Classification automatique pour la compréhension de la parole : vers des systèmes semi-supervisés et auto-évolutifs

Pierre Gotab

ACADEMIE D'AIX-MARSEILLE
UNIVERSITE D'AVIGNON ET DES PAYS DE VAUCLUSE
ECOLE DOCTORALE SCIENCES ET AGROSCIENCES (ED 536)

THÈSE

Pour obtenir le titre de Docteur en Sciences
de l'Université d'Avignon et des Pays de Vaucluse

Présentée par

Pierre GOTAB

CLASSIFICATION AUTOMATIQUE POUR LA COMPREHENSION DE LA PAROLE

Vers des systèmes semi-supervisés et auto-évolutifs

Soutenue publiquement le 4 décembre 2012
devant un jury composé de :

LEFÈVRE Fabrice	Professeur, LIA, Université d'Avignon	*Président du jury*
ESTEVE Yannick	Professeur, LIUM, Université du Maine	*Rapporteur*
GRAVIER Guillaume	CR-HDR, IRISA, Université de Rennes 1	*Rapporteur*
DAMNATI Géraldine	DR, Orange Labs, France Télécom R&D	*Examinatrice*
FAVRE Benoit	MCF, LIF, Aix Marseille Université	*Examinateur*
SCHNEIDER Jean-Jacques	DR, Semantia	*Examinateur*
BÉCHET Frédéric	Professeur, LIF, Aix Marseille Université	*Directeur de thèse*
LINARÈS Georges	Professeur, LIA, Université d'Avignon	*Directeur de thèse*

LABORATOIRE INFORMATIQUE D'AVIGNON
Université d'Avignon et des Pays de Vaucluse

Remerciements

Je remercie grandement mon directeur de thèse, Frédéric Béchet, qui m'a accompagné tout au long de cette thèse et ce du début à la fin, malgré la distance. Il a réussi à en faire une expérience plutôt agréable, et même si je ne me destine pas à la recherche, je dois beaucoup à ces trois années de thèse.

Je remercie également Géraldine Damnati qui a été une pièce maîtresse de cette thèse, sans qui les expérimentations sur des données réelles n'auraient pas eu lieu et leur valeur pratique d'autant diminuée. Également Benoît Favre, Liva Ralaivola et Lionel Delphin-Poulat avec qui j'ai eu l'occasion de travailler et qui ont contribué aux recherches présentées dans ce manuscrit.

Je tiens à remercier particulièrement mes rapporteurs Yannick Esteve et Guillaume Gravier qui ont eu à évaluer mon travail avec la lourde tâche de devoir lire ce manuscrit.

Je me dois également de remercier la région PACA qui a financé une partie de cette thèse ainsi que Jean Jacques Schneider qui a pris en charge l'autre partie, qui m'a invité dans ses locaux à plusieurs reprises et grâce à qui j'ai pu mettre en pratique certaines expérimentation théoriques.

Je remercie mes parents, sans qui je ne serais pas là, et qui ont enduré mon choix de faire de si longues études loin de ma famille. Mes collègues et amis qui ont eu à me supporter et qui me supportent encore pour certains, et ce bien plus assidument qu'autrefois...

Résumé

La compréhension automatique de la parole est au confluent des deux grands domaines que sont la reconnaissance automatique de la parole et l'apprentissage automatique. Un des problèmes majeurs dans ce domaine est l'obtention d'un corpus de données conséquent afin d'obtenir des modèles statistiques performants. Les corpus de parole pour entraîner des modèles de compréhension nécessitent une intervention humaine importante, notamment dans les tâches de transcription et d'annotation sémantique. Leur coût de production est élevé et c'est la raison pour laquelle ils sont disponibles en quantité limitée.

Cette thèse vise principalement à réduire ce besoin d'intervention humaine de deux façons : d'une part en réduisant la quantité de corpus annoté nécessaire à l'obtention d'un modèle grâce à des techniques d'apprentissage semi-supervisé (*Sef-Training*, *Co-Training* et *Active-Learning*) ; et d'autre part en tirant parti des réponses de l'utilisateur du système pour améliorer le modèle de compréhension.

Ce dernier point touche à un second problème rencontré par les systèmes de compréhension automatique de la parole et adressé par cette thèse : le besoin d'adapter régulièrement leurs modèles aux variations de comportement des utilisateurs ou aux modifications de l'offre de services du système.

Mots clés : Compréhension de la parole, apprentissage automatique, apprentissage semi-supervisé, apprentissage en ligne, self-training, co-training, active-learning, oracle partiel, lazy perceptron.

Abstract

Two wide research fields named Speech Recognition and Machine Learning meet with the Automatic Speech Language Understanding. One of the main problems in this domain is to obtain a sufficient corpus to train an efficient statistical model. Such speech corpora need a lot of human involvement to transcript and semantically annotate them. Their production cost is therefore quite high and they are difficultly available.

This thesis mainly aims at reducing the need of human intervention in two ways: firstly, reducing the amount of corpus needed to build a model thanks to some semi-supervised learning methods (*Self-Training*, *Co-Training* and *Active-Learning*); And lastly, using the answers of the system end-user to improve the comprehension model.

This last point addresses another problem related to automatic speech understanding systems: the need to adapt their models to the fluctuation of end-user habits or to the modification of the services list offered by the system.

Keywords: Speech language understanding, machine learning, semi-supervised learning, online learning, self-training, co-training, active-learning, lazy oracle, lazy perceptron.

Table des matières

Glossaire

Annotation (sémantique) : généralement manuelle : indice donné par un expert relatif au sens d'un énoncé. Ce peut être par exemple de lui attribuer une classe.

Apprentissage automatique : champ d'étude qui vise à faire apprendre une connaissance à une machine de manière automatique.

Biais : erreur récurrente dans l'estimation d'un paramètre faussant un modèle ou un résultat. Le terme vient de la statistique.

Bruit : erreur dans une donnée résultant d'un problème de production, d'acheminement, de traitement, etc. Le terme est tiré de la théorie de l'information développée par (Shannon 1948).

Classe : catégorie à laquelle appartiennent des énoncés. Elle est représentée par un label.

Classification automatique : forme d'apprentissage automatique visant à attribuer une classe (préférentiellement la bonne) à un énoncé donné. On distingue plusieurs types de supervision en classification :

- **Classification supervisée** : les différentes classes possibles sont données par un expert et le système attribue une de ces classes à un énoncé donné.
- **Classification semi-supervisée** : les différentes classes possibles sont données par un expert mais le système tire parti d'énoncés non annotés pour s'améliorer.
- **Classification non supervisée** : le système choisit lui-même les classes, en regroupant les énoncés par similarité par exemple.

Corpus : ensemble de *traces* éventuellement transcrites et annotées. On parle de :

- **Corpus d'apprentissage** : corpus utilisé pour produire ou améliorer un modèle.
- **Corpus de test** : corpus utilisé pour évaluer le modèle.
- **Corpus de développement** : corpus utilisé lors de la construction du modèle pour estimer certains paramètres et/ou éviter le sur-apprentissage, sans toutefois faire entrer en jeu le corpus de test, ce qui biaiserait l'étude.

Disfluences : artefacts caractéristiques de la parole spontanée qui se matérialisent sous la forme de faux départs, de répétitions de mots outils, d'interjections telles « euh », …

Etiquette : voir *Label*.

Label : représentation de l'annotation manuelle d'un énoncé lui attribuant une classe. Autrement dit : le nom d'une classe.

Lexique : ensemble ou sous-ensemble des mots d'un corpus.

Traces : productions unitaires des utilisateurs du système. Par exemple : phrases, énoncés, tours de parole, documents, …

Transcription : représentation textuelle d'un message oral. Une transcription manuelle est faite par un expert, une transcription automatique est produite par un système de Reconnaissance Automatique de la Parole.

Treillis d'hypothèses : graphe orienté dont les arêtes représentent une production (la plupart du temps : un mot) associée à une probabilité. On y cherche généralement le chemin de plus haute probabilité, dont la suite de mots constitue alors l'énoncé transcrit.

