Segmentation et classification : deux politiques complémentaires

Alexandre Labadié (1), Yann Romero (1), Laurianne Sitbon (2)
(1){alexandre.labadie,yann.romero}@iup.univ-avignon.fr
(2)laurianne.sitbon@univ-avignon.fr
LIA - Université d'Avignon et des pays du Vaucluse
Agroparc, BP 1228, 84911, Avignon, France

Mots-clefs : méthode d'apprentissage, segmentation thématique, chaînes lexicales, n-grammes, modèles de Bayes, algorithme de Viterbi

Keywords: learning methods, topic segmentation, lexical chains n-grams, Bayes' model, Viterbi's algorithm

Résumé Dans cet article, nous nous sommes attachés à étudier le problème de l'extraction de segments de texte non pertinents dans un texte selon deux approches distinctes. La première consiste à appliquer au texte un algorithme de segmentation sans apprentissage afin de pouvoir ensuite le classifier, bloc par bloc, à l'aide d'un modèle bayésien. La seconde approche consiste à déterminer la probabilité de chacun des orateurs à partir d'un modèle tri-gramme. Dans un second temps, nous avons appliqué un certain nombre d'optimisations relatives aux contraintes imposées.

Abstract In this paper, we tried to study the problem of extracting irrelevant text segments from a text using two different methods. The first one applies a segmentation algorithm to the text in order to classify it block by block with a BayesŠs model. The second one determines the probability of each speaker with a 3-gram model. Furthermore, we applied different optimization.

Introduction

Le problème posé par la détection de phrases de François Mitterand au sein de discours de Jacques Chirac tel que le propose DEFT05[1] peut être décomposé en deux problèmes distincts. Le premier est la détection de ruptures au sein du discours. En effet, l'insertion de segments de texte issus d'un orateur différent, qui de plus traitent d'un sujet différent, provoquent une certaine incohérence dans le discours. Nous sommes partis du principe que cette incohérence pouvait être considérée de la même manière qu'un changement de thème au sein d'un texte. Aussi nous nous sommes attachés à détecter ces "ruptures thématiques" avec les algorithmes sans apprentissage que sont C99 ou encore celui de la méthode LIA_seg. Le deuxième problème posé par cet atelier était de classifier les portions de texte obtenues par la segmentation. Ici encore, nous avons exploré deux champs de possibilité, à savoir une classification via un modèle bayésien et une classification grâce à un apprentissage par n-grammes. Tout l'enjeu de notre démarche aura été de démontrer l'efficacité de la combinaison de méthodes avec et sans apprentissage.
Nous verons donc dans cet article les résultats de ces différentes méthodes. Dans la section 1, nous présentons la combinaison d'algorithmes de segmentation avec une méthode de classification. La section 2 est consacrée à une approche par n-grammes de la classification qui ne fait appel à aucun filtrage. Nos résultats sont exposés et analysés dans la section 3. Enfin, nous tentons de dégager les enseignements de notre démarche dans la section 4.

1 Combinaison de méthodes de segmentation sans apprentissage avec une classification bayésienne

Nous avons choisi de segmenter le texte dans un premier temps afin de permettre une classification plus robuste. En effet la segmentation thématique fournit des unités de traitement plus riches et tout aussi cohérentes que les phrases. On notera que le texte en entrée des méthodes décrites dans cette section aura été au préalable traité avec l'étiqueteur LIA_tagger[2], pour ne conserver que les lemmes d'un nombre limité de catégories (noms, verbes, adjectifs et noms propres).

1.1 Une segmentation sans apprentissage : Variante de C99

Nous avons utilisé ici une variante sur l'algorithme C99 (Choi F., 2000). Nous nous appuyons donc sur l'idée de base de la méthode, à savoir que les mesures de similarité entre des segments de textes courts sont statistiquement insignifiantes, et que donc seul des classements locaux sont à considérer pour ensuite appliquer un algorithme de catégorisation sur la matrice de similarité. Dans un premier temps, une matrice de similarité est donc construite, représentant la similarité entre toutes les phrases du texte à l'aide d'une mesure de similarité classique : le cosinus. On le calcule donc pour chaque paire de phrases du texte, en utilisant chaque mot commun entre les phrases, après avoir éliminé du texte les mots inutiles et la ponctuation et avoir lemmatisé les mots conservés. Ainsi en considérant $f_{i,j}$ la fréquence du mot j dans le segment j, la similarité

