), d'autre part du point de vue de la référence (vrai ou faux). Ainsi, l'ensemble des résultats peuvent être décrits en terme de vrais positifs (VP), vrais négatifs (VN), faux positifs (FP) et faux négatifs (FN) pour chacune des classes représentées (cf. matrice 5.1).

Ces métriques vont nous permettre d'évaluer la qualité du modèle de résolution de l'anaphore, pour voir avec quel degré de pertinence il est capable de déterminer la classe d'une paire de mentions. Or ce que nous souhaitons évaluer est la qualité de la résolution de la coréférence, qui doit comparer l'ensemble de chaînes de coréférence retournées par le système à celui des vraies chaînes d'un texte. Il existe pour ce faire des métriques spécialisées décrites dans la section suivante, pour lesquelles il nous faut au préalable reconstituer les chaînes de coréférence à partir des résultats du modèles de classification. Nous évoquions précédemment le caractère transitif des chaînes de coréférence, sur lequel nous nous basons pour cette tâche : parmi toutes les paires classées comme coréférentes par le système, nous regroupons sous un même ensemble toutes celles qui ont au moins une mention en commun. A l'issue de ce traitement, nous disposons d'un ensemble de chaînes de coréférence qui sera fourni, avec l'ensemble des vraies chaînes, aux programmes d'évaluation (Nous appellerons désormais T l'ensemble des vraies chaînes, et S l'ensemble de celles prédites par le système).

5.3.2 Métriques spécifiques à la tâche

5.3.2.1 MUC

Cette métrique (dont le nom est issu de la campagne d'évaluation *Message Understanding Conference*) s'intéresse à l'évaluation des liens de coréférence qui sont communs à S et T. Précisément, le rappel correspond au rapport entre le nombre de liens communs à S et T et le nombre total de liens dans T, et la précision au rapport entre le nombre de liens communs aux deux et le nombre total de liens dans S : les suppressions sont ainsi pénalisées par le rappel, et les insertions par la précision. Cette prise en compte individuelle de chaque erreur est le principal point faible de cette mesure, qui ne pénalisera pas par exemple un système dont l'ensemble final sera composé d'une unique longue chaîne comprenant toutes les mentions : par exemple, deux ensembles $T = \{\{m_1,m_2,m_3,m_6\}, \{m_4,m_5,m_7\}\}$ et $S = \{\{m_1,m_2,m_3,m_4,m_5,m_6,m_7\}\}$ ne diffèrent que par l'ajout d'un seul lien dans l'ensemble prédit par le système. Le résultat sera donc très bon, alors que pour une interprétation humaine, l'erreur est bien plus importante. S'ajoute à ce problème le fait que la prise en compte des liens comme élément du calcul ne permet pas de considérer les singletons.

5.3.2.2 B^3

C'est pour pallier aux faiblesses de MUC qu'a été élaborée cette deuxième métrique, dont l'unité de base est la mention plutôt que le lien. Ainsi pour chaque mention m d'un texte, on considère S_m comme la chaîne du système contenant m et T_m la chaîne de coréférence contenant m. Le rappel est alors le rapport entre le nombre de mentions communes à S_m et T_m et le nombre total de mentions de T_m, et la précision à l'inverse est le rapport de l'intersection des deux ensembles et du nombre total de mentions dans S_m. Les résultats proposés par B^3 sont significativement plus représentatifs de la qualité du système évalué, d'une part parce que les singletons sont pris en compte (et ils constituent dans notre cas la plus grande partie des données de test), d'autre part parce qu'un système qui renverrait une unique chaîne finale serait cette fois sévèrement sanctionnée.

5.3.2.3 CEAF

Une autre stratégie d'évaluation, développée dans les travaux de (Luo, 2005), est basée sur l'entité, c'est-à-dire la référence commune à tous les maillons d'une chaîne de coréférence (le nom complet de la mesure est d'ailleurs *Constrained Entity Aligned F-Measure*). La méthode de calcul est cette fois plus complexe, et surtout bien plus longue, puisqu'il s'agit de trouver, pour chaque chaîne de S, la chaîne de T la plus proche. Cette sélection s'opère par mesure de similarité entre les deux chaînes (*i.e.* le nombre de mentions communes), sachant que dès qu'une chaîne de S a été associée à une chaîne de T, elle ne peut plus être associée à aucune autre (une chaîne du système correspond donc à une unique chaîne de référence). Une première partie de l'algorithme de calcul recherche donc les meilleures correspondances possibles entre chaînes prédites et chaînes de coréférence, puis une seconde phase détermine, à partir de ces correspondances, quelle est la meilleure correspondance globale de l'ensemble de toutes les chaînes.

Malgré la pertinence d'une telle évaluation, l'implémentation de CEAF devient rapidement complexe puisqu'elle teste toutes les associations possibles entre les chaînes prédites et celles de référence, et le temps de calcul peut s'avérer très long. A cause de ces contraintes de temps de calcul, nous n'avons pu, dans le cadre de ce stage, évaluer nos résultats pour cette métrique.

		COREF			NOT_COREF		
		P	R	F_{mesure}	P	R	F_{mesure}
small_trainingSet	SVM	82.14	92.62	87.06	87.47	71.89	78.92
	J48	87.71	89.77	88.73	85.25	82.45	83.83
	NaiveBayes	88.76	46.52	61.05	55.15	91.78	68.90

TABLE 5.2 – Résultats des différents modèles de classification en terme de précision, rappel et F-mesure pour chacune des deux classes, COREF et NOT_COREF

5.3.2.4 BLANC

Définie comme *BiLateral Assessment of Noun-phrase Coreference*, il s'agit de la métrique la plus récente mise au point dans les travaux de (Recasens, 2010), dont la vocation est de considérer conjointement les liens de coréférence et ceux de non-coréférence. Concrètement, deux paramètres sont pris en compte avec d'une part, les réponses du système pour une paire, et d'autre part, les réponses de référence : ces deux paramètres ayant chacun deux valeurs possibles (positif ou négatif pour le premier, vrai ou faux pour le second), l'ensemble forme une matrice de confusion sur laquelle s'appuie le calcul de BLANC, dont la formule de F-mesure est en fait la moyenne des F-mesure des deux classes. La pertinence de cette mesure tient à la prise en compte de la sur-représentation des instances négatives par rapport aux positives dans un corpus, et met en évidence la capacité du système évalué à caractériser ces instances négatives comme telles.

5.4 Série d'expériences

5.4.1 Évaluation de la classification

Rappelons que la première étape de notre système de résolution de la coréférence consiste en une classification binaire des paires de mentions. Nous testons les trois différents algorithmes présentés sur le corpus *small_trainingSet*, en validation croisée. Cette méthode d'évaluation permet de découper un ensemble de données en n plis (en l'occurrence, nous choisissons n=5) et de réaliser n expériences avec à chaque fois *n-1* plis utilisés pour l'apprentissage et le *n-ième* pour le test. C'est la moyenne des n évaluations qui mesure la qualité globale du modèle construit. Le tableau 5.2 présente les résultats obtenus selon les différents programmes d'apprentissage sur les deux classes COREF et NOT_COREF.

C'est l'arbre de décision qui semble obtenir les meilleures performances , puisque les F-mesure de chacune des deux classes sont les meilleures parmi les trois. Cette observation se traduit par le fait que le modèle généré par cet algorithme reconnaît aussi bien les instances coréférentes que les non-coréférentes, sans privilégier une classe particulière. Les résultats du modèle de NaiveBayes démontrent au contraire qu'il classe préférentiellement les instances en NOT_COREF, puisque le rappel de cette classe est excellent (le meilleur

Soit T l'ensemble des vraies chaînes tel que
$T = \{\{m_1,m_2,m_3,m_6\}, \{m_4,m_5,m_7\}\}$

Soit l'ensemble de test contenant toutes les paires possibles, tel que

$[m_1,m_2]$

$[m_1,m_3]$ $[m_2,m_3]$

$[m_1,m_4]$ $[m_2,m_4]$ $[m_3,m_4]$

$[m_1,m_5]$ $[m_2,m_5]$ $[m_3,m_5]$ $[m_4,m_5]$

$[m_1,m_6]$ $[m_2,m_6]$ $[m_3,m_6]$ $[m_4,m_6]$ $[m_5,m_6]$

$[m_1,m_7]$ $[m_2,m_7]$ $[m_3,m_7]$ $[m_4,m_7]$ $[m_5,m_7]$ $[m_6,m_7]$

Les paires négatives en rouge sont susceptibles d'entraîner la fusion des deux chaînes si elles sont considérées comme coréférentes par le classifieur, puisque la règle de transitivité réunit sous la même chaîne de coréférence deux instances de la classe COREF ayant une mention en commun.