Acronymes

FSM – Final State Machine (Machine à états finis)

IA – Intelligence Artificielle

IER – Interpretation Error Rate (Taux d'erreur d'interprétations)

RAP – Reconnaissance Automatique de la Parole

SER – Sentence Error Rate (Taux d'erreur de phrases)

SLU – Spoken Language Understanding (Compréhension de la parole)

SVM – Support Vector Machine (Machine à vecteurs de support)

TALN – Traitement Automatique de la Langue Naturelle

TAP – Traitement Automatique de la Parole

WER – Word Error Rate (Taux d'erreur mots)

Introduction

L'apprentissage automatique est une forme d'Intelligence Artificielle (IA) qui confère à une machine la capacité d'évoluer (ou, parfois, malheureusement, de régresser) en acquérant de nouvelles connaissances. La compréhension de la parole – qu'un enfant acquiert en quelques années – n'est pas une tâche aisée pour une machine. Une machine, tout comme le cerveau humain, doit d'abord reconnaître la parole (c'est-à-dire savoir ce qui a été dit) avant de le comprendre (c'est-à-dire interpréter le sens du message). Le formalisme des grammaires génératives introduit par Noam Chomsky (Chomsky 1957) s'inscrit dans une démarche de théorisation du langage et permet une formalisation qu'il était possible d'apprendre – voire de faire apprendre – à une machine. Dans les années 80, des approches statistiques bien différentes du formalisme linguistique initial ont vu le jour et ont rapidement gagné en popularité du fait de leur facilité de mise en œuvre : plutôt que de faire appel à des experts pour formaliser une langue donnée – travail extrêmement long, pointu et fastidieux – il s'agissait de créer un modèle probabiliste à partir d'un échantillon représentatif de la langue à modéliser – tâche réalisable par une machine – de façon automatique. A partir de là, et notamment grâce à l'augmentation de la puissance de calcul et de la capacité de stockage des machines, il devenait possible de réaliser de nombreuses tâches de Traitement Automatique de la Langue Naturelle (TALN) telles que la traduction automatique, le résumé automatique, la fouille de données, la reconnaissance et la compréhension de la parole.

Le champ d'application du TALN sur lequel porte cette thèse est la compréhension de la parole. Elle se fait usuellement au travers d'un certain nombre d'étapes. La première, optionnelle, peut être de transformer l'expression initiale de la langue en un format maximisant les performances de l'apprentissage automatique. Pour un message oral par exemple, on aura recours à un décodeur dans une étape de Reconnaissance Automatique de la Parole (RAP) afin d'obtenir le message sous forme textuelle voire sous forme d'un treillis d'hypothèses textuelles. En effet la modélisation de contenu sémantique est réputée plus aisée en travaillant sur le texte que sur le signal acoustique, car ce dernier présente une très haute variabilité.

La seconde étape vers la compréhension est la phase d'apprentissage : on construit un modèle qui servira de support à la compréhension. Ce modèle est produit à l'aide de connaissances préalables. Ces connaissances peuvent provenir d'experts du domaine ou, pour des modèles statistiques, d'un certain nombre de données représentatives du phénomène à modéliser (appelé *corpus d'apprentissage*), et il est bien souvent nécessaire de faire appel à des humains pour *annoter* voire *transcrire* ces données.

La troisième étape est l'utilisation de ce modèle pour proposer une compréhension à un énoncé donné. Elle peut utiliser plusieurs modèles et croiser leurs résultats pour affiner la compréhension.

Dans des systèmes déployés – c'est à dire fournissant des services au grand public – s'appuyant sur un traitement automatique de la langue naturelle, et plus précisément en ce qui concerne les systèmes de compréhension de la parole (SLU), une autre étape entre en jeu : l'adaptation des modèles. En effet, de tels systèmes doivent nécessairement s'adapter pour suivre l'évolution des habitudes et du langage des utilisateurs, ainsi que celle des services offerts. Là encore, le système à besoin d'experts qui vont régulièrement mettre à jour les modèles, classiquement en fournissant un nouveau corpus d'apprentissage adapté à l'évolution constatée des utilisateurs ou des services.

Ces quatre grandes phases ou étapes (transformation, apprentissage, compréhension et adaptation) ne sont pas nécessairement séquentielles et peuvent se combiner pour maximiser leur efficacité. Généralement un modèle issu de l'apprentissage est utilisé de nombreuses fois dans une étape de compréhension, et la phase d'adaptation n'intervient bien souvent que lorsque le modèle donne des signes de faiblesse.

Chaque étape peut être accompagnée d'une mesure de performance, nécessaire pour comparer et faire évoluer les techniques. Nous verrons tout au long de cette thèse plusieurs façons de combiner ces étapes, et comment l'utilisation de mesures de performance peut servir à la phase d'adaptation.

Problématique

Un des problèmes majeurs en TALN et qu'obtenir un corpus d'apprentissage est difficile : cela coute cher et n'est pas rapide étant donné que sa constitution implique une large part d'opérations manuelles fastidieuses (collecte, anonymisation, transcription et annotation, pour ne citer que les plus courantes). Ces corpus sont donc bien souvent disponibles en quantité limitée, or la qualité des modèles dépend en grande partie de la taille de ces corpus(Evermann, et al. 2005).

C'est pourquoi un panel de méthodes a été développé afin de réduire le coût de production de tels corpus ou bien à palier le manque de données d'apprentissage. Les méthodes semi-supervisées visent à réduire l'intervention des experts humains, et les méthodes relevant de l'apprentissage interactif cherchent à tirer parti de l'interaction entre le système et les utilisateurs afin d'adapter les modèles aux changements constatés. Dans la première partie de cette thèse je passerai en revue quelques algorithmes usuels servant de support à des méthodes d'apprentissage semi-supervisés, puis je traiterai de l'apprentissage interactif en seconde partie.

Une autre difficulté inhérente à l'objet de l'étude est l'incertitude omniprésente lorsque l'on travaille sur une langue naturelle. Etant naturelle, elle est par essence ambigüe (l'expression de la langue est implicite et se repose sur un savoir partagé) et changeante (les langues vivantes évoluent). De plus, lorsqu'il s'agit de langue naturelle orale, le locuteur introduit du bruit sous la forme de disfluences (répétitions, reformulations, hésitations, faux départs, …). La somme de ces facteurs nous oblige à travailler avec des modèles robustes, c'est-à-dire résistants au bruit. Nous verrons tout au long de cette thèse et plus particulièrement au chapitre 7 comment cette incertitude peut introduire un biais dans les systèmes automatiques, et comment tenter d'y remédier.

La compréhension comme un problème de classification

La classification consiste à ranger des objets dans des catégories. Étant donné un objet (ou *exemple*), un système de classification (appelé *classifieur*) doit être capable de lui attribuer une *classe* (on parle aussi de lui attribuer une *étiquette* ou un *label*)automatiquement.

La compréhension de la parole consiste à donner un sens à un énoncé oral. En cela elle se rapproche d'un problème de classification si l'on considère que l'on choisit un sens parmi N sens possibles, le sens étant la classe attribuée à l'énoncé. Bien entendu un sens est quelque chose d'abstrait qui peut être bien plus complexe qu'une simple étiquette, et qui pourrait par exemple être représenté par un arbre ou un graphe composé d'entités reliées les unes aux autres (Woods 1975) (Béchet, et al. 2010). Cependant, d'une part ramener un problème de compréhension à un problème de classification permet d'utiliser de nombreux outils théoriques et pratiques et offre un cadre expérimental plus aisé à maîtriser. Et d'autre part, l'usage de structures complexes pour représenter le sens d'un message induit une difficulté accrue dans la phase de prédiction de cette structure. Autrement dit, plus un système de compréhension est complexe, moins il est robuste. Or, comme dit précédemment, la robustesse est essentielle lors du traitement de la langue naturelle, surtout orale.

Certains problèmes de compréhension peuvent se simplifier en des problèmes de classification multi-classes (Haffner, Tur et Wright 2003) avec éventuellement un prétraitement ou un post-traitement (e.g. composition d'hypothèses), et c'est d'ailleurs ce qui est fait dans des systèmes de SLU déployés (*AT&T How may I help you?* (Gorin, Riccardi and Wright 1997), *AT&T VoiceTone*© (Gupta, et al. 2006), *France Telecom 3000* (Damnati, Béchet et De Mori 2007)*, Bell financial call center* (Carpenter et Chu-Carroll 1998)). De plus, même si cette thèse se restreint à étudier la compréhension au travers de la classification automatique, certaines stratégies présentées peuvent s'appliquer à des problèmes d'apprentissage plus complexes que de la simple classification.