[1] http://www.lri.fr/ia/fdt/DEFT05

[2] version uniquement probabiliste de ECSta (Spriet T. and El-Beze M., 1998) distribué par le LIA sous licence GPL

entre les segments x et y est :

$$sim(x,y) = \frac{\sum_j f_{x,j} \times f_{y,j}}{\sqrt{\sum_j f_{x,j}^2 \times \sum_j f_{y,j}^2}} \quad (1)$$

On effectue ensuite un « classement local », en déterminant pour chaque paire d'unités textuelles, le rang de sa mesure de similarité par rapport à ses $m \times n - 1$ voisins, m et n étant les dimensions du masque de classement choisi. Le rang est le nombre d'éléments voisins ayant une mesure de similarité plus faible, conservé sous la forme d'un ratio r afin de prendre en compte les effets de bord.

$$r = \frac{\text{rang}}{\text{nombre de voisins dans le masque}} \quad (2)$$

Une matrice de rang est ainsi créée et vient remplacer la matrice de similarité. Comme dans l'algorithme original, nous considérons la densité D des carrés, le long de la diagonale de la matrice de rang :

$$D = \frac{\sum_{k=1}^{m} s_k}{\sum_{k=1}^{m} a_k} \quad (3)$$

avec a_k l'aire du carré et s_k son poids qui est la somme des tous les rangs des phrases qu'il contient.

Là où notre méthode varie par rapport à l'algorithme original dans l'exploitation de la matrice de rang et des densités calculées. Nous calculons cette densité pour chaque carré de la diagonale ayant pour coin inférieur la limite inférieure de la diagonale.

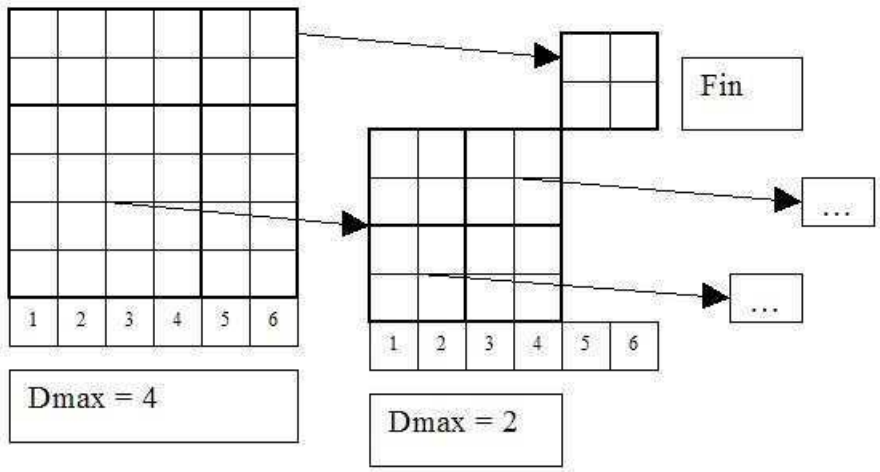

Figure 1: algorithme récursif appliqué à la matrice de rang

La plus forte densité obtenue indique le segment où il y a le plus de chance qu'une rupture dans la logique du discours se soit produite. Nous pouvons donc considérer les deux sous matrices de rang ainsi obtenues le long de la diagonale observée et répéter la même opération de manière récursive sur les matrices suivantes jusqu'à ce que le segment frontière donné par le calcul des densités soit la limite supérieure de la diagonale de la matrice courante ou jusqu'à ce que le nombre de segments contenus dans la matrice soit égal à deux.

1.2 Segmentation avec LIA_seg

L'outil LIA_seg utilisé pour effectuer la segmentation est détaillé et évalué dans (Sitbon L. and Bellot P., 2005). La segmentation s'effectue à l'aide de chaînes lexicales pondérées.

Une chaîne lexicale relie des termes de manière linéaire dans un texte. Les méthodes actuelles de segmentation les utilisent pour relier les occurrences d'un même terme (ou lemme) qui sont "proches". Une chaîne est rompue lorsque le nombre de termes qui séparent deux occurrences dépasse une valeur fixée appelée hiatus. La figure 2 illustre le processus de création des chaînes. On peut alors recenser pour chaque phrase les chaînes actives.