FIGURE 5.2 – Illustration d'une fusion de chaînes en conséquence à une mauvaise classification en positif

parmi les trois), et parallèlement celui de la classe COREF est plutôt mauvais. En plus de ce rappel, la précision de la classe COREF pour ce même modèle est la meilleure, ce qui conforte la précédente idée que NaiveBayes a tendance à prédire la classe majoritaire en négligeant le rappel de la classe sous-représentée. Quant aux résultats de SVM, ils se rapprochent plus de ceux de J48 puisqu'ils paraissent équilibrés entre les deux classes et selon les trois différentes mesures : ils restent néanmoins un peu moins bons que ceux de l'arbre de décision.

5.4.2 Évaluation des chaînes de coréférence

Ce que nous souhaitons comparer maintenant sont les performances de différents systèmes de détection des chaînes de coréférence, qui varient en fonction de différents paramètres mais qui sont évalués sur les mêmes données. A l'issue de ces comparaisons, nous pourrons mettre en relation nos propres résultats avec ceux des systèmes de l'état de l'art, bien qu'il faille rester prudent quant au fait que la langue étudiée soit dans notre cas du français, oral qui plus est.

5.4.2.1 Variation d'algorithme et d'échantillonnage

Cette première série de test combine différents facteurs de variation puisque nous testons neuf systèmes d'apprentissage (chacun des trois algorithmes est appris respectivement sur les trois ensembles d'apprentissage) sur les trois ensembles de test, et ce selon les trois métriques spécifiques à notre tâche. Afin d'analyser précisément ces résultats, il est utile de connaître la distribution des mentions et des chaînes dans les données d'apprentissage et dans celles de test : rappelons que ce qui change dans chacune des versions des corpus (d'apprentissage et de test) sont les instances non-coréférentes, c'est-à-dire que dans tous les cas, les vraies chaînes de coréférence sont les mêmes. Le tableau 5.4 décrit la répartition de ces chaînes en fonction du nombre de mentions qu'elles contiennent, pour ces deux ensembles. Nous observons une nette prépondérance des singletons, dans les deux ensembles de données, ce qui influencera certainement les résultats, notamment l'incapacité de MUC à prendre en compte ce type de données.

		testSet_ all			*testSet_ balanced*			*testSet_ window20*		
		MUC	B^3	BLANC	MUC	B^3	BLANC	MUC	B^3	BLANC
small_ trainingSet	SVM	65.22	25.06	43.74	83.72	72.15	75.60	71.43	43.24	47.74
	J48	66.96	30.38	45.39	82.21	75.52	80.06	72.46	43.91	49.34
	NaiveBayes	72.92	54.40	49.42	53.21	76.88	66.39	62.22	74.77	63.62
medium_ trainingSet	SVM	64.93	24.14	46.14	84.15	76.33	79.55	71.50	46.05	50.73
	J48	67.58	34.51	49.37	**81.33**	**79.83**	**80.96**	74.49	55.00	54.36
	NaiveBayes	75.82	60.75	52.00	53.21	76.88	66.39	62.22	74.77	63.62
big_ trainingSet	SVM	64.93	24.14	48.09	77.32	78.71	75.56	70.30	50.78	52.63
	J48	67.89	35.33	49.50	79.45	79.62	80.19	72.92	58.00	55.34
	NaiveBayes	**75.70**	**63.43**	**54.70**	53.21	76.88	66.39	**62.22**	**74.77**	**63.62**

TABLE 5.3 – Résultats des tests selon les métriques MUC, B^3 et BLANC, en fonction des ensembles d'apprentissage et de test et des algorithmes d'apprentissage (les 3 métriques sont ici résumées à leur F-mesure)

	Ensemble d'apprentissage		Ensemble de test	
	#	%	#	%
Mentions	102608	100	174	100
Entités	58735	100	99	100
Singletons	48597	82.87	81	81.81
2-mentions	4154	7.07	10	10.10
3-mentions	1852	3.15	2	2.02
4-mentions	1020	1.73	1	1.01
5-10-mentions	2044	3.48	2	2.02
>10-mentions	1068	1.81	3	3.03

TABLE 5.4 – Distribution des données dans les ensembles d'apprentissage et de référence

Le tableau 5.3 présente les résultats de ces différents tests, desquels nous pouvons tirer plusieurs conclusions. La première concerne les résultats sur l'ensemble de test *testSet_balanced*, dont les meilleurs sont obtenus avec l'algorithme J48 appris sur les données de *medium_trainingSet* (en bleu dans le tableau), ce qui rejoint les précédents résultats de classification. En supposant connaître les classes pour générer cet ensemble, nous avions introduit un biais dans les données en ne conservant que celles se rapprochant le plus des exemples d'apprentissage : les résultats sont meilleurs du fait que peu d'instances négatives sont soumises au classifieur, ce qui réduit considérablement les risques d'intégrer de faux liens de coréférence lors de la reconstitution des chaînes par transitivité. Ce risque est d'ailleurs celui qui fait chuter la plupart des résultats des autres tests, puisqu'on observe pour la plupart des systèmes que l'ensemble des chaînes prédites contient une très longue chaîne contenant la moitié des mentions du texte. Le problème est qu'une seule paire négative classée en positive, entraîne lors de la reconstitution des chaînes la fusion de deux chaînes de coréférence normalement distinctes (cf figure 5.2). Or en augmentant le nombre d'instances négatives dans l'ensemble de test, on augmente également le risque de générer ce type d'erreur, d'où l'observation fréquente d'une très longue chaîne dans les résultats. Néanmoins, on observe dans le tableau que ce facteur d'erreur n'influe pas aussi négativement sur les résultats de MUC que sur les deux autres, comme nous l'avions supposé lors de la description de cette métrique.

Le système présentant les meilleurs résultats est celui incluant le modèle de classification appris par l'algorithme NaiveBayes sur le plus grand des trois ensembles d'apprentissage (résultats en rouge dans le tableau). Au vu des précédentes observations, corrélées avec les bons résultats observés pour NaiveBayes en terme de précision pour la classe des coréférents (cf section 5.4.1), ces résultats s'expliquent : la majorité des instances de l'ensemble de test fournies au modèle sont classées par NaiveBayes en NOT_COREF, puisqu'il privilégie cette classe. Grâce à ce système, très peu d'instances non-coréférentes sont mal classées, et donc très peu de faux liens de coréférence sont générés lors de la reconstitution des chaînes. En revanche, le même système obtient un bien moins bon rappel pour cette classe COREF que les autres systèmes incluant SVM ou J48, mais les résultats de BLANC restent malgré tout meilleurs sur les F-mesures de NaiveBayes. Concernant l'ensemble de test, on observe que c'est pour celui généré par récupération des 20 mentions précédentes que les résultats sont les plus hauts.

		COREF			NOT_COREF		
		P	R	F_{mesure}	P	R	F_{mesure}
small_trainingSet	SVM	90.85	46.52	61.53	55.60	93.46	69.72
	J48	71.25	81.09	63.54	68.91	60.92	79.32
	NaiveBayes	88.20	45.12	59.70	54.46	91.58	68.30

TABLE 5.5 – Résultats des modèles de classification appris sur un sous-ensemble d'attributs, en terme de précision, rappel et F-mesure pour chacune des deux classes, COREF et NOT_COREF

5.4.2.2 Variation de l'ensemble d'attributs

Dans cette section, nous effectuons les mêmes tests en faisant varier cette fois-ci les attributs décrivant les données. Dans un premier temps, nous évaluons la qualité des nouveaux modèles de classification appris à partir de l'ensemble d'attributs restreints décrits dans le tableau 5.1, en termes de précision, rappel et F_mesure. Afin de pouvoir comparer ces résultats aux précédents, nous construisions ces modèles à partir du même ensemble d'apprentissage, à savoir *small_trainingSet*. Nous évaluons ensuite trois systèmes de détection des chaînes de coréférence, intégrant chacun l'un des précédents modèles de classification (les tests sont restreints aux deux ensembles *textSet_All* et *textSet_window20*).

Les nouveaux résultats de classification (cf. tableau 5.5) rejoignent les précédents sur le fait que l'arbre de décision obtient toujours les meilleures F-mesure. Les résultats du modèle appris par NaiveBayes sont également similaires, mais ce sont ceux de SVM qui sont particulièrement remarquables : alors que précision et rappel étaient relativement équilibrés dans le précédent modèle appris sur SVM, on observe cette fois-ci la même chose que pour NaiveBayes, à savoir un bon résultat en faveur de la précision pour la classe COREF (au détriment du rappel), et parallèlement un bon résultat en faveur du rappel pour la seconde classe (au détriment de la précision, cette fois). Malgré ces forts déséquilibres entre précision et rappel, ce modèle obtient pour les deux classes de meilleures F-mesure que le modèle appris sur NaiveBayes, ce qui laisse envisager que son intégration dans un système de détection de la coréférence offrira les performances les plus élevées. C'est le tableau 5.6 qui présente les résultats de ces différents systèmes, pour lesquels il est surprenant de constater qu'ils sont majoritairement meilleurs que les mêmes systèmes appris sur des données plus complètes (*i.e.* avec plus d'attributs). Ce sont particulièrement sur les métriques B^3 et BLANC que les évolutions sont remarquables, mais uniquement pour les systèmes intégrant SVM (en effet, tous les résultats de J48 sont en baisse par rapport aux précédents, et ceux de NaiveBayes sont assez proches de ceux obtenus à partir de l'ensemble d'attribut complet). Ainsi, les résultats de B^3 ont respectivement augmenté de 45 et 31 points pour les ensembles de test *testSet_all* et *testSet_window20*, et ceux de BLANC ont augmenté pour ces mêmes ensembles de 13 et 16 points. Ces résultats sont très encourageants quant à la perspective de mettre en place un système complet de détection

		testSet_ all			*testSet_ window20*		
		MUC	B^3	BLANC	MUC	B^3	BLANC
small_ trainingSet	SVM	67.03	60.17	56.19	**63.38**	**74.47**	**63.07**
	J48	63.52	22.25	50.61	65.42	41.18	54.14
	NaiveBayes	73.02	56.66	50.09	62.69	75.10	63.62

TABLE 5.6 – Résultats des tests selon les métriques MUC, B^3 et BLANC, pour les systèmes intégrant les modèles appris sur un ensemble restreint d'attributs (les 3 métriques sont ici résumées à leur F-mesure)

des chaînes de coréférence, pour lequel aucune annotation ne serait fournie en entrée.