Il existe deux grandes classes de méthodes de classification : la *classification supervisée* où l'ensemble des classes possibles sont initialement données au système, et la *classification non supervisée* où le système doit trouver lui-même les classes (généralement en regroupant les objets par similarité). La *classification semi-supervisée* se situe entre les deux : le système connait l'ensemble des classes mais peut tirer partie d'exemples non étiquetés (dont on ne connait pas la classe) pour s'améliorer.

En ce qui concerne la classification supervisée ou semi-supervisée, le système construit usuellement un modèle de classification à partir d'un ensemble représentatif d'exemples étiquetés qu'on appelle le *corpus d'apprentissage*, puis il utilise ce modèle afin de classer de nouveaux exemples non étiquetés au vu de leurs caractéristiques (appelées *paramètres*).

Le choix des paramètres et la qualité du corpus d'apprentissage est essentiel. Prenons l'exemple d'un système censé pouvoir classer des véhicules dans une catégorie parmi Camion, Voiture ou Moto. La couleur du véhicule n'est certainement pas un critère pertinent, c'est-à-dire discriminant. Il vaudrait mieux choisir des paramètres tels que les

dimensions, le poids ou le nombre de roues. Cependant, utiliser la couleur comme l'un des paramètres peut ne pas être dénué de sens. En effet, s'il s'avère que la plupart des motos sont noires ou rouges, un véhicule d'une autre couleur à moins de chance d'être une moto.

Pour ce qui est du corpus d'apprentissage, il se doit d'être représentatif. Si le système apprend un modèle sur des exemples présentant une majorité de camions sans remorque (ayant donc uniquement 4 roues) ou ne présentant aucune moto munie de side-car, il y a fort à parier qu'il ne sache que faire d'une information du type « ce véhicule à 4 roues » ou « ce véhicule à 3 roues », voire, s'il s'est fortement basé sur le nombre de roues pour déterminer la classe du véhicule, qu'il fasse de nombreuses erreurs. Dans le premier cas on dit qu'il y a un *biais* sur la représentation des camions, et dans le second que la moto munie d'un side-car n'a pas été modélisée. S'il s'agit maintenant de classer un triporteur, qui n'a que trois roues, le système lui attribuera nécessairement une mauvaise classe vu qu'il ne connait pas la bonne (puisque l'on n'est pas dans le cadre d'un apprentissage non supervisé). Pour remédier à ce problème il est parfois utile de modéliser une classe supplémentaire dite « rejet » (*garbage class* en anglais) pour y mettre de coté les exemples atypiques ou bruités.

De tels problèmes arrivent fréquemment lorsque le corpus d'apprentissage est disponible en faible quantité. Les techniques de classification semi-supervisée permettent de minimiser ces problèmes en tirant parti d'exemples non étiquetés pour corriger, généraliser et renforcer les modèles.

Organisation du document

Dans une première partie je présente et compare plusieurs méthodes de classification appliquées à des tâches de compréhension de la parole :

Après avoir présenté « France Telecom 3000 », un système de compréhension à base de connaissances manuelles déployé depuis 2005 (chapitre 1), je montre comment en dériver un système à base de corpus annoté utilisant des classifieurs statistiques (chapitre 2) afin de pouvoir utiliser deux techniques populaires actuelles visant à réduire l'effort d'annotation : le *Self-training* (chapitre 3), le *Co-training* (chapitre 4).

La seconde partie est classiquement dédiée à la présentation de méthodes originales en commençant par l'intégration d'un critère d'*Active-learning* basé sur un accord inter-classifieurs (chapitre 5).

Dans ce même chapitre j'aborde également une particularité rencontrée dans toutes les applications TALN déployées sur lesquelles j'ai travaillé : la variabilité temporelle des données et son influence sur l'apprentissage et l'évaluation des modèles.

Puis je présente une méthode appliquant l'*Active-learning* au processus de constitution des règles du système à base de connaissances (chapitre 6) et donnant des résultats très positifs.

Enfin je présente deux concepts, dans l'esprit de l'apprentissage en ligne, basés sur l'idée de tirer parti de l'utilisateur pour réduire l'effort d'annotation :

- l'*Oracle partiel* (chapitre 7) qui permet à la fois d'adapter le modèle de compréhension et le modèle de langage sans aucun besoin d'annotation,
- et le *Lazy Perceptron* (chapitre 8) qui est un algorithme capable d'apprendre efficacement des modèles de compréhension dans le contexte de l'*Oracle partiel (Lazy Oracle)*, en tirant notamment parti de contre-exemples, et plus généralement en présence de données partiellement annotées.

Lorsque cela est nécessaire, j'introduis au cœur des chapitres des pré-requis nécessaires tels que des mesures d'évaluation ou des algorithmes.

Première partie

Comparaison de méthodes de classification appliquées à des tâches de compréhension de la parole

Chapitre 1 – **Modèles à base de connaissances**

Certains systèmes de compréhension produisent des hypothèses sémantiques à l'aide de connaissances fournies par des procédés manuels. C'est le cas des premiers systèmes, avant que la puissance de calcul et la capacité de stockage de l'outil informatique ainsi que la disponibilité de corpus ne permette la constitution de modèles probabilistes et plus généralement de modèles à base d'apprentissage automatique.

Les connaissances, souvent sous forme de règles et de grammaires sont écrites par des experts du domaine, c'est-à-dire ayant une connaissance approfondie des différents services proposés par le système et de la façon dont les utilisateurs s'expriment pour les utiliser, ainsi qu'une capacité à formaliser les connaissances (sous forme de grammaires génératives par exemple).

Ce travail n'est pas évident et demande du temps, aussi de tels modèles peuvent prendre plusieurs mois pour voir le jour, depuis la phase de collecte des données jusqu'à leur formalisation. De même, faire évoluer ces modèles demande une bonne expertise du système afin d'augmenter la couverture des règles pour suivre l'évolution du comportement des utilisateurs, ou pour rédiger de nouvelles règles correspondant à de nouveaux services déployés.

De plus, la plupart des analyseurs syntaxiques et sémantiques sont déterministes, et échouent dans l'analyse de phrases agrammaticales ou comportant des disfluences, or ces erreurs syntaxiques et sémantiques sont pléthore dans les système de compréhension orale. Ainsi il faut utiliser des analyseurs capables de fournir des analyses partielles. Ces analyses doivent également être pondérées pour en privilégier les plus probables, au risque d'augmenter la complexité de façon exponentielle.

Un des premiers systèmes conçu pour traiter ces problèmes liés à la compréhension de la parole est le parseur TINA du MIT (Seneff 1992). Un autre exemple, plus récent et proche du grand public, est le système SIRI.

Ces contraintes ont amené les systèmes de compréhension présentés en introduction à adopter une stratégie robuste utilisant deux modèles :

- un modèle syntaxique chargé de projeter les éléments syntaxiques du message dans l'ensemble des concepts en relation avec l'objectif du système,
- et un modèle sémantique permettant de composer ces concepts afin d'en déduire l'interprétation sémantique du message.

Nous verrons que les systèmes fonctionnant à base de règles manuelles, que ce soit pour des raisons historiques ou parce que cela permet une meilleure maîtrise du système, donnent de très bons résultats, et sont ainsi aptes à réaliser des tâches de compréhension avec succès.

Dans ce chapitre je vais décrire un système à base de règles manuelles déployé par France Télécom depuis 2005.

1.1 Le service « France Télécom 3000 »

Le service téléphonique « France Télécom 3000 » (FT3000) est le premier service déployé par France Télécom qui exploite la langue naturelle. Il a été ouvert au public en octobre 2005 et n'a cessé d'évoluer depuis afin de prendre en compte les nouveaux services et offres commerciales. Ce service téléphonique permet aux clients d'obtenir des informations, de souscrire à près de 30 services différents, ainsi que d'accéder à la gestion de leur ligne téléphonique et de leur compte client.

Le système effectue de la compréhension de la parole en temps réel grâce à un processus classique :

- Un module de RAP utilisant un décodeur basé sur un modèle de langage N-gram produit une hypothèse textuelle W (la 1-best) avec un taux d'erreur mot (WER, voir 1.2 page 17) autour de 40%.
- Puis le module de compréhension utilise des règles manuelles pour produire une interprétation sémantique structurée de la forme prédicat-arguments.