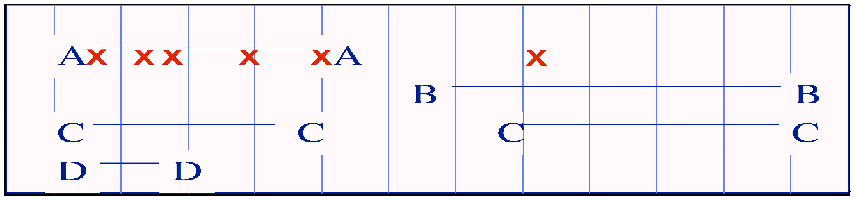

Figure 2: Construction des chaînes lexicales

Les applications des chaînes lexicales utilisent actuellement des hiatus définis de manière empirique, et la notion d'activité d'une chaîne est binaire (elle est active ou non active). (M. Galley et al., 2003) proposent une pondération des chaînes en fonction de la compacité et de la fréquence du terme considéré, avec un hiatus d'une distance de 11 phrases défini empiriquement. Le poids d'une chaîne associée à un terme m est défini par :

$$score(Chaine, m) = freq(Chaine, m) \times log(\frac{L_{texte}}{L_{chaine}}) \quad (4)$$

Où $freq(Chaine, m)$ est le nombre d'occurrences du terme m dans la chaîne, L_{texte} la longueur du texte, et L_{chaine} la longueur de la chaîne (on ne peut pas dépendre de la taille des textes à segmenter).

Puis on calcule les similarités à chaque fin de phrase, considérée comme une rupture thématique potentielle. La similarité est calculée avec :

$$sim(A, B) = \frac{\sum_m score(A, m) \times score(B, m)}{\sqrt{\sum_m score(A, m) \times \sum_m score(B, m)}} \quad (5)$$

Où A et B sont les ensembles de vecteurs représentant les poids des chaînes lexicales actives dans les n phrases avant et après (nous avons choisi $n = 2$), $score(X, m)$ étant le poids maximal du terme m dans l'ensemble des vecteurs X.

Les frontières retenues sont alors celles dont la similarité est localement la plus basse (inférieure aux 3 fins de phrases précédentes et suivantes) et en dessous d'un seuil déterminé par :

$$sim_{limit} = \mu + \frac{\sigma}{2} \quad (6)$$

où μ et σ sont la moyenne et la variance de toutes les similarités calculées (M. Galley et al., 2003).

1.3 Une classification à partir d'un modèle bayésien simple

L'apprentissage se fait seulement sur les mots jugés utiles à savoir les noms communs, les noms propres, les verbes et les adjectifs.

Afin de pouvoir attribuer les segments de texte à l'un ou l'autre des orateurs, nous nous sommes appuyés sur un modèle probabiliste de type bayésien. Ainsi, la probabilité d'un mot pour un orateur O est la fréquence de ce mot pour O sur le nombre total de mots de O (ici w_O est le nombre d'apparitions du mot w pour l'orateur O et N_O le nombre total de mots observés pour l'orateur O).

$$P(w|O) = \frac{w_O}{N_O} \tag{7}$$

De fait, la probabilité qu'un segment ait été émis par l'orateur O est le produit des probabilités de chacun des mots du segment pour O (on fait l'hypothèse que les $P(w_i|O)$ sont indépendants les uns des autres).

$$P(s|O) \approx \prod_i P(w_i|O) \tag{8}$$

Cette probabilité n'est toutefois pas utilisée telle quelle. En effet, on intègre à cette probabilité la notion de longueur de phrase au travers d'une loi normale calculée sur la moyenne et l'écart type de la longueur des segments pour chacun des présidents. Ainsi $N_O(x)$ est la probabilité qu'un segment de longueur x ait été émis par l'orateur O. On a donc :

$$P'(s|O) = P(s|O) \times N_O(x) \tag{9}$$

Ensuite, elle est normalisée de telle sorte que $\sum_i P(O_i|s) = 1$. Au cours de l'apprentissage, on apprend également les probabilités de trigrammes de mots pour chaque orateur que l'on emploiera plus loin lors des post-traitements.