5.4.2.3 Interprétation des arbres de décision

Plutôt que d'analyser ici tous les arbres de décision générés par l'ensemble de nos tests, nous décidons d'en comparer deux, appris sur le même ensemble (*small_ trainingSet*) mais qui se distinguent par l'ensemble de traits utilisés pour l'apprentissage (complet pour l'un, restreint pour l'autre). Le plus petit des deux arbres, comprenant le moins d'attributs, atteint une longueur de 221 lignes, pour une profondeur de 18 niveaux, et un total de 123 feuilles. Il n'est donc pas envisageable de décrire chacun des tests qui le composent. Nous nous limitons donc à ne décrire que le haut de l'arbre, en se concentrant uniquement sur les 3 premiers niveaux, qui représentent les attributs les plus discriminants permettant au modèle de classer une instance. Les figures 5.3 et 5.4 décrivent ces nœuds, de la façon suivante :

- La première ligne constitue le nœud racine de l'arbre, puis chaque ligne suivante correspond à l'un de ses descendants, dont le niveau est déterminé par le nombre d'indentations précédant sa déclaration ;
- Si l'attribut permet la classification sans faire de nouveau test, la ligne décrit le nom de l'attribut, sa valeur, la classe des instances qu'il permet de discriminer puis entre parenthèses le nombre d'instances correctement classées sur ce critère, ainsi que celui des instances mal classées sur ce critère ;

  e.g. EMBEDDED = YES : NOT_COREF (770.0/23.0)).
- Si en revanche l'attribut nécessite de nouveaux critères pour déterminer une classe, il n'est déclaré que par son nom et sa valeur.

  e.g. EMBEDDED = NO.

L'arbre construit à partir de tous les attributs (figure 5.3) considère l'identité stricte (ID_FORM) entre deux formes comme le trait séparant le mieux les deux classes : il n'est cependant pas suffisant pour classer directement une instance en COREF, puisque même si sa valeur est YES, d'autres tests sont nécessaires à une classification. Les travaux de (Soon et al., 2001) intégraient également dans leur système un arbre de décision (appris sur 12 traits), dont la racine était elle aussi la correspondance des deux formes. Il est étonnant de constater que pour le second arbre (figure 5.4), appris sur un ensemble restreint d'attributs

```
ID_FORM = YES
|   ID_DEF = YES
|   |   ID_GENRE = YES: COREF (11180.0/49.0)
|   |   ID_GENRE = NO
|   |   ID_GENRE = NA
|   ID_DET = NO
|   |   M2_TYPE = N: COREF (71.0)
|   |   M2_TYPE = P: COREF (0.0)
|   |   M2_TYPE = PR
|  ID_DEF = NA: NOT_COREF (1.0)
ID_FORM = NO
|   ID_GENRE = YES
|   |   M1_EN = NO
|   |   M1_EN = PERS
|   |   M1_EN = FONC
|   |   M1_EN = LOC
|   |   M1_EN = ORG
|   |   M1_EN = PROD
|   |   M1_EN = TIME
|   |   M1_EN = AMOUNT
|   |   M1_EN = EVENT
|   |   M1_EN = NULL: COREF (0.0)
|   |   M1_EN = UNK: COREF (0.0)
|   ID_GENRE = NO
|   |   M2_NEW = YES
|   |   M2_NEW = NO
|   |   M2_NEW = UNK: NOT_COREF (0.0)
|   |   M2_NEW = NULL: NOT_COREF (0.0)
|   ID_GENRE = NA
|   |   ID_EN = YES
|   |   ID_EN = NO: NOT_COREF (540.0/10.0)
|   |   ID_EN = NA
ID_FORM = NA: COREF (0.0)
```

FIGURE 5.3 – Extrait de l'arbre de décision appris sur l'ensemble d'attributs complet (taille de l'arbre entier = 3823 lignes)

```
EMBEDDED = YES: NOT_COREF (770.0/23.0)
EMBEDDED = NO
|   INCL_RATE <= 0
|   |   DISTANCE_WORD <= 0
|   |   DISTANCE_WORD > 0
|   INCL_RATE > 0
|   |   COM_RATE <= 0.25
|   |   COM_RATE > 0.25: COREF (18396.0/717.0)
EMBEDDED = NA: COREF (0.0)
```

FIGURE 5.4 – Extrait de l'arbre de décision appris sur un sous-ensemble d'attributs (taille de l'arbre entier = 221 lignes)

dont ID_FORM, le même trait ne soit plus considéré en tant que racine, et même pas considéré du tout puisqu'à aucun moment il n'est utilisé dans les tests. Sur ce modèle-ci, l'attribut le plus discriminant est EMBEDDED, qui vérifie si les deux mentions d'une paire ne sont pas incluses l'une dans l'autre. Nous avions d'ailleurs formulé l'hypothèse, lors de la description des traits, que celui-ci pourrait être un facteur déterminant lors de la classification, et en effet, le premier test signifie que si deux mentions font partie d'un même syntagme complexe, elle sont directement classées en NOT_COREF.

Nous regardons maintenant les niveaux suivants de l'arbre, et plus précisément ceux à l'issue desquels une classe est attribuée. Il nous paraît intéressant de constater combien d'instances peuvent être classées après seulement trois tests, et pour ce faire, nous comparons les chiffres décrivant le nombre d'instances correctement classées à l'issue d'un test au nombre total d'instances du corpus, soit 74105.
Concernant le premier arbre de décision, contenant plus d'attributs, les observations sont les suivantes :

- Si ID_FORM = YES + ID_DEF = YES + ID_GENRE = YES, alors l'instance appartient à la classe COREF (un peu plus de 15% des instances remplissent ces trois critères, pour lesquelles 99,6% appartiennent effectivement à la classe COREF, contre seulement 0,4% qui appartiennent en fait à celle des NON_COREF).
- Si ID_FORM = YES + ID_DEF = NO + M2_TYPE = N, alors l'instance appartient à la classe COREF (environ 0,1% des instances remplissent ces conditions, et toutes sont effectivement dans la classe COREF).
- Si ID_FORM = YES + ID_GENRE = NA + ID_EN = NO, alors l'instance appartient à la classe NOT_COREF (un peu moins de 1% des instances correspondent à ces critères, et parmi elles 98% sont effectivement non-coréférentes, contre seulement 2% de coréférentes).

Concernant le second arbre, voici les nouveaux cas observés :

- Si EMBEDDED = YES, alors l'instance n'est pas coréférente (sur un peu plus d'1% des instances répondant à ce critères, 97% sont effectivement NON_COREF, contre 3% de COREF).
- Si EMBEDDED = NO + INCL_RATE $>$ 0 + COM_RATE $>$ 0.25, alors l'instance est coréférente (un peu plus d'un quart des instances remplissent ces trois critères, parmi lesquelles 99% sont effectivement classées en COREF, et seulement 1% en NOT_COREF).

Les résultats obtenus jusqu'à présent par différentes séries de test ne sont comparables que dans le cadre restreint de ce travail et de nos données. La section suivante met ces résultats en perspective avec ceux des systèmes de l'état de l'art, afin d'en évaluer les éventuels gain ou perte en terme de performance.