Le module de compréhension procède en trois étapes :

- La séquence de mots $W = w_1, \dots w_n$ issue de la RAP est convertie en une séquence de concepts élémentaires $C = c_1, \dots, c_m$ grâce à une grammaire conceptuelle comportant environ 1200 règles manuelles. Par exemple la phrase *« Je cherche des informations à propos de services pour téléphoner moins cher le soir »* correspond à la séquence de concepts *« [Information] [Téléphone] [Pas_cher] [Soir] »*.
- Une autre série de près de 2900 règles manuelles compose les concepts pour construire des interprétations structurées prédicat-arguments. L'ordre des règles est important et la première qui concorde produit l'interprétation retenue.
- Enfin, un processus sélectionne l'interprétation la plus probable parmi les différentes interprétations proposées en se basant sur des règles contextuelles dépendant de l'état du gestionnaire de dialogue. L'exemple précédent aurait généré l'interprétation *« Info(Tarifs, Soir) »*, *Info* étant le prédicat et *Tarifs* et *Soir* les arguments.

Les modèles sémantiques utilisés par le service comportent 400 concepts différents (entités nommées, commandes de dialogue, mots clés) et près de 2000 structures prédicat-arguments possibles qui se décomposent en 54 prédicats et 343 arguments possibles.

1.2 Mesures de performance

Les mesures de performance sont essentielles pour confronter des systèmes et surtout pour améliorer un système donné. Elles servent à la fois à guider et conforter la démarche scientifique et, dans notre cas, à permettre d'améliorer le système de compréhension.

1.2.1 **Word Error Rate (WER)**

Le module de compréhension se base sur la sortie d'un module de RAP. Les données issues de systèmes de RAP sont incertaines, et cette incertitude peut être mesurée en partie. Un tel système peut classiquement produire une unique phrase, qu'on appelle 1-best (c'est la meilleure hypothèse), ou bien une série des N meilleures hypothèses que l'on appelle les N-best.

La mesure de performance la plus courante pour une telle sortie est le Word Error Rate (WER) se base sur la différence entre la phrase produite par le système de RAP et la phrase de référence, il somme trois types d'erreurs :

- Les *insertions* lorsqu'un mot apparaît dans la sortie mais n'est pas présent dans la référence,
- les *délétions* lorsqu'un mot présent dans la référence ne se trouve pas dans la sortie,
- les *substitutions* : lorsqu'un mot de la référence est remplacé par un autre dans la sortie.

Le WER est la somme de ces erreurs divisé par le nombre de mots de la référence :

$$WER = \frac{\#\ insertions + \#\ délétions + \#\ substitutions}{\#\ mots\ dans\ la\ référence}$$

Il est susceptible de dépasser 1 car le nombre d'insertions est illimité.

1.2.2 **Interpretation Error Rate (IER)**

On peut, dans le cas de la présence d'une classe « rejet », parler en termes d'insertions, substitutions et délétions. La classe rejet est attribuée à toute *trace* jugée hors du domaine couvert par l'application, vide, trop bruitée ou toute autre raison valide de rejeter la trace au lieu de tenter de lui attribuer une classe. Ayant des exemples pouvant se retrouver sans classe, le système peut faire trois types d'erreurs :

- des insertions lorsqu'un exemple à rejeter se voit attribué une classe,
- des délétions lorsqu'un exemple valide est rejeté à tort,
- des substitutions lorsque le système attribue la mauvaise classe à un exemple valide.

On parle de Sentence Error Rate (SER) comme la moyenne de ces trois types d'erreurs :

$$SER = \frac{\#\ insertions + \#\ délétions + \#\ substitutions}{\#\ exemples\ totaux}$$

Cependant, plus le système fait face à des exemples à rejeter (dans le cas de FT3000 ce peut être des utilisateurs qui raccrochent immédiatement), plus sa performance est susceptible d'augmenter. En effet, le nombre d'exemples totaux augmente avec le nombre d'exemples à rejeter, et s'ils sont effectivement rejetés par le système, la performance augmente alors que la performance de compréhension en tant que telle peut ne pas avoir varié.

Une autre mesure tenant compte de cette particularité est appelée Interpretation Error Rate (IER) et définie ainsi :

$$IER = \frac{\#\ insertions + \#\ délétions + \#\ substitutions}{\#\ exemples\ valides}$$

Elle ne prend pas en compte le fait qu'un exemple invalide soit effectivement rejeté, en revanche elle continue de compter les insertions et les délétions comme des erreurs. L'IER est donc supérieur au SER et est considéré comme plus représentatif de la performance d'un système du point de vue de l'utilisateur.

1.3 Résultats

Le système FT3000 évalué sur un ensemble de 4554 tours de dialogue, arrive à trouver la bonne interprétation dans 76,5% des cas soit 23,5% d'IER (Table 1).

WER	Substitutions	Insertions	Délétions	SER	IER
40,2%	9,7%	11,9%	1,9%	18,6%	23,5%

Table 1 : performance du système à base de règles manuelles FT3000

Ce résultat peut paraitre médiocre mais il faut garder à l'esprit que le contexte est loin d'être optimal. Les conversations téléphoniques contiennent de la parole hautement spontanée, dont la qualité varie fortement selon le type de matériel utilisé et de réseau emprunté. Le Word Error Rate de 40,2% est là pour le rappeler. Un IER de 23,5% est donc un bon score dans ce contexte. De plus, cet IER est calculé sur un tour de dialogue donné, et non sur l'intégralité du dialogue. Le système FT3000 possède un automate de dialogue capable de clarifier la compréhension à l'aide de quelques tours de plus.

1.4 Conclusion

Les systèmes à base de règles manuelles sont aptes à réaliser des tâches de compréhension de la parole et obtiennent des résultats très honorables. Le système « France Télécom 3000 » en est un bon exemple. Cependant leur mise en place et leur évolution est coûteuse en termes de ressources humaines et de temps.

L'expertise nécessaire à la création, la correction et l'enrichissement de l'ensemble des connaissances manuelles servant au système est souvent très pointue –non seulement en ce qui concerne le formalisme (dans le cas de grammaires par exemple, il faut veiller à ce que certaines règles n'en masquent pas d'autres, et donc avoir une vision globale de l'ensemble des règles) mais également relativement à l'expertise linguistique permettant d'introduire des connaissances à large couverture, principalement pour des systèmes traitant de la parole spontanée du fait des disfluences et de l'agrammaticalité qui par leur caractère aléatoire nécessite de laisser une certaine latitude dans la modélisation du phénomène linguistique qui nous intéresse – ce qui rend de tels systèmes difficilement maintenables, ou tout du moins à un cout important.

L'idée que nous suivrons au travers de ce manuscrit sera de tenter de réduire la nécessité d'expertise humaine en la remplaçant par des techniques et des systèmes

automatiques ou semi-automatiques tout en essayant d'impacter le moins possible sur la performance du système.

Le chapitre suivant présente une alternative aux systèmes à base de règles qui demande moins d'expertise humaine : les systèmes de compréhension à base de modèles statistiques construits à partir de corpus.

Chapitre 2 – **Modèles statistiques à base de corpus**

L'idée générale est – à l'inverse des modèles à base de règles manuelles où des humains tentent de modéliser les données – de laisser des systèmes automatiques modéliser eux-mêmes ces données.

Comme mentionné dans l'introduction, ces systèmes ont besoin d'un corpus d'apprentissage afin de constituer leurs modèles. Ce corpus est à minima transcrit et annoté ce qui est une opération fastidieuse mais demandant moins d'expertise et d'efforts que la constitution d'un modèle entièrement manuel.

Ce chapitre présente un système automatique à base de corpus utilisant des classifieurs comme alternative pour réaliser la tache de compréhension du service FT3000 présenté dans le chapitre précédent.

Après une brève introduction théorique, je présente deux classifieurs, basés sur la régression logistique et le boosting, utilisés pour cette tâche, puis je détaillerai le corpus d'apprentissage tiré du service FT3000 et les performances atteintes par la version automatique du module de compréhension.