$$P(w_1, w_2, w_3|O) = \frac{P(w_1, w_2, w_3, O)}{P(w_1, w_2, w_3)} \tag{10}$$

1.4 La jonction des deux méthodes

L'objectif de notre démarche était de combiner une méthode sans apprentissage de segmentation avec une méthode de classification sujette à apprentissage. Aussi plutôt que d'étiqueter chaque segment indépendamment, nous avons tiré parti de la segmentation en utilisant le classifieur sur les « groupes de segments contigus » obtenus lors de la segmentation. Ainsi, on calcule la probabilité qu'un groupe G soit issu d'un orateur O pour chacun des orateurs.

$$P(G|O) \approx \prod_i P'(s_i|O) \tag{11}$$

On effectue ensuite une normalisation afin que $\sum_i P(O_i|G) = 1$ et l'orateur qui obtient ainsi la meilleure probabilité donne son étiquette à l'ensemble du groupe.

1.5 Viterbi en post-traitement

La dernière étape de cette approche est un post-traitement à partir d'un automate de Markov avec lequel on applique l'algorithme de Viterbi. Pour cet automate, on prend pour probabilités d'émission et de transition non seulement les probabilités de chacun des orateurs pour un bloc, mais aussi les probabilités de continuité d'un bloc à l'autre. Si s_1 est le segment frontière du bloc supérieur et s_2 celui du bloc inférieur et si s_1 contient n mots pleins alors la probabilité de continuité pour l'orateur O entre s_1 et s_2 (et donc entre les deux blocs de segments) est :

$$P(\textit{continuit}|O) = \max\{P(w_{s_1,n-1}, w_{s_1,n}, w_{s_2,1}|O), P(w_{s_1,n}, w_{s_2,1}, w_{s_2,2}|O)\} \tag{12}$$

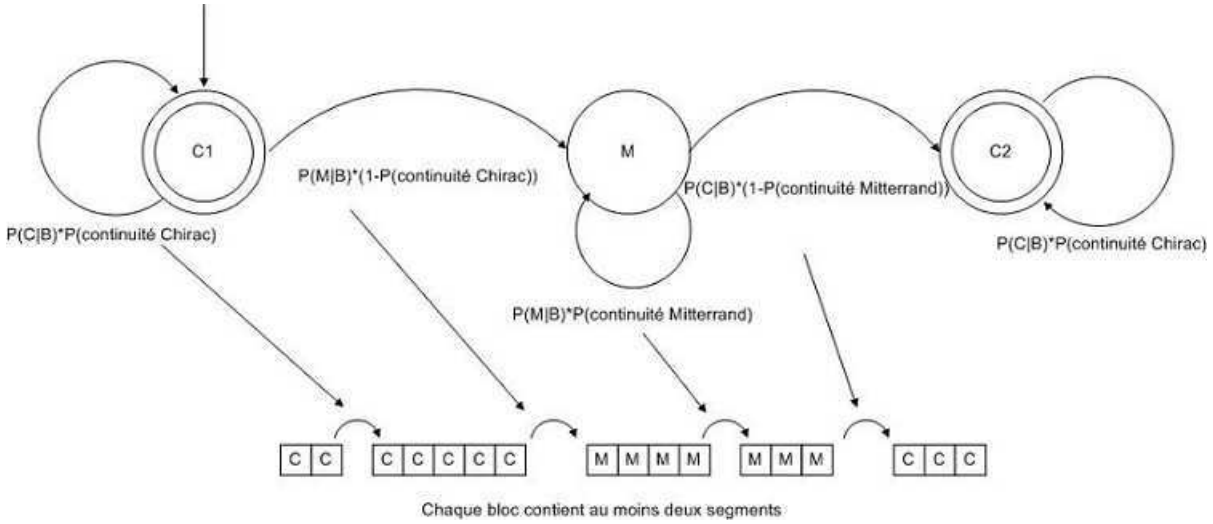

Figure 3: Automate de Markov utilisé dans le post-traitement

On obtient ainsi l'automate représenté en figure 3.