5.4.3 Comparaison aux résultats des systèmes existants

Dans les précédents travaux concernant l'automatisation de la résolution de la coréférence, les évaluations se font sur deux principaux corpus proposés par les deux principales conférences proposant ce type de tâche, MUC et ACE. La première des deux a proposé au cours de ses différentes campagnes d'évaluation un total de 7 différents corpus (de MUC-1 à MUC-7) à partir desquels les systèmes sont testés. Quant à ACE, les systèmes étudiés dans le cadre de ce travail utilisent différentes versions dont la plus récente est ACE2004. Tous les systèmes ne sont pas évalués sur tous les ensembles de tests disponibles, et nous faisons le choix ici de conserver, lorsque les résultats sont nombreux, uniquement ceux appliqués aux corpus les plus récents. Cette diversité de corpus rend compliquées les analyses comparatives des résultats, puisqu'un système peut être bon sur un corpus, et non sur un autre. Nous devrons prendre ce biais en considération lors de la comparaison de nos résultats à ceux des systèmes existants puisque nos données sont différentes, et qui

System	Mention	Corpus	MUC	B^3	CEAF	BLANC
(Soon et al., 2001)	System	MUC-7	60.4			
(Ng and Cardie, 2002b)	System	MUC-7	63.4			
(Yang et al., 2003)	True	MUC-7	60.2			
(Haghighi and Klein, 2007)	True	MUC-6	81.9	75.0	72.0	
(Haghighi and Klein, 2007)	True	ACE-2004	79.6	79.0	73.3	
(Denis, 2007)	True	ACE-2	71.6	72.7	67.0	
(Bengtson and Roth, 2008)	True	ACE-2004	75.1	80.1	75.0	75.6
(Stoyanov et al., 2010)	System	MUC-7	62.8	65.9		
(Stoyanov et al., 2010)	System	ACE-2005	67.4	73.7		
(Longo, 2013)	System	corpus multi-genres [2]	36.0	69.7	54.8	59.5

TABLE 5.7 – Résumé des performances des systèmes de résolution de la coréférence

plus est, les données de test sont issues du même corpus que celles d'apprentissage, et non sur des données dites « réelles ».

Une autre remarque concerne le type de système évalué : il est en effet nécessaire de bien distinguer les systèmes *end-to-end*, qui débutent par une phase de détection des mentions potentiellement référentielles, de ceux dont la tâche est de reconstituer les chaînes de coréférence à partir d'un corpus déjà annoté en mentions. Le degré de difficulté de la tâche est bien plus important dans le premier cas, puisqu'il y a un risque d'erreurs supplémentaire. Dans le tableau comparatif 5.7, on parle de *true mention* et de *system mention*.

En observant les précédents résultats, notre système semble présenter de moins bonnes performances que des systèmes équivalents évalués sur des corpus déjà annotés en mentions. Les meilleurs que nous ayons obtenus après toutes nos séries de tests sont de 63.68 pour MUC, 74.47 pour B^3 et 63.07 pour BLANC : pour cette première métrique, notre score ne dépasse que celui du système de (Yang et al., 2003), intégrant un modèle *twin-candidate*. Par rapport aux systèmes intégrant un modèle de classification binaire basé sur un corpus annoté en mentions, tels que celui de (Bengtson and Roth, 2008), nos résultats sont bien moins bons, puisque l'on note une baisse du résultat de MUC d'environ 12 points par rapport au leur ; même constat pour B^3 qui baisse de 6 points. Concernant les résultats de BLANC, seuls (Bengtson and Roth, 2008) ont intégré cette métrique dans leur procédure d'évaluation, et de nouveau, le score dépasse le nôtre d'environ 12 points. Ces observations sont toutefois à nuancer du fait que notre système n'est pas testé sur les mêmes données que ceux auxquels nous comparons nos résultats. Or les données de test influencent les résultats, comme le démontrent les différentes performances du système de (Stoyanov et al., 2010) : l'évaluation sur ACE-2005 a permis au système de gagner presque 5 points pour MUC et presque 8 pour B^3.

Concernant les performances sur la résolution de la coréférence en français, c'est aux résultats du système de (Longo, 2013) que nous confrontons les nôtres. Sur l'ensemble des trois métriques qui constituent notre évaluation, les résultats de notre système sont

2. Le corpus est constitué de trois types de documents, des romans, des articles de presse et des articles juridiques. Les mesures renseignées correspondent aux moyennes obtenues par le système sur chacun de ces genres.

meilleurs que ceux de (Longo, 2013), mais rappelons que celui-ci est un système *end-to-end* dont les performances peuvent chuter à cause d'un mauvais repérage des expressions référentielles.

Chapitre 6

Conclusions et Perspectives

Depuis que la tâche résolution de la coréférence occupe une place importante dans les problématiques de traitement automatique des langues, beaucoup de travaux se sont attachés à développer des systèmes de détection de chaînes de coréférence pour l'anglais. Très peu cependant, mis à part celui décrit dans (Longo, 2013), ont étudié le phénomène et ses pistes de résolution automatique sur du français, et ce travail présente l'un de ces premiers systèmes appris automatiquement sur un corpus annoté. En plus de cette évolution de langue, c'est une distinction de canal qui caractérise ce travail, puisqu'à ce jour aucun modèle de résolution basé sur l'apprentissage supervisé n'avait été développé spécifiquement pour l'oral, or le corpus annoté dont nous disposons est justement composé de discours oraux : le système conçu est donc spécifique à la détection des chaînes de coréférence dans les énoncés de français oral.

Les résultats d'évaluation obtenus par notre modèle de résolution, bien qu'assez proches de ceux de certains systèmes de l'état de l'art, sont peu généralisables en envisageant de tester le système sur des données « réelles ». Néanmoins, ces résultats expérimentaux nous ont permis d'observer certaines propriétés du phénomène étudié ainsi que d'envisager certaines pistes de travail pour améliorer les performances du modèle de classification, étendre ses capacités à celles d'un système *end-to-end*, et pour en compléter l'évaluation.

Les différents tests opérés au cours de ce travail nous ont permis d'observer que la réduction du nombre d'attributs décrivant les données pouvaient faire augmenter les performances de certains modèles de classification. Il apparaît donc que c'est moins la quantité de traits qui améliore les résultats, que leur pertinence individuelle et surtout leur pertinence au sein d'un ensemble : nous avons en effet observé qu'un même trait pouvait ne pas être intégré de la même manière dans deux modèles de classification distincts, selon l'ensemble d'attributs dans lequel il apparaît. Une perspective de travail serait de s'attacher précisément à cette sélection d'attributs, par exemple via une méthode de sélection ascendante qui évaluerait un modèle appris sur un ensemble ne contenant qu'un trait, puis ajouterait de manière incrémentale un nouveau trait à l'ensemble, en ne le conservant que si les résultats de classification sont meilleurs que pour l'ensemble précédent.

Les résultats observés pour nos différents tests ont également fait apparaître des différences de performance dépendant de l'ensemble de données fourni aux algorithmes d'apprentissage. Alors que la plupart des systèmes de l'état de l'art proposent de récupérer toutes les mentions insérées entre deux mentions effectivement coréférentes, nous avons limité cette récupération à seulement trois mentions (au maximum) du fait d'un temps de calcul conséquent. Il est tout à fait possible que ce choix ait influé sur les résultats de classification, pas forcément négativement d'ailleurs puisque certains classifieurs se révèlent meilleurs lorsqu'ils ont été appris sur un ensemble restreint de données. Nous pourrions

envisager de générer un ensemble d'apprentissage basé sur la même méthode que celle décrite par les systèmes existants, afin d'évaluer son apport aux modèles de classification.

Dans ce travail, la tâche de classification ne permet de ranger les instances que sous deux classes : soit COREF, soit NOT_COREF. En procédant ainsi, nous ne distinguons pas les différentes formes de reprises, et ne prenons pas en compte le fait que chacune d'entre elles est susceptible d'avoir des propriétés qui lui sont propres. Notamment, pour l'anaphore pronominale, on pourrait supposer que la distance entre les deux mentions d'une paire ne doit pas excéder un certain seuil. Plusieurs méthodes sont envisageables pour intégrer cette considération : d'une part, modifier les paramètres de classification pour que les différentes classes correspondent aux différents types de reprise, à savoir anaphore fidèle, infidèle, pronominale et associative ; d'autre part, s'inspirer des recherches de (Denis, 2007) qui propose d'apprendre des modèles spécifiques pour chaque type de reprises, à partir des mêmes traits qui se verront attribuer des poids différents en fonction du type d'expressions pris en compte.

Dans le cadre de ce mémoire, nous considérions disposer d'un corpus annoté en expressions référentielles, à partir duquel nous pouvions former des paires coréférentes et non-coréférentes à fournir aux programmes d'apprentissage. En plus de cette délimitation des frontières, ces unités sont annotées par des informations linguistiques telles que la catégorie syntaxique, le genre ou le nombre, et supposer partir de telles données restreint l'utilisation du système dont les entrées doivent être richement annotées. Une évolution majeure du système déjà en place serait d'intégrer une phase de prétraitement destinée à repérer et annoter les expressions potentiellement référentielles, avant de les soumettre au système de détection des chaînes de coréférence. Pour l'étape de repérage des unités, nous envisageons de faire un appel à un chunker en paramétrant sa recherche pour la récupération des chunks nominaux, ainsi qu'à un détecteur d'entités nommées. A l'issue de cette tâche, un analyseur morphosyntaxique déterminerait, pour chacune de ces unités, sa catégorie syntaxique, ses genre et nombre, et éventuellement sa tête nominale.