2.1 **Cadre théorique**

Comme je l'ai mentionné en introduction, un modèle statistique s'estime à partir de données existantes. Il permet de modéliser le lien existant entre des variables d'entrée, notées usuellement X, et des variables de sorties notées Y.

Dans le contexte de la classification X est un espace vectoriel dont chaque dimension représente un paramètre constitutif des données à classer, et Y est l'ensemble des classes possibles. Un exemple $(x, y) \in X \times Y$ est alors un vecteur de données associé à une classe y.

Afin de déterminer la classe $\tilde{y}$ d'un exemple x donné, le classifieur estime les probabilités à postériori $p(y|x)$ que l'exemple appartienne à une classe donnée, et retient la classe qui maximise cette probabilité :

$$\tilde{y} = \underset{y_i \in Y}{\operatorname{argmax}}\, p(y_i|x)$$

Il existe deux grandes familles de modèles :

- les modèles génératifs qui cherchent à modéliser le lien entre les données et les classes, c'est à dire la probabilité jointe $P(X, Y)$, puis qui déterminent la classe d'un exemple grâce au théorème de Bayes $P(Y|X) = \frac{P(X|Y)\,P(Y)}{P(X)} = \frac{P(X,Y)}{P(X)}$;
- et les modèles discriminants qui se cantonnent à estimer directement la probabilité qu'un ensemble de données appartienne à une classe donnée, c'est à dire $P(Y|X)$.

Les modèles discriminants modélisant directement et uniquement la loi de probabilité permettant de classifier un exemple donné, sans chercher à modéliser la loi (plus complexe) liant X et Y, ont naturellement connu un franc succès dans le domaine de la

classification supervisée et donné lieu à une famille de méthodes telles que le *Boosting*, les arbres de décision, la régression logistique et les SVM.

Les modèles génératifs, quant à eux, sont autant capables de prédire la classe sachant les données que de prédire (ou générer) des données sachant une classe. Ils sont donc intéressants lorsque l'on a besoin de générer des observations, comme en reconnaissance automatique de la parole avec des *Modèles de Markov Cachés (HMM)* ou les *Grammaires non contextuelles probabilistes (SCFG)*. Ils sont également utilisés lorsqu'on a peu de données d'apprentissage mais une bonne connaissance à priori du lien entre X et Y. Dans cette famille on peut citer pour exemple *Naïve Bayes* ou les *Mixtures de gaussiennes (GMM)*.

Enfin, en comparaison des règles manuelles, l'intérêt d'utiliser des modèles statistiques discriminants pour traiter des corpus de parole spontanée, hormis qu'ils nécessitent moins d'expertise, est double. Premièrement, ces modèles sont moins sensibles aux phénomènes de disfluences et d'agrammaticalité caractéristiques de ces corpus, comparé à des règles manuelles qui peinent à prévoir la quasi infinité de formes erronées possibles. Deuxièmement – et nous le verrons plus en détail dans le chapitre 7.3 – ils sont capables de modéliser le biais introduit par la phase de Reconnaissance Automatique de la Parole.

Le cadre applicatif se prête bien à l'utilisation de tels modèles. Le service FT3000 s'apparente à un système de routage d'appels, il n'y a donc pas de relations complexes entre les concepts ou d'information au sein de structures sémantiques à extraire, ce qui permet d'appliquer des méthodes de classification très simples. Les méthodes discriminantes à large marge sont connues pour donner de très bons résultats dans ce cadre (Hsu, Chang et Lin 2003, rev. 2010).

2.2 Classifieurs utilisés

2.2.1 Liblinear

Liblinear (Lin, Fan, et al. 2008) implémente un modèle de régression logistique et également une version optimisée des SVM à noyau linéaire (Vapnik, 2000). Il est très rapide et est particulièrement indiqué dans la classification de documents – et donc de textes – comme le montre l'appendice C.2 du guide de Libsvm (Hsu, Chang et Lin 2003, rev. 2010).

Le principe de la régression logistique est assez proche de la régression linéaire à ceci près qu'on utilise une fonction de la forme :

$$p(X) = \frac{1}{1 + e^{-\alpha - \beta X}}$$

où α et β sont des constantes déterminées par la méthode du maximum de vraisemblance afin de minimiser l'erreur de classification. L'avantage est, au contraire de la régression linéaire, de pouvoir obtenir une probabilité (comprise dans [0..1]) associée à un exemple donné.

Lorsque l'on a plus d'un paramètre, β et X deviennent des vecteurs dont chaque dimension est associée à un paramètre.

2.2.2 **Icsiboost**

Icsiboost est une version open-source du classifieur BoosTexter, basé sur AdaBoost (Freund et Schapire 1996), un algorithme de boosting (Schapire 1990). Icsiboost à l'avantage d'être facile à mettre en œuvre, et possède des facilités telles que la génération de N-gram pour le texte. BoosTexter a été initialement conçu pour la classification de textes (Schapire and Singer 2000).

L'algorithme AdaBoost consiste à construire une multitude d'apprenants faibles, qui combinés formeront l'apprenant fort, qui se résume à un vote pondéré des apprenants faibles. Il fonctionne de manière itérative. A chaque itération, l'apprenant faible qui minimise le nombre d'erreurs de classification est choisi. Les exemples qu'il aura mal classés auront un poids plus fort lors de l'itération suivante, afin que le prochain apprenant faible se concentre sur ces exemples difficiles à classer. Et ainsi de suite.

La Figure 1 donne un exemple naïf illustré, avec deux classes (positive et négative) et deux paramètres (abscisse et ordonnée). Les apprenants faibles sont de simples fonctions linéaires qui séparent l'espace en deux.

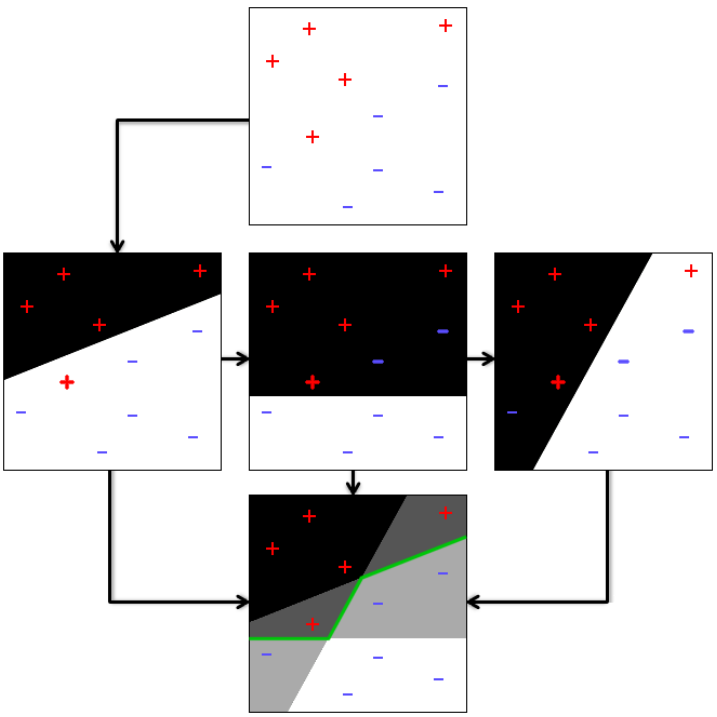


Figure 1 : exemple de combinaison d'apprenants faibles pour résoudre un problème plus complexe

Icsiboost utilise des arbres de décision binaire comme apprenants faibles. Pour les paramètres numériques il s'agit d'un seuil (décision positive si le paramètre est supérieur au seuil), et pour les paramètres textuels il se base sur la présence ou l'absence du S/N-gram (décision positive s'il est présent). Le score d'appartenance d'un exemple à une classe est la somme des poids des arbres binaires de décision positive moins celle des arbres de décision négative.

2.3 Le corpus FT3000

Depuis le déploiement du système FT3000 en 2005, des traces de dialogue ont été collectées, transcrites et annotées en structures sémantiques de la forme prédicat-arguments.

Le corpus d'entraînement est composé de 16 600 dialogues collectés sur une période de 10 mois consécutifs à partir d'un sous ensemble aléatoire tiré du flux de conversations temps réel. Le corpus de test est constitué de 2460 dialogues collectés sur 10 jours consécutifs postérieurs au corpus d'entraînement. Les dialogues sont découpés en tours de parole (chaque intervention de l'utilisateur est traitée à part).