L'automate que nous utilisons devant permettre de trouver le meilleur chemin parmi des groupes de segments d'au moins deux segments chacun, sa forme est relativement simple. Ainsi il ne comporte que trois états et suppose que l'état initial et l'état final soient tous les deux des « blocs Chirac ». Ce postulat peut sembler faux, en effet rien indique que l'on commence ou finisse obligatoirement par un ensemble de segments de Chirac dans l'énoncé de l'atelier. Pourtant c'est la forme de l'automate qui a donné les meilleurs résultats, du fait probablement, qu'elle correspond plus à la réalité du corpus.

2 Apprentissage et optimisation

Cette méthode consiste à comparer les probabilités des n-grammes trouvés en apprentissage avec ceux détectés lors du test. Nous n'effectuons aucun filtrage, seuls les signes de ponctuations sont supprimés. En effet, le but de la méthode est de faire la distinction entre des discours de Chirac et de Mitterrand, donc on se focalise seulement sur les mots.

2.1 L'apprentissage par n-gramme

Nous utilisons le corpus d'apprentissage remis par DEFT. Le corpus est divisé en deux parties : 80 % pour l'apprentissage et 20 % pour le test. Nous résultats sont présentés en section.

Sur le corpus d'apprentissage, nous calculons les probabilités d'avoir un n-gramme. Ces probabilités sont évaluées pour Chirac et Mitterrand. Puis, pour chaque type de n-gramme (unigramme, bigramme, ...) on détermine une probabilité minimale entre Chirac et Mitterand. Cette probabilité minimale sera utilisée lors du processus de test.

La classification se fera sur chaque phrase du corpus de test. Premièrement, nous déterminons les n-grammes de la phrase. Puis nous recherchons si ces n-grammes ont été trouvés en phase d'apprentissage. Trois cas peuvent se produire :

- Aucun n-gramme n'a été trouvé. Dans ce cas, rien n'est fait.
- Un n-gramme a été trouvé chez Chirac ou Mitterrand. On ajoute au score du vainqueur, la probabilité du n-gramme calculé en phase d'apprentissage. On ajoute au score du perdant la probabilité minimale. Ceci évite que le perdant soit mis à l'écart.
- Le n-gramme a été trouvé chez Chirac et Mitterrand. Dans ce cas, on ajoute au score des deux classes les probabilités respectives.

Une fois la détection achevée, les scores sont normalisés par le maximum des deux. Puis, on récupère l'exponentielle de chaque score. On effectue encore une normalisation par la somme des n-grammes entre Chirac et Mitterrand. Puis on applique la formule suivante :

$$f_n = Score_{ngrammeChirac} - Score_{ngrammeMitterand} \; avec \; n = \{1, 2, 3\} \tag{13}$$

$$f = \sum_{n=1}^{3} \lambda_n * f_n \tag{14}$$

Si $f < \varepsilon$, alors la phrase appartient à Chirac Sinon elle appartient à Mitterand. ε vaut -0.025. λ_n varie en fonction des n-grammes. λ_1 correspond au modèle unigramme, λ_2 correspond au modèle bigramme et λ_3 correspond au modèle trigramme.

Unigramme λ_1	Bigramme λ_2	Trigramme λ_3
0.56	0.4	0.1

Table 1: Valeurs des lambdas en fonctions de n-grams

Les lambdas ont été déterminés par approches successives.

2.2 Optimisation par alternance et par contrainte

L'optimisation par alternance est une méthode triviale qui consiste à transformer les phrases étiquetées Chirac en Mitterrand si elles se trouvent entre deux phrases de Mitterrand. Le cas inverse est également traité.

L'optimisation par contrainte est une méthode qui consiste à étiqueter les deux premières phrases et les deux dernières phrases d'un discours comme étant du Chirac. Cette méthode résulte des observations effectuées sur le corpus d'apprentissage. Le corpus débute et se termine en moyenne par deux phrases de Chirac.

3 Expériences

Les trois exécutions présentées à l'atelier correspondent aux trois approches décrites précédement. La table 2 récapitule les résultats obtenus pour chacune des tâches de l'atelier avec chacune des méthodes.