Une dernière perspective concerne l'évaluation du système, à la fois pour les données testées et les métriques utilisées. Les résultats obtenus pour nos différents systèmes n'intègrent pas pour l'instant le score de CEAF, à cause du très long temps de calcul qu'il nécessite. Pour véritablement intégrer et comparer nos résultats de recherche à ceux des systèmes existants, il est nécessaire d'implémenter ce calcul sur nos données, et cette tâche prioritaire est déjà à ce jour en train d'être étudiée. Un second facteur doit être intégré à l'évaluation pour maximiser l'objectivité des résultats, il s'agit de l'ensemble de test, qui, dans nos précédentes évaluations, étaient composés d'un sous-ensemble des données du corpus ANCOR. Bien que ce sous-ensemble soit complètement indépendant du corpus d'apprentissage, les données restent liées par leur appartenance au même corpus, et l'évaluation peut s'en retrouver biaisée. L'idéal pour contourner cette dépendance serait d'avoir accès à un corpus de français oral sans lien avec le projet ANCOR, pour lequel un système *end-to-end* chercherait les chaînes de coréférence. Cette perspective en entraîne d'ailleurs une autre à propos du format des données traitées : à l'heure actuelle, nos programmes sont conçus pour extraire des informations de fichiers structurés au format Transcriber, mais il serait nécessaire d'élargir nos analyses à d'autres formats de représentation de discours oraux pour généraliser l'usage de notre outil.

Même si beaucoup d'étapes restent à franchir avant d'atteindre les performances des meilleurs des systèmes existants, les conclusions de ce travail préliminaire sont prometteuses quant aux innovations possibles en matière de résolution de la coréférence en français.

Bibliographie

Antoine, J.-Y., Schang, E., Muzerelle, J., Lefeuvre, A., Pelletier, A., Eshkol, I., Maurel, D., and Villaneau, J. (2013). Corpus ancor_centre présentation générale. *http ://tln.li.univ-tours.fr/Tln_ Corpus_ Ancor.html.*

Bengtson, E. and Roth, D. (2008). Understanding the Value of Features for Coreference Resolution. *Empirical Methods in Natural Language Processing (EMNLP 2010)*, pages 236–243.

Cai, J. and Strube, M. (2010). Evaluation Metrics For End-to-End Coreference Resolution Systems. *Proceedings of SIGDIAL 2010 : the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 28–36.

Corblin, F. (1995). *Les chaînes de référence dans le discours.* Presses Universitaires de Rennes.

Denis, P. (2007). *New Learning Models for Robust Reference Resolution.* PhD thesis, University of Texas at Austin.

Gilbert, N. and Riloff, E. (2013). Domain-Specific Coreference Resolution with Lexicalized Features. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistices (ACL 2013)*, pages 81–86.

Haghighi, A. and Klein, D. (2007). Unsupervised coreference resolution in a nonparametric bayesian model. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 848–855.

Hendrickx, I., Hoste, V., and Daelemans, W. (2007). Evaluating Hybrid Versus Data-Driven Coreference Resolution. *Anaphora : Analysis, Algorithms and Applications. 6th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC 2007)*, pages 137–150.

Hoste, V. (2005). *Optimization Issues in Machine Learning of Coreference Resolution.* PhD thesis, University of Antwerp, Antwerp, Belgium.

Kobdani, H. and Schütze, H. (2010). Sucre : A Modular System for Coreference Resolution. In *Proceedings of the 5th International Workshop on Semantic Evaluation (ACL 2010)*, pages 92–95.

Lappin, S. and Leass, H. J. (1994). An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20 :535–561.

Lassalle, E. and Denis, P. (2013). Learning a hierarchy of specialized pairwise models for coreference resolution. In *Proceedings of TALN 2013 (Volume 1 : Long Papers)*, pages 118–131. ATALA.

Longo, L. (2013). *Vers des moteurs de recherche intelligents : un outil de détection automatique de thèmes.* PhD thesis, Université de Strasbourg.

Luo, X. (2005). On Coreference Resolution Performance Metrics. *Proceedings of Human Language Technology - Empirical Methods in Natural Language Processing (EMNLP 2005)*.

Mitkov, R. (2002). *Anaphora resolution*. Longman.

Muzerelle, J., Lefeuvre, A., Antoine, J.-Y., Schang, E., Maurel, D., Villaneau, J., and Eshkol, I. (2013). Ancor, premier corpus de français parlé d'envergure annoté en coréférence et distribué librement. In *TALN2013*.

Muzerelle, J., Schang, E., Antoine, J.-Y., Eshkol, I., Maurel, D., Boyer, A., and Nouvel, D. (2012). Annotations en chaînes de coréférences et anaphores dans un corpus de discours spontané en français. In *Actes 3éme Congrés Mondial de Linguistique Française (CMLF 2012)*.

Ng, V. and Cardie, C. (2002a). Combining Sample Selection and Error-Driven Pruning for Machine Learning of Coreference Rules. *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 55–62.

Ng, V. and Cardie, C. (2002b). Improving Machine Learning Approcahes to Corefrence Resolution. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

Raghunathan, K., Lee, H., Rangarajan, S., Chambers, N., Surdeanu, M., Jurafsky, D., and Manning, C. (2010). A Multi-Pass Sieve for Coreference Resolution. *Empirical Methods in Natural Language Processing (EMNLP 2010)*, pages 247–277.

Recasens, M. (2010). *Coreference : Theory, Resolution, Annotation and Evaluation*. PhD thesis, University of Barcelona.

Recasens, M. and Hovy, E. (1997). A Deeper Look into Features in Coreference Resolution. *Anaphora Processing and Application*, 5847 :29–42.

Schang, E., Muzerelle, J., Boyer, A., Antoine, J.-Y., Eshkol, I., and Maurel, D. (2011). Coreference and anaphoric annotations for spontaneous speech corpora in french. In *Proc. 8th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC 2011)*.

Schnedecker, C. (2012). La notion de saillance : problémes définitoires et avatars. In *La saillance : Aspects linguistiques et communicatifs de la mise en évidence dans un texte*, pages 23–43.

Soon, W. M., Ng, H. T., and Lim, D. C. Y. (2001). A Machine Learning Approach to Coreference Resolution of Noun Phrases. *Computational Linguistics*, 27(4) :521–544.

Stoyanov, V., Cardie, C., Gilbert, N., Riloff, E., Buttler, D., and Hysom, D. (2010). Reconcile : A Coreference Resolution Research Platform. Technical report.

Uryupina, O. (2004). Linguistically Motivated Sample Selection for Coreference Resolution. *Proceedings of DAARC-2004*.

Yang, X., Zhou, G., Su, J., and Tan, C. L. (2003). Coreference resolution using competition learning approach. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 176–183, Stroudsburg, PA, USA. Association for Computational Linguistics.

Annexes

Annexe A

Illustration des fichiers et formats

Cette annexe présente des extraits de fichiers permettant d'observer l'évolution du format des données au cours du projet. Ils sont présentés dans l'ordre chronologique des traitements appliqués.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE Trans SYSTEM "trans-14.dtd">
  <Trans scribe="(unknown)" audio_filename="00000014" version="4" version_date="130903">
   <Speakers>
    <Speaker id="spk1" name="hotesse" check="no" dialect="native" accent="" scope="local"/>
    <Speaker id="spk2" name="client" check="no" dialect="native" accent="" scope="local"/>
    <Speaker id="spk3" name="hotesse2" check="no" dialect="native" accent="" scope="local"/>
  </Speakers>
  <Episode>
   <Section type="report" startTime="0" endTime="45.936">
    <Turn startTime="0" endTime="1.719" speaker="spk1">
     <Sync time="0"/>U B S bonjour
    </Turn>
    <Turn speaker="spk2" startTime="1.719" endTime="5.976">
     <Sync time="1.719"/>oui bonjour e j'ai appelé ce matin pour avoir e
      monsieur Nom vous m'avez dit de rappeler
    </Turn>
    <Turn speaker="spk1" startTime="5.976" endTime="13.021">
     <Sync time="5.976"/>oui vous ê(tes) vous êtes madame c'est pour une une
      connexion e Internet peut-être pour votre e
    </Turn>
    <Turn speaker="spk2" startTime="13.021" endTime="17.3">
     <Sync time="13.021"/>voilà rappeler la date donc pour que c'était pour
      confirmer ma préinscription
    </Turn>
    <Turn speaker="spk1" startTime="17.3" endTime="20.217">
     <Sync time="17.3"/>d'accord je vais vous passer monsieur Nom hein
    </Turn>
    <Turn speaker="spk2" startTime="20.217" endTime="21.761">
     <Sync time="20.217"/>d'accord
    </Turn>
    <Turn speaker="spk1" startTime="21.761" endTime="40.02">
     <Sync time="21.761"/>t'as compris de quoi il s'agit Prénom hein j'ai pas
      dit Prénom2 hein j'ai dit Prénom hein
    </Turn>
    <Turn speaker="spk3" startTime="40.02" endTime="40.503">
     <Sync time="40.02"/>allo
    </Turn>
    <Turn speaker="spk1" startTime="40.503" endTime="42.085">
     <Sync time="40.503"/>une une préinscription
    </Turn>
    <Turn speaker="spk1 spk3" startTime="42.085" endTime="43.412">
     <Sync time="42.085"/>
     <Who nb="1"/>Prénom hein
     <Who nb="2"/>oui cette après-midi quoi
    </Turn>
    <Turn speaker="spk3" startTime="43.412" endTime="45.936">
     <Sync time="43.412"/>ouais ouais Prénom3 ouais
    </Turn>
   </Section>
  </Episode>
 </Trans>
```

FIGURE A.1 – Fichier corpus *019_ 00000014.ac*