Les deux corpus sont transcrits manuellement mais seul le corpus de test est annoté manuellement avec les structures sémantiques prédicat-arguments. Les annotations sémantiques du corpus d'apprentissage (tags et structure prédcat-arguments) ont été obtenues par application du module de compréhension présenté en 1.1 (page 17) en utilisant en entrée la transcription manuelle de référence. Les modèles de RAP ont été entraînés sur le corpus d'apprentissage et utilisés pour transcrire le corpus de test, avec un taux d'erreur mot de 40,2%. A chaque tour de dialogue est associé une liste de concepts et l'état du dialogue fourni par le gestionnaire de dialogue.

Au final, pour chaque tour de dialogue on dispose de la transcription manuelle, des concepts correspondants, de l'état du gestionnaire de dialogue à ce moment là, de l'index du tour dans le dialogue, et de la structure prédicat-arguments de référence pour le test et celle déduite par le module de compréhension pour l'apprentissage.

Un certain nombre (20% dans le corpus d'entraînement et 15,2% dans le test) de tours de dialogue est inexploitable car vide (l'utilisateur ne prononce rien, il a raccroché, ce qu'il a dit était inexploitable) et d'autres tours (24,6% dans l'entraînement et 17,7% dans le test) n'ont pas d'interprétation associée (hors du domaine). Ces tours de dialogue doivent être rejetés par le système de dialogue. Voir la Table 2 pour plus de détails sur le corpus.

Corpus	Train	Test
# dialogues	16 600	2 460
# tours de dialogue	50 000	5 370
# tours de dialogue non vides	39 973	4 554
# tours de dialogue avec interprétation valide	27 657	3 605
# mots	81 000	13 000
# prédicats	43	40
# structures prédicat-arguments	369	188

Table 2 : détails du corpus FT3000

2.4 Le système automatique

L'idée était de remplacer le module de compréhension à base de règles manuelles par un système utilisant les classifieurs statistiques présentés en 2.1 et 2.2.2 (page 21) entraînés sur le corpus d'apprentissage en 2.3 (page 24).

2.4.1 **Réduire le nombre de classes**

Les modèles statistiques ont des difficultés à modéliser une distribution non uniforme sur un nombre important de classes, or c'est exactement le cas dans le service FT3000. Le nombre de combinaisons prédicat/arguments est trop élevé (plus de 18000 possibilités théoriques et près de 400 en pratique) pour espérer entraîner directement des classifieurs sur ces interprétations structurées.

En effet, certaines combinaisons n'ont – dans le corpus d'apprentissage – qu'un nombre d'exemples très faible (en dessous de la dizaine) aboutissant à une modélisation très fragile de ces combinaisons.

La structure du problème de classification étant naturellement donnée par la définition des cibles, de la forme prédicat/arguments, nous avons choisi de séparer le problème en deux tâches : prédire le prédicat et les arguments séparément, puis fusionner les prédictions pour produire l'interprétation structurée finale. Cela revient à deux problèmes de classification avec un nombre de classes raisonnable (43 prédicats pour le premier problème et 173 arguments pour le second).

Pour chaque exemple, chaque tâche sort l'ensemble des classes prédites (les prédicats p et les arguments a) associées à leurs scores de confiance (respectivement $s_P(p)$ et $s_A(a)$). Connaissant la liste des combinaisons prédicat-arguments valides V, nous sélectionnons le couple qui maximise le produit des scores de confiance des deux hypothèses :

$$\max_{(p,a)\in V} s_P(p) \times s_A(a)$$

2.4.2 **Protocole expérimental**

Tous les paramètres sont utilisés pour construire les modèles (la transcription de référence, la liste des concepts et le contexte de dialogue (numéro du tour et état du gestionnaire de dialogue)). Icsiboost est entraîné sur les bigrammes de la sortie de RAP et des concepts alors que Liblinear les considère comme sacs de mots (les mots sont considérés individuellement, sans que leur ordre entre en compte). Les classifieurs sont utilisés avec leurs paramètres par défaut, le nombre maximal d'itérations de boosting d'Icsiboost est fixé à 1000. Puis les classifieurs sont évalués sur le corpus de test. Les résultats, comparés au système initial sont présentés en Table 3.

SLU	Substitutions	Insertions	Délétions	IER
Règles manuelles	9,7%	11,9%	1,9%	23,5%
Liblinear	14,3%	16,8%	0,8%	32,0%
Icsiboost	13,4%	16,8%	0,8%	31,1%

Table 3 : performance du système de compréhension du FT3000 à base de classifieurs comparé au système à base de règles originel

2.5 **Conclusion**

Nous constatons que le système originel à base de règles manuelles surpasse le système automatique à base de classifieurs. Cela peut s'expliquer par plusieurs raisons :

- tout d'abord par le fait que la quantité de connaissances utilisées dans le système déployé (2900 règles) couvre bien plus d'exemples que ceux présents dans le corpus d'apprentissage, notamment grâce à la capacité de généralisation des experts les rédigeant.
- Puis par la difficulté qu'ont les classifieurs à modéliser la classe « rejet » conduisant à de nombreuses insertions. En effet un message peut être rejeté pour de multiples raisons (RAP médiocre, message hors domaine) et le fait de tous les réunir sous une même classe « rejet » induit une mauvaise modélisation par les classifieurs.
- Enfin, les classifieurs apprennent leurs modèles sur les sorties du système à base de règles qui – même s'il utilise les transcriptions manuelles – produit des erreurs. Ainsi le risque est de modéliser d'avantage le système à base de règles que le phénomène initial.

Cependant l'idée première n'est pas de comparer les deux systèmes pour déterminer lequel est le meilleur, mais simplement de montrer qu'il est possible de remplacer un système à base de règles manuelles par un système automatique à base de corpus qui permet l'utilisation des techniques semi-supervisées présentées tout au long de cette thèse.

Les chapitres suivants présenteront trois de ces techniques : le *Self-training*, le *Co-training* et l'*Active-learning*.

Chapitre 3 – **Self-training**

Le *Self-training* est une technique d'apprentissage semi-supervisée par renforcement qui consiste à faire apprendre à un classifieur sa propre sortie. Les prédictions du classifieur obtenues à partir d'exemples non annotés sont ajoutées aux données annotées et le classifieur est ré-entrainé sur l'ensemble.

Une stratégie commune est de définir un seuil de confiance afin de décider quels exemples seront retenus pour être réappris. Ce seuil s'appuie sur une mesure de confiance donnée par le classifieur.

3.1 **Mesures de confiance**

Les mesures de confiance évaluent à quel point une donnée est fiable, c'est à dire *supposée* proche de la réalité.

Pour un système de compréhension de la langue naturelle orale des incertitudes apparaissent au niveau des données sur lesquelles le module de compréhension se base, c'est-à-dire au niveau de la sortie du module de RAP qui est sujet à interprétation du signal acoustique.

La sortie du système de compréhension est elle aussi sujette à caution, même avec une RAP parfaite, ne serait-ce que parce que les modèles sont limités au domaine d'application du système, souvent restreint, ou tout simplement parce qu'il n'est pas parfait et peut donc produire des erreurs.

Les mesures de confiance permettent au système de prendre des décisions en conséquence, comme faire répéter l'utilisateur si elles sont trop faibles. Elles permettent aussi au système de s'améliorer lui-même de façon automatique, en renforçant les parties du modèle qui génèrent des données peu fiables par exemple.

3.1.1 **Dans les données sources**

Dans 1.2 (page 17) nous avions introduit le WER comme une mesure de l'incertitude des données issues de systèmes de RAP. Cependant le WER est une mesure de performance d'un système de RAP et non une mesure directe de confiance. En effet on a besoin de la transcription de référence pour obtenir le WER. Or cette transcription n'est pas forcément disponible, à fortiori pour un système de compréhension temps réel !

Les systèmes de RAP se basent sur un score acoustique rendant compte de la proximité entre le signal sonore d'entrée et le modèle acoustique et un score linguistique exprimant la plausibilité du syntagme produit au vu de la langue modélisée.