	Tache 1	Tache 2	Tache 3
C99 + Bayes	0.759816	0.741895	0.745863
LIA_seg + Bayes	0.668947	0.657491	0.65957
3-grammes	0.727665	0.731987	0.741702

Table 2: Récapitulatif des résultats officiels de l'équipe junior du LIA

3.1 C99 + Bayes

La combinaison de notre variante de C99 et d'une classification bayésienne nous a donné les meilleurs résultats avec des valeurs autour de 0.75 lorsqu'un filtrage est appliqué sur l'apprentissage bayésien. Cette valeur se dégrade nettement sans filtrage (table 3).

	Développement	Développement sans filtrage	Test	Test sans filtrage
tâche 1	0.74697	0.715526	0.759816	0.740328

Table 3: Les résultats de C99 avec une classification bayésienne sur la tâche 1

Même si ces résultats ne sont pas présentés ici, on notera que les meilleures performances de la segmentation sont obtenues avec un filtrage quasi inexistant. En effet lors de nos expériences nous avons constaté une baisse de performance de la segmentation lorsque l'on réduisait le champ des mots utilisés. On peut donc déduire que contrairement au cas où l'on souhaite trouver des frontières entre thèmes, pour différencier deux orateurs les mots outils et la ponctuation sont utiles. Cela peut facilement s'expliquer par le fait que les deux orateurs ont des habitudes, des « manies » dans leur façon de s'exprimer qui se manifestent par l'emploi de certaines formes riches en mots outils. On remarquera que les résultats sont tirés vers le bas par la méthode de classification. En effet, si une classification idéale[3] avait été appliquée sur les résultats de la segmentation du corpus de test le F_{score} obtenu aurait été de 0.931596 sur la tâche 1 avec en moyenne 3.9 phrases par segment.

3.2 LIA_seg + Bayes

Afin de pouvoir généraliser nos résultats, nous avons procédé aux expérimentations avec LIA_seg sans optimisations contextuelles. Elles auraient pu consister à ajouter des frontières systématiquement aux changements de discours par exemple. De même nous avons utilisé les paramètres standards de LIA_seg, au lieu de les déterminer de manière empirique. Ceci explique en partie les meilleures performances de la méthode précédente. Nous nous attacheront ici à montrer comment nous aurions pu optimiser nos résultats, et quelle est la part de réussite liée à l'utilisation de la segmentation.

Nous avons évalué la pertinence de la méthode de segmentation en calculant le F_{score} "en cas idéal", c'est à dire si la classification des segments est optimale. Pour cela, on établit la liste des changements de locuteur dans le corpus contenant les index. On les compare aux frontières calculées par le segmenteur : pour chaque changement, on recherche la frontière calculée la plus proche. Si elle est avant un passage de Chirac vers Mitterand (C->M) ou après un passage M -> C, on considère que les phrases seront ajoutées, et on ajoute la différence dans M_{aj}. Si elle est après un passage C -> M ou avant un passage M ->, on considère que les phrases seront

[3]Classification qui donnerait la bonne étiquette à chaque segment avec une probabilité de 100 %

oubliées à l'extraction, et on ajoute la différence dans M_{oub}. Le nombre total de phrases de Mitterand dans l'index M_{tot} est compté dans l'index directement.

le F_{score} défini par :

$$F_{score} = \frac{2 \times M_{corrects}}{M_{tot} + Nb_{segments}} \tag{15}$$

devient alors :

$$F_{score} = \frac{2 \times (M_{tot} - M_{oublis})}{(M_{tot} + M_{ajouts} - M_{oublis}) + M_{tot}} \tag{16}$$

Les résultats de ce calcul du F_{score} en cas idéal sur DEFT05 sont présentés dans la table 4.

	tâche1	tâche2	tache3
F_{score}	0,909762	0.909473	0,908757

Table 4: Calcul du F_{score} en cas de classification idéale des segments thématiques.

Les résultats en cas idéal montrent que le classifieur n'est peut-être pas tout à fait adapté aux informations qu'on lui fournit. On pourrait améliorer la classification en proposant des segments plus grands avec LIA_seg, c'est à dire en augmentant le seuil de décision de rupture 6. Mais alors la fiabilité des frontières trouvées et le F_{score} idéal baisseraient également. Une étude empirique permettrait de déterminer le meilleur ratio.