```
<unit id="TXT_IMPORTER_1378212108853">
 <metadata>
  <author>TXT_IMPORTER</author>
  <creation-date>1378212108853</creation-date>
  <lastModifier>n/a</lastModifier>
  <lastModificationDate>0</lastModificationDate>
 </metadata>
 <characterisation>
<type>paragraph</type>
  <featureSet/>
 </characterisation>
<positioning>
 <start>
  <singlePosition index="1976"/>
 </start>
 <end>
  <singlePosition index="1984"/>
 </end>
</positioning>
</unit>
<unit id="jmuzerelle_1378212126204">
 <metadata>
  <author>jmuzerelle</author>
  <creation-date>1378212126204</creation-date>
  <lastModifier>jmuzerelle</lastModifier>
  <lastModificationDate>1378212131734</lastModificationDate>
 </metadata>
 <characterisation>
  <type>N</type>
  <featureSet>
   <feature name="GEN_REF">SPEC</feature>
   <feature name="NEW">YES</feature>
   <feature name="EN">ORG</feature>
   <feature name="DEF">DEF_SPLE</feature>
   <feature name="GP">NO</feature>
   <feature name="GENRE">F</feature>
   <feature name="NB">SG</feature>
  </featureSet>
 </characterisation>
 <positioning>
  <start>
   <singlePosition index="579"/>
  </start>
  <end>
   <singlePosition index="584"/>
  </end>
 </positioning>
</unit>
```

FIGURE A.2 – Extrait du fichier d'annotation *019_ 00000014.aa*, avec un élément *unit* décrivant un faux paragraphe et un autre décrivant une unité référentielle

```
   <Turn speaker="spk1 spk3" startTime="42.085" endTime="43.412">
     <Sync time="42.085" />
     <Who nb="1" />Prénom hein
    <Who nb="2" />oui cette après-midi quoi
   </Turn>
   <Turn speaker="spk3" startTime="43.412" endTime="45.936">
    <Sync time="43.412" /> ouais ouais Prénom3 ouais
   </Turn>
  </Section>
 </Episode>
</Trans>
<annotations>
 <metadata corpusHashcode="1998-729927" />
  <unit id="jmuzerelle_1378212126204">
   <characterisation>
    <type>N</type>
    <featureSet>
     <feature name="GEN_REF">SPEC</feature>
     <feature name="NEW">YES</feature>
     <feature name="EN">ORG</feature>
     <feature name="DEF">DEF_SPLE</feature>
     <feature name="GP">NO</feature>
     <feature name="GENRE">F</feature>
     <feature name="NB">SG</feature>
    </featureSet>
   </characterisation>
  </unit>
  <unit id="jmuzerelle_1378212137064">
   <characterisation>
    <type>N</type>
    <featureSet>
     <feature name="GEN_REF">SPEC</feature>
     <feature name="NEW">YES</feature>
     <feature name="EN">NO</feature>
     <feature name="DEF">DEF_DEM</feature>
     <feature name="GP">NO</feature>
     <feature name="GENRE">M</feature>
```

FIGURE A.3 – Extrait du fichier *fichierConcat019_00000014_clean.xml* issu de la concaténation de *019_00000014.ac* et *019_00000014.aa*

```
<Turn speaker="spk2" startTime="1.719" endTime="5.976">
 <Sync time="1.719" />
 oui bonjour e
 <anchor id="jmuzerelle_1378212199314">j'</anchor>
 ai appelé
 <anchor id="jmuzerelle_1378212137064">ce matin</anchor>
 pour avoir e
 <anchor id="jmuzerelle_1378212144084">monsieur Nom</anchor>
 <anchor id="jmuzerelle_1378212214674">vous</anchor>
 <anchor id="jmuzerelle_1378212218804">m'</anchor>
 avez dit de rappeler
</Turn>
<Turn speaker="spk1" startTime="5.976" endTime="13.021">
 <Sync time="5.976" />
 oui
 <anchor id="jmuzerelle_1378212224654">vous</anchor>
 ê(tes)
 <anchor id="jmuzerelle_1378212228434">vous</anchor>
 êtes
 <anchor id="jmuzerelle_1378212150514">madame</anchor>
 c'est pour une
 <anchor id="jmuzerelle_1378212155964">
 une connexion e
  <anchor id="jmuzerelle_1378212172764">Internet</anchor>
 </anchor>
 peut-être pour votre e
</Turn>
<Turn speaker="spk2" startTime="13.021" endTime="17.3">
 <Sync time="13.021" />
 voilà rappeler
 <anchor id="jmuzerelle_1378212180924">la date</anchor>
 donc pour que c'était pour confirmer
 <anchor id="jmuzerelle_1378212186604">ma préinscription</anchor>
</Turn>
<Turn speaker="spk1" startTime="17.3" endTime="20.217">
 <Sync time="17.3" />
 d'accord
 <anchor id="jmuzerelle_1378212240724">je</anchor>
 vais
 <anchor id="jmuzerelle_1378212244524">vous</anchor>
 passer
 <anchor id="jmuzerelle_1378212193244">monsieur Nom</anchor>
 hein
</Turn>
```

FIGURE A.4 – Extrait du fichier *fichierAnnot019_00000014_clean.xml* avec annotation ancrée des unités référentielles

```
<Turn speaker="spk2" startTime="1.719" endTime="5.976" num="2">
 <Sync time="1.719" />
 oui bonjour e
 <anchor id="jmuzerelle_1378212199314" num="2">j'</anchor>
 ai appelé
 <anchor id="jmuzerelle_1378212137064" num="3">ce matin</anchor>
 pour avoir e
 <anchor id="jmuzerelle_1378212144084" num="4">monsieur Nom</anchor>
 <anchor id="jmuzerelle_1378212214674" num="5">vous</anchor>
 <anchor id="jmuzerelle_1378212218804" num="6">m'</anchor>
 avez dit de rappeler
</Turn>
<Turn speaker="spk1" startTime="5.976" endTime="13.021" num="3">
 <Sync time="5.976" />
 oui
 <anchor id="jmuzerelle_1378212224654" num="7">vous</anchor>
 ê(tes)
 <anchor id="jmuzerelle_1378212228434" num="8">vous</anchor>
 êtes
 <anchor id="jmuzerelle_1378212150514" num="9">madame</anchor>
 c'est pour une
 <anchor id="jmuzerelle_1378212155964" num="10">
 une connexion e
  <anchor id="jmuzerelle_1378212172764" num="11">Internet</anchor>
 </anchor>
 peut-être pour votre e
</Turn>
<Turn speaker="spk2" startTime="13.021" endTime="17.3" num="4">
 <Sync time="13.021" />
 voilà rappeler
 <anchor id="jmuzerelle_1378212180924" num="12">la date</anchor>
 donc pour que c'était pour confirmer
 <anchor id="jmuzerelle_1378212186604" num="13">ma préinscription</anchor>
</Turn>
<Turn speaker="spk1" startTime="17.3" endTime="20.217" id="5">
 <Sync time="17.3" />
 d'accord
 <anchor id="jmuzerelle_1378212240724" num="14">je</anchor>
 vais
 <anchor id="jmuzerelle_1378212244524" num="15">vous</anchor>
 passer
 <anchor id="jmuzerelle_1378212193244" num="16">monsieur Nom</anchor>
 hein
</Turn>
```

FIGURE A.5 – Ilustration de l'addition d'un nouvel attribut numérique *num* aux éléments xml

```
<unit id="jmuzerelle_1378212322524">
 <characterisation>
  <type>PR</type>
  <featureSet>
   <feature name="GEN_REF">SPEC</feature>
   <feature name="NEW">NO</feature>
   <feature name="EN">PERS</feature>
   <feature name="DEF">DEF_SPLE</feature>
   <feature name="GP">NO</feature>
   <feature name="GENRE">UNK</feature>
   <feature name="NB">SG</feature>
  </featureSet>
 </characterisation>
 <positioning>
  <start>
   <singlePosition index="1473"/>
  </start>
  <end>
   <singlePosition index="1474"/>
  </end>
 </positioning>
</unit>
<relation id="jmuzerelle_1378212417644">
 <characterisation>
  <type>DIRECTE</type>
  <featureSet>
   <feature name="ID_LOC">NO</feature>
   <feature name="NOMBRE">YES</feature>
   <feature name="GENRE">YES</feature>
 </featureSet>
 </characterisation>
 <positioning>
  <term id="jmuzerelle_1378212193244"/>
  <term id="jmuzerelle_1378212144084"/>
 </positioning>
</relation>
```

FIGURE A.6 – Illustration des métadonnées des éléments *unit* et *relation* avant prétraitement