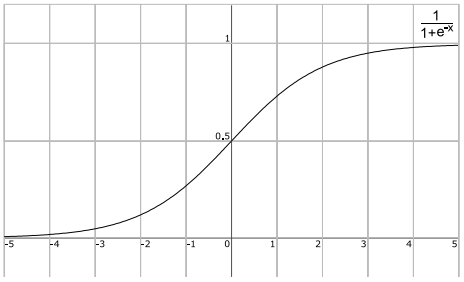
Un système de RAP peut généralement sortir non pas des phrases mais un treillis d'hypothèses, qui est un graphe de mots dont les arêtes portent les probabilités de transition d'un mot à un autre, probabilités étant une savante fusion entre scores acoustique et linguistique. La 1-best est alors le chemin de plus haute probabilité traversant ce graphe.

La combinaison des scores acoustique et linguistique donne une bonne idée de la confiance à accorder à une transcription automatique. Ces scores peuvent-être calculés au niveau de la phrase ou sur chacun de ses mots.

3.1.2 **Dans les sorties des classifieurs**

La plupart des classifieurs fournissent la probabilité que la classe choisie soit la bonne voire associent une probabilité d'appartenance d'un exemple à chaque classe. Cette probabilité a postériori est un bon indicateur de la confiance à mettre dans le choix du classifieur.

Cependant, certains classifieurs – notamment Icsiboost qui fonctionne par combinaison d'hypothèses – ne produisent pas une probabilité d'appartenance à une classe mais un score de classification non normalisé. Il est réputé positif si la classe est un candidat potentiel (dans une classification multilabel on produit l'ensemble des classes de score positif, et seulement la classe de plus haut score sinon). Afin d'obtenir un score entre 0 et 1 qui peut s'apparenter à une probabilité on utilise une fonction de normalisation :



$$P_\alpha(x) = \frac{1}{1 + e^{-\alpha x}} \in [0..1] \,, \forall x \in \mathbb{R}$$

Où x est le score de sortie du classifieur et α un coefficient à déterminer, qui va plus ou moins « écraser » la distribution autour de 0,5.

3.2 **Algorithme**

Soit C un classifieur capable de prédire la classe $\hat{y}_i$ d'un exemple (x_i, y_i) avec une confiance s_i ; un corpus d'apprentissage annoté L et un ensemble d'exemples non annotés U ; et un *seuil* de confiance. Un exemple est un couple (x_i, y_i) associant un vecteur de données x_i à une classe y_i (qui peut être inconnue dans le cas d'un exemple non annoté).

On sépare U en plusieurs partitions U_k étant traitées tour à tour à chaque itération.

```
Entraîner C sur L
Pour chaque U_k
      Pour chaque exemple (x_i, ∅) ∈ U_k
            Prédire ŷ_i avec une confiance s_i
            Si s_i > seuil ajouter (x_i, ŷ_i) à L
      Ré-entrainer C sur L
```

L'intérêt de faire plusieurs passes de self-training est d'exploiter le gain potentiel des modèles à une itération k donnée lors de l'itération $k + 1$.

3.3 Application

3.3.1 Protocole expérimental

J'ai expérimenté le self-training sur plusieurs corpus sans avoir de résultats réellement convaincants ni statistiquement représentatifs. L'une des expériences a été réalisée avec le corpus « France Télécom 1013 » :

Le service FT1013 est un service de routage automatique d'appels. C'est un numéro de service après vente pour la gestion des problèmes techniques sur les lignes téléphoniques fixes des particuliers.

Originellement déployé comme un service de routage DTMF (utilisant les touches numérotées du téléphone) il à progressivement évolué vers un serveur vocal interactif depuis novembre 2007.

Le système est entre autres capable de détecter automatiquement les appels hors domaine et de rediriger les utilisateurs vers le bon service. Il détecte également lorsque les utilisateurs rapportent un problème technique sur leur ligne et lance un diagnostic automatique de la ligne avant de transférer l'appel à un technicien.

Pour des raisons de confidentialité, je ne peux pas détailler le corpus, mais il comporte 66 cibles de routage plus une classe « rejet », 19 400 exemples dans le corpus d'apprentissage et 4 860 dans le test.

corpus	# énoncés	# mots	période
bootstrap	3 200	25k	Mars → Juillet 2008
corpus1	2 700	24k	Octobre 2008 → Avril 2009
corpus2	13 500	130k	Mai 2009
test	4 860	42k	Mai 2009

Le classifieur Icsiboost a été entraîné sur un *bootstrap* de 3 154 exemples puis a tenté de tirer parti des 15 666 exemples non annotés pour renforcer son modèle. Le *seuil* à été défini à 0,9 sur 3 225 exemples pris aléatoirement parmi les 23 225 exemples servant à tester la qualité du modèle. C'est le seuil faisant le moins diverger les modèles sur l'ensemble des exemples.

3.3.2 **Résultats**

Les résultats sont présentés sur la Figure 2. Avec le seuil choisi le classifieur intègre à ses modèles des exemples ayant un taux d'IER de 20,7% en moyenne. La courbe est cependant assez chaotique et oscille entre 40,6% et 41,5% d'IER ce qui ne permet pas d'affirmer que l'apport du self-training est positif ou négatif.

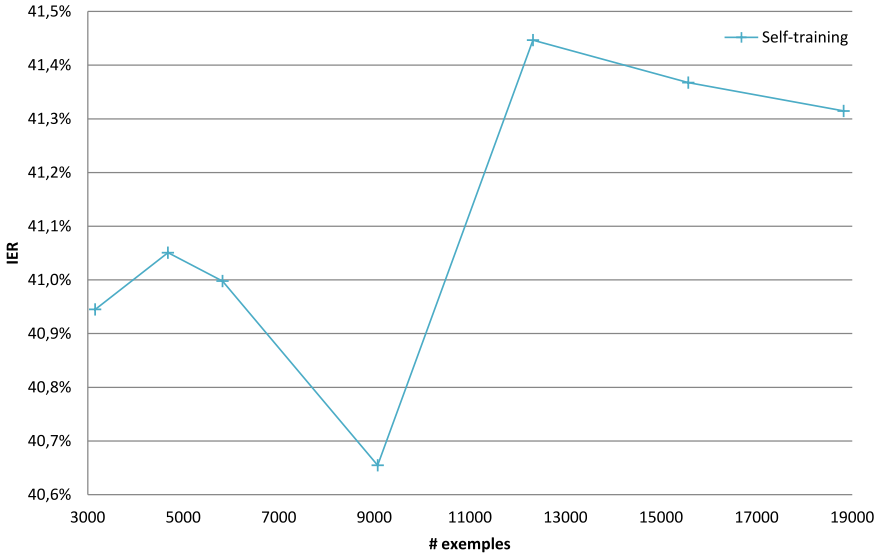


Figure 2 : Expérience de Self-training sur le corpus FT1013

3.4 **Conclusion**

Le self-training semble échouer à apporter de l'information utile au renforcement du modèle. A un taux d'IER proche de 41% on peut se douter que le classifieur génèrera beaucoup d'erreurs et apprendra ces erreurs, ce qui conduira à une divergence des modèles. Même si les exemples réappris sont globalement meilleurs (ils ont un IER de 20,7%) que la performance initiale du classifieur (40,9%), les « bons » exemples qui renforcent les modèles, risquant un sur-apprentissage, ne contrebalancent pas le bruit et les erreurs introduits par les « mauvais » exemples.

Le problème majeur du self-training est l'absence d'information extérieure. Le classifieur est supposé apporter lui-même une information supplémentaire tirée des exemples non annotés en ne se basant que sur sa propre sortie, son score de confiance compris. Afin d'éviter ces problèmes de sur-apprentissage et de divergence, l'idée d'utiliser plusieurs classifieurs différents qui permettraient de s'apporter les uns les autres l'information nécessaire à l'amélioration de leurs modèles à émergé. Cette stratégie a été baptisée *Co-training* et sera développée dans le chapitre suivant.

Chapitre 4 – **Co-training**

Le co-training (Blum et Mitchell 1998) consiste à entraîner plusieurs classifieurs, chacun basé sur une vue du corpus, puis à les améliorer entre eux à l'aide d'une masse importante de données non annotées ; les classifieurs plus à même de classer un exemple donné jouant le rôle de « professeurs » pour les autres.