Nous avons essayé une classification avec les SVM (Support Vector Machines), qui donnait de bons résultats sur les segments de référence (en apprenant sur la moitié du corpus d'apprentissage, avec une matrice creuse contenant l'index des lemmes présents dans chaque segment et leur nombre d'occurrences, avec un noyau gausssien (std = 2), et les paramètres par défaut de SVM-Torch, on avoisinait les 92% de bonne classification pour les segments de la deuxième moitié du corpus d'apprentissage). Cependant la méthode ne parvenait pas à classifier les segments issus de notre méthode, peut être parce qu'ils sont trop petits (6,7 phrases en moyenne) .

3.3 n-grammes, probabilités et optimisations

	Appr. par n-grammes	Opt. par alternance	Opt. par contrainte
tâche 1	0.6152	0.7253	0.727663
tâche 2	0.6230	0.7298	0.731987
tâche 3	0.6341	0.7384	0.741702

Table 5: Apprentissage par n-gramme et optimisations

La méthode d'apprentissage par alternance donne d'assez bon résultats. Couplée à l'optimiseur, nous nous retrouvons avec des résultats satisfaisants. Elle est d'autant plus représentatif qu'il y a beaucoup de termes dans la phrase à analyser. En effet, le score de la phrase contiendra une somme de probabilité. Ce score sera d'autant plus représentatif s'il y a beaucoup de termes.

$$ScoreHommePolitique_{k-gramme} = \sum_{n=1}^{n} p_k \; avec \; k = 1, 2, 3 \tag{17}$$

Le problème des phrases courtes dans les discours est résolu avec les méthodes d'optimisation et plus particulièrement celle d'alternance. La méthode n'est pas tres efficace sur la détection

de rupture entre du Chirac et du Mitterrand. En effet, les résultats montrent que la méthode d'apprentissage détecte des blocs de discours de Mitterrand. Or, d'après la règle du concours, il n'y a qu'un bloc de Mitterand dans un discours.

La méthode d'apprentissage est fiable pour une détection de phrases mais un prétraitement doit être effectué pour juger la crédibilité de la détection.

4 Conclusion

Les méthodes que nous avons testées pour le filtrage se sont révélées plutôt efficaces, si on les compare à la moyenne des résultats de DEFT. Ce sont des méthodes stochastiques fondées sur la cohésion lexicale et les modèles de langage. Cependant, la tâche proposée par DEFT contenait des changements thématiques en même temps que les changements de locuteur, il serait intéressant maintenant de tester la robustesse de nos méthodes sur d'autres types de corpus ou pour des traitements plus « fins ».
Il serait util de pouvoir tenir compte de la syntaxe des phrases, dans la segmentation comme dans la classification. En effet, la structure des phrases est sans aucun doute caractéristique d'un orateur, surtout dans un domaine tel que la politique. La possibilité d'intégrer cette notion devrait permettre d'améliorer encore les performances des méthodes employées. Enfin, une analyse fonctionnelle ainsi que la gestion des anaphores enrichirait le composant dédié au calcul de continuité.

Remerciements

Nous remercions Marc El-Bèze et Juan Manuel Torres pour leurs conseils et leur disponibilité, Frédéric Bechet qui aura consacré un peu de son temps à nous préparer les corpus et enfin Laurent Gillard qui nous aura supporté avec un grand stoïcisme dans son espace de travail.

Références

L. Sitbon and P. Bellot. (2005), Segmentation thématique par chaînes lexicales pondérées, Actes de *TALN'05* Dourdan, France.

L. Sitbon and P. Bellot. (2004), Adapting and comparig linear segmentation methods for french., Actes de *RIAO'04* Avignon, France.

M. Galley and K. McKeown and E. Folser-Lussier and H. Jing. (2003), Discourse Segmentation of Multi-Party Conversation., Actes de *ACL'03* Sapporo, Japan.

Freddy Y. Y. Choi (2000), Advances in domain independent linear text segmentation, *Proceedings of NAACL-00.*

Christopher D. Manning and Hinrich Schültz (1999), *Foundations of Statistical Natural Language Processing* The MIT Press, Cambridge, Massachusetts, London, England.

Thierry Spriet and Marc El-Beze (1998), *Introduction of Rules into a Stochastic Approach for Language Modelling* NATO ASI Series F, Keith Ponting.