```
<unit id="jmuzerelle_1378212322524">
 <characterisation>
  <type>PR</type>
   <featureSet>
    <feature name="GEN_REF">SPEC</feature>
    <feature name="NEW">NO</feature>
    <feature name="EN">PERS</feature>
    <feature name="DEF">DEF_SPLE</feature>
    <feature name="GP">NO</feature>
    <feature name="GENRE">UNK</feature>
    <feature name="NB">SG</feature>
    <feature name="CONTENT">j'</feature>
    <feature name="SPEAKER">spk1</feature>
    <feature name="PREVIOUS">hein</feature>
    <feature name="NEXT">ai</feature>
 </featureSet>
  </characterisation>
<positioning>
   <start>
    <singlePosition index="1473"/>
   </start>
   <end>
    <singlePosition index="1474"/>
   </end>
  </positioning>
 </unit>
 <relation id="jmuzerelle_1378212417644">
  <characterisation>
   <type>DIRECTE</type>
   <featureSet>
    <feature name="ID_LOC">NO</feature>
    <feature name="NOMBRE">YES</feature>
    <feature name="GENRE">YES</feature>
    <feature name="Distance_Turn">3</feature>
    <feature name="Distance_Mention">12</feature>
    <feature name="Distance_Char">228</feature>
    <feature name="Distance_Word">40</feature>
   </featureSet>
  </characterisation>
  <positioning>
   <term id="jmuzerelle_1378212193244" />
   <term id="jmuzerelle_1378212144084" />
  </positioning>
 </relation>
```

FIGURE A.7 – Illustration des métadonnées des éléments *unit* et *relation* après prétraitement

```
<?xml version="1.0" encoding="UTF-8"?>
<METADATA>
  <PAIRE id="1">
    <feature name="M2_FORM">une préinscription</feature>
    <feature name="M1_PREVIOUS">confirmer</feature>
    <feature name="M2_NOMBRE">SG</feature>
    <feature name="M2_EN">NO</feature>
    <feature name="DISTANCE_MENTION">12</feature>
    <feature name="COM_RATE">0.33333334</feature>
    <feature name="M1_DEF">DEF_SPLE</feature>
    <feature name="M2_PREVIOUS">une</feature>
    <feature name="ID_PREVIOUS">NO</feature>
    <feature name="ID_GENRE">YES</feature>
    <feature name="ID_FORM">NO</feature>
    <feature name="DISTANCE_TURN">5</feature>
    <feature name="M2_TYPE">N</feature>
    <feature name="DISTANCE_WORD">31</feature>
    <feature name="DISTANCE_CHAR">153</feature>
    <feature name="M2_NEXT">$</feature>
    <feature name="classe">YES</feature>
    <feature name="M1_FORM">ma préinscription</feature>
    <feature name="M2_SPK">spk1</feature>
    <feature name="M2_GENRE">F</feature>
    <feature name="EMBEDDED">NO</feature>
    <feature name="M1_GENRE">F</feature>
    <feature name="ID_SPK">NO</feature>
    <feature name="INCL_RATE">0.5</feature>
    <feature name="ID_SUBFORM">NO</feature>
    <feature name="ID_EN">NA</feature>
    <feature name="M2_NEW">NO</feature>
    <feature name="ID_NEW">NO</feature>
    <feature name="M1_EN">NO</feature>
    <feature name="ID_NEXT">YES</feature>
    <feature name="ID_NOMBRE">YES</feature>
    <feature name="M1_TYPE">N</feature>
    <feature name="M1_SPK">spk2</feature>
    <feature name="M1_NEW">YES</feature>
    <feature name="M1_NEXT">$</feature>
    <feature name="ID_TYPE">YES</feature>
    <feature name="M2_DEF">INDEF</feature>
    <feature name="ID_DEF">NO</feature>
    <feature name="M1_NOMBRE">SG</feature>
  </PAIRE>
  <PAIRE id="2">
    <feature name="M2_FORM">une préinscription</feature>
    <feature name="M1_PREVIOUS">vais</feature>
    <feature name="M2_NOMBRE">SG</feature>
    <feature name="M2_EN">NO</feature>
    <feature name="DISTANCE_MENTION">10</feature>
    <feature name="COM_RATE">0.0</feature>
    <feature name="M1_DEF">DEF_SPLE</feature>
```

FIGURE A.8 – Extrait du fichier *featureFile019_ 00000014_ clean.xml* décrivant des instances d'apprentissage

```
@RELATION coreference
@ATTRIBUTE M2_FORM string
@ATTRIBUTE M1_PREVIOUS string
@ATTRIBUTE M2_NOMBRE {SG,PL,UNK,NULL}
@ATTRIBUTE M2_EN {NO,PERS,FONC,LOC,ORG,PROD,TIME,AMOUNT,EVENT,NULL,UNK}
@ATTRIBUTE DISTANCE_MENTION real
@ATTRIBUTE COM_RATE real
@ATTRIBUTE M1_DEF {INDEF,EXPL,DEF_DEM,DEF_SPLE,NULL,UNK}
@ATTRIBUTE M2_PREVIOUS string
@ATTRIBUTE ID_PREVIOUS {YES,NO,NA}
@ATTRIBUTE ID_GENRE {YES,NO,NA}
@ATTRIBUTE ID_FORM {YES,NO,NA}
@ATTRIBUTE DISTANCE_TURN real
@ATTRIBUTE M2_TYPE {N,P,PR,NULL,UNK}
@ATTRIBUTE DISTANCE_WORD real
@ATTRIBUTE DISTANCE_CHAR real
@ATTRIBUTE M2_NEXT string
@ATTRIBUTE M1_FORM string
@ATTRIBUTE M2_SPK string
@ATTRIBUTE M2_GENRE {M,F,UNK,NULL}
@ATTRIBUTE EMBEDDED {YES,NO,NA}
@ATTRIBUTE M1_GENRE {M,F,UNK,NULL}
@ATTRIBUTE ID_SPK {YES,NO,NA}
@ATTRIBUTE INCL_RATE real
@ATTRIBUTE ID_SUBFORM {YES,NO,NA}
@ATTRIBUTE ID_EN {YES,NO,NA}
@ATTRIBUTE M2_NEW {YES,NO,UNK,NULL}
@ATTRIBUTE ID_NEW {YES,NO,NA}
@ATTRIBUTE M1_EN {NO,PERS,FONC,LOC,ORG,PROD,TIME,AMOUNT,EVENT,NULL,UNK}
@ATTRIBUTE ID_NEXT {YES,NO,NA}
@ATTRIBUTE ID_NOMBRE {YES,NO,NA}
@ATTRIBUTE M1_TYPE {N,P,PR,NULL,UNK}
@ATTRIBUTE M1_SPK string
@ATTRIBUTE M1_NEW {YES,NO,UNK,NULL}
@ATTRIBUTE M1_NEXT string
@ATTRIBUTE ID_TYPE {YES,NO,NA}
@ATTRIBUTE M2_DEF {INDEF,EXPL,DEF_DEM,DEF_SPLE,NULL,UNK}
@ATTRIBUTE ID_DEF {YES,NO,NA}
@ATTRIBUTE M1_NOMBRE {SG,PL,UNK,NULL}
@ATTRIBUTE class {COREF, NOT_COREF}

@DATA
NO "confirmer" NO NO NO spk2 DEF_SPLE "ma préinscription" YES NO 5 N "une préinscription" NO
SG YES NO F "une" NO YES F N NA 12 NO 31 153 spk1 "$" INDEF SG "$" YES COREF

NO "quoi" NO NO NO spk1 EXPL "il" NO NO 2 PR "une préinscription" YES SG
NO NO F "une" NO YES M N NA 6 YES 16 74 spk1 "s'" INDEF SG "$" NO NOT_COREF
```

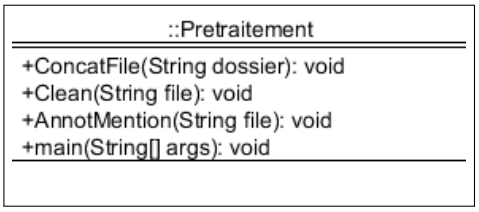
FIGURE A.9 – Extrait du fichier *wekaFormat019_ 00000014_ clean.arff* décrivant des instances d'apprentissage au format **.arff* (les instances sont ici représentées sur deux lignes pour des raisons de lisibilité, mais n'en constituent qu'une dans le véritable fichier)

Annexe B

Description des programmes et traitements

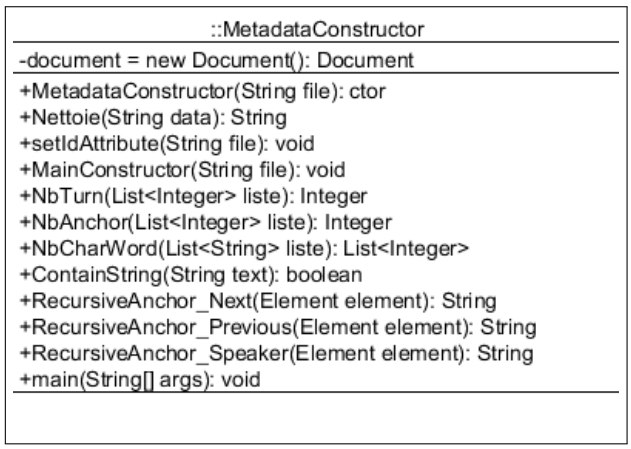
Cette annexe décrit les différentes étapes de traitement des fichiers, de leur format initial jusqu'à celui attendu par l'API de Weka. Sont également décrites les étapes suivantes de construction et d'utilisation du modèle de classification, puis l'évaluation des résultats après reconstitution des chaînes de coréférence.
Pour chaque étape, nous décrivons les entrées et sorties du programme et présentons un diagramme de la classe qui opère ce traitement. Nous précisions entre parenthèses le nombre de lignes de code écrites pour les traitements décrits. Ces informations renseignent globalement sur le rapport entre le travail fourni et le traitement à opérer.

1. **Prétraitements** (520 lignes)
   - **input** : données du corpus initial telles qu'illustrées en A.1 et A.2
   - **output** : fichiers concaténés, et unités référentielles balisées dans le texte (cf. figures A.3 et A.4)
   - **résumé** : à partir de deux fichiers xml, l'un contenant le texte, l'autre ses annotations, la méthode *ConcatFile* commence par concaténer les deux fichiers, puis la méthode *Clean* supprime les éléments non pertinents pour notre problématique et enfin la méthode *AnnotMenton* encadre les unités référentielles de balises <anchor>.

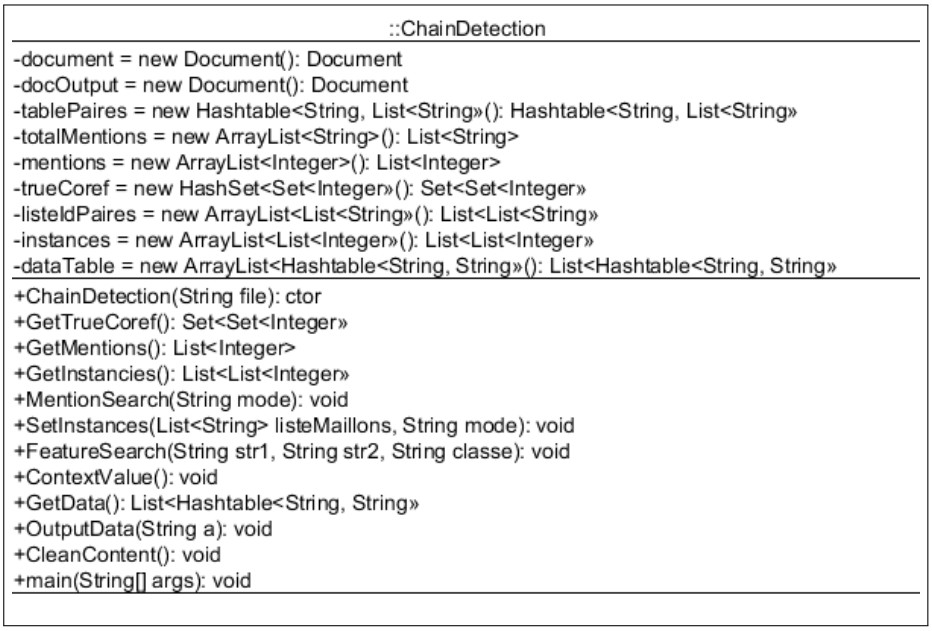


2. **Annotations des unités et relations** (263 lignes)
   - **input** : fichier issu du précédent traitement, dont les métadonnées des éléments sont illustrées en A.6
   - **output** : même fichier avec de nouvelles annotations pour les éléments *unit* et *relation* (cf. A.7)
   - **résumé** : la méthode *setIdAttribute* commence par ajouter aux éléments de chaque niveau de l'arbre xml (*Turn, Section, anchor*) un attribut numérique spécifiant sa position par rapport au début du fichier (cf illusration en A.5). La méthode *MainConstructor* appelle ensuite successivement les méthodes *NbTurn, NbAnchor, NbCharWord* pour calculer les distances séparant deux unités d'une paire coréférente de chaque élément *relation*. Concernant les métadonnées des éléments *unit*, la même méthode *MainConstructor* appelle successivement les

méthodes *RecursiveAnchor_Speaker, RecursiveAnchor_Previous, RecursiveAnchor_Next* qui recherche respectivement le locuteur d'une unité, ainsi que ses contextes gauche et droit.

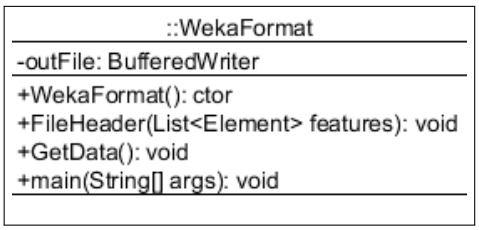


3. **Génération des instances d'apprentissage**
   (a) **Récupération des paires** (480 lignes)
   - **input** : fichier prétraité, avec métadonnées riches et annotations ancrées des unités
   - **output** : fichier xml contenant une liste d'éléments *PAIRE* décrivant les instances d'apprentissage (cf. figure A.8)
   - **résumé** : la méthode *MentionSearch* crée une liste contenant tous les maillons d'une chaîne, puis cette liste est fournie à la méthode *SetInstancies* qui construit des paires de mentions de la manière décrite en section 4.2 du mémoire (le paramètre *mode* permet de préciser le nombre d'instances négatives à construire). Chacune des instances est passée à la méthode *FeatureSearch* qui calcule les traits descriptifs, et toutes ces informations sont concaténées dans un même fichier xml via la méthode *OutputData*.

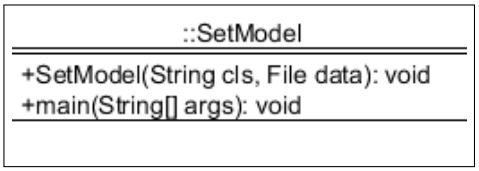


   (b) **Transformation au format .arff** (128 lignes)
   - **input** : fichier xml décrivant les instances (figure A.8)

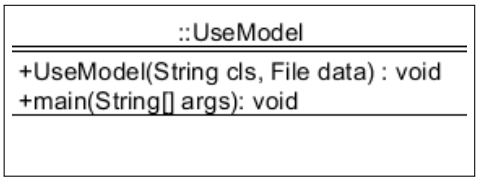
- **output** : fichier au format .arff décrivant les instances (figure A.9)
- **résumé** : la constructeur *WekaFormat* crée un fichier puis la méthode *FileHeader* y inscrit en en-tête la liste des attributs. La méthode *GetData* fait ensuite une itération sur chacun des fichiers xml d'instances, en récupérant les informations des éléments *PAIRE* avant de les transposer au format attendu par Weka.



4. **Apprentissage du modèle de classification** (45 lignes)
   - **input** : fichier .arff
   - **output** : modèle de classification
   - **résumé** : le constructeur *SetModel* attend deux paramètres que sont le fichier contenant les instances et le type de classifieur utilisé (dans notre cas, soit SMO, soit J48 soit NaiveBayes). A l'issue du calcul, un fichier au format **.model* est généré.



5. **Evaluation du classifieur**
   (a) **Classification des données de test + calcul de BLANC** (53 lignes)
   - **input** : fichier .arff décrivant les données de test avec leurs vraies classes
   - **output** : résultats de la classificaiton en terme de précision, rappel, F_mesure et BLANC et données de test annotées par les classes prédites par le modèle
   - **résumé** : le constructeur *UseModel* attend deux paramètres qui sont le fichiers d'instances de test et le classifieur précédemment généré. L'API Weka propose des méthodes d'évaluation des résultats grâce auxquelles on calcule celui de BLANC



   (b) **Evaluation selon MUC et B^3** (269 lignes)
   - **input** : fichier **.arff* contenant les résultats de classification
   - **output** : score de MUC et B^3
   - **résumé** : le constructeur *chainComparator* prend en entrée le fichier **.arff*, puis la méthode *SystemReponse* reconstitue les chaînes prédites en comparant les instances classées en COREF par le système à l'ensemble des instances précédemment générées via la classe *ChainDetection*. L'ensemble des chaînes est ensuite fourni à la méthode *AddSingletons* qui complète cet ensemble par les mentions non classées. Cet ensemble est comparé à l'ensemble de référence, qui est généré une fois sur le même principe à partir des vraies classes des instances. Pour les métriques d'évaluation, c'est la méthode *ClusterScore* du package *LingPipe* qui se charge de leur calcul à partir des deux

ensembles de données fournies en entrée (ces procédures sont implémentées dans la méthode principale *main*).