Le co-training a initialement été appliqué à de la classification de documents HTML en prenant les deux vues que sont le contenu sémantique d'une page et celui des liens pointant vers elle. Peu après (Collins et Singer 1999) et (Pierce et Cardie 2001) évaluèrent le co-training sur des tâches d'apprentissage automatique de la langue naturelle : la classification d'entités nommées et l'identification de phrases nominales. Les vues choisies sont dans un cas l'orthographe et le contexte et dans l'autre les contextes droit et gauche de la fenêtre d'analyse.

L'algorithme original présenté par (Blum et Mitchell 1998) a subi toutes sortes de modifications et d'adaptations à travers la littérature parue depuis :

- La contrainte d'indépendance des vues à été relaxée (Nigam et Ghani 2000)
- Utilisation d'un ensemble d'exemples commun aux classifieurs (Sakkar 2001) ou bien un ensemble distinct par classifieur (Hwa, et al. 2003)(Müller, Rapp et Strube 2002)
- Dans le même ordre d'idées, séparer le corpus dès la première itération (Müller, Rapp et Strube 2002)ou bien laisser les classifieurs apprendre sur le même corpus de base.
- Utiliser un *pool* d'exemples non annotés (comme défini dans (Blum et Mitchell 1998)) géré de plusieurs façons : Aucun *pool*, le non annoté est traité d'un seul bloc (Guz, Cuendet et Tur 2007) ou bien un *pool* de taille fixe réapprovisionné de façon aléatoire (Pierce et Cardie 2001)
- Obliger les classifieurs à classer un certain nombre d'exemples à chaque itération (Denis, Gilleron et Tommasi 2002)
- Respecter la distribution des classes a priori en ne retenant que les $k * f_c$ exemples les mieuxclassés de la classe c de probabilité a priori f_c ($k$étant une constante empirique)

Dans notre cas, les deux classifieurs commencent leur apprentissage sur le même corpus (A), puis se constituent un corpus personnel (G_i) qui grossira à mesure qu'il sera approvisionné par l'autre classifieur.

Le pool d'exemples non annotés (U) est séparé en autant de partitions (U_i) que l'on souhaite d'itérations de co-training, et à chaque itération i, on soumet la partition U_i aux deux classifieurs.

Pour chaque exemple, si un classifieur annonce un score de confiance supérieur à un certain seuil, il y apposera son label et ajoutera cet exemple au corpus d'apprentissage de l'autre classifieur.

Lorsqu'il est difficile de définir plusieurs vues décorrélées sur un ensemble de données, (Goldman et Zhou 2000) proposent d'utiliser plusieurs classifieurs différents basés sur l'ensemble des paramètres P. Et c'est la différence de méthode de classification qui va introduire la complémentarité nécessaire au co-training.

L'objectif est de valider l'idée d'utiliser deux classifieurs différents plutôt que deux vues différentes sur les données, et d'observer le comportement de notre algorithme sur un nombre important de données relativement simples à classer. Dans ce but, nous avons utilisé le corpus de la campagne DÉfi Fouille de Textes 2008 (détaillé en 4.2) qui se porte bien à la tâche : c'est un corpus de langue naturelle (écrite) qui ne comporte pas de grande difficulté de classification (f-score de la baseline autour de 85%) et il est disponible en quantité importante. Après cette validation expérimentale, nous appliquerons notre algorithme au corpus FT1013.

4.1 **Algorithme**

Soit un ensemble de données d'apprentissage A, de données de test T et de données non étiquetées U ; et un ensemble de paramètres P. Un exemple $e \in A$ est un couple $e = (x, y)$ où x est un vecteur de dimension $|P|$ représentant les différentes valeurs de chaque paramètre, et y est la classe de référence de cet exemple.

On découpe U en N partitions de taille égale notées $U_1, U_2, \ldots U_N$.

On entraîne deux classifieurs C_1 et C_2sur A. Un exemple $e = (x, y)$ classé par un classifieur C_i donne $C_i(e) = (\hat{y}, s)$ où $\hat{y}$ est la classe attribuée par C_i avec un score de confiance $s \in [0..1]$.

0n définit un seuil $seuil_i \in [0..1]$ de confiance minimale pour chaque classifieur C_i

On constitue deux ensembles $G_1 = G_2 = A$.

```
Pour k allant de 1 à N
      Chaque exemple e = (x, y) ∈ U_k est classé par C_1 et C_2 :
                  C_1(e) = (ŷ_1, s_1) et C_2(e) = (ŷ_2, s_2)
            Si s_1 > seuil_1, G_2 := G_2 ∪ {(x, ŷ_1)}
            Si s_2 > seuil_2, G_1 := G_1 ∪ {(x, ŷ_2)}
      Entraîner C_i sur G_i
      Tester C_i sur T
```

4.2 **Corpus DEFT08**

DEFT (DÉfi Fouille de Textes) est une campagne d'évaluation ayant lieu chaque année depuis 2005. Elle porte sur la fouille de textes en langue française.

Pour l'année 2008, elle traitait de la classification automatique de textes, et plus particulièrement de la détection du genre et du thème d'articles provenant du journal Le Monde et de l'encyclopédie Wikipedia.

Même s'il ne s'agit pas d'énoncés oraux, l'intérêt de ce corpus est qu'il rend compte d'un problème de classification réel, dont les données ont été annotées par des humains. En effet, les classes ont été définies par Le Monde et Wikipedia et choisies par les auteurs des articles. Il se prête donc à des expériences à la frontière entre apprentissage théorique et linguistique appliquée, et comprend des similitudes avec les problèmes de compréhension que l'on rencontre dans des corpus de langue orale : il existe sans doute des erreurs d'annotation, ou, plus souvent, de l'ambiguïté et des recouvrements entre les classes. Des articles peuvent traiter de plusieurs thèmes mais ne sont rangés que dans un seul (un article de littérature scientifique sera rangé dans Littérature ou dans Sciences ?), et des thèmes peuvent être fortement imbriqués (Sport et Télévision par exemple). On suppose tout de même qu'il existe un lien fort entre le contenu d'un article et son thème principal, et c'est ce qui permettra de construire des modèles pour les classifier.

Le corpus fourni pour l'apprentissage est assez important, de l'ordre d'une dizaine de milliers d'exemples, c'est donc un contexte très favorable à la classification automatique semi-supervisée à l'aide de méthodes statistiques. Deux corpus sont fournis :

- un ensemble de 15 223 articles pour l'apprentissage et 10 596 articles pour le test. Chaque article est associé à une classe parmi : art (ART), économie (ECO), sport (SPO), télévision (TEL) réparties ainsi :

ART	ECO	SPO	TEL
30.41%	8.88%	37.88%	22.82%

- un ensemble de 23 550 articles en apprentissage et 15 693 en test associés aux classes : france (FRA), international (INT), littérature (LIV), sciences (SCI), société (SOC) réparties ainsi :

FRA	INT	LIV	SCI	SOC
14.12%	22.52%	19.43%	27.87%	16.04%

4.3 Mesures de performance : précision, rappel et f-score

Nous avions introduit l'IER en 1.2.2 (page 18) pour mesurer la performance de systèmes de compréhension orale avec une classe « rejet ».

En classification automatique, on utilise couramment les mesures de *précision*, de *rappel* et de *f-score* qui proviennent du domaine de la recherche d'information.

La précision mesure en quelque sorte la qualité des résultats obtenus, et le rappel mesure leur exhaustivité.

On définit la précision et le rappel pour une classe *c* donnée comme suit :

$$precision_c = \frac{\#\ bien\ classés\ en\ c}{\#\ classés\ en\ c}$$

$$rappel_c = \frac{\#\ bien\ classés\ en\ c}{\#\ de\ classe\ c}$$

Il y a une nette différence entre les exemples « classés en *c* » et « de classe *c* ». Un exemple peut être classé *c* sans être de classe *c*, c'est ce qu'on appelle un *faux positif* (il a été faussement classé en *c*). De même un exemple classé *c'* alors qu'il est de classe *c* est appelé un *faux négatif* (il a été jugé à tort étranger à la classe *c*). Un *vrai positif* est un exemple bien classé de classe *c* et un *vrai négatif* un exemple de classe *c'* qui n'a pas été classé en *c* (peu importe la classe dans laquelle il a été effectivement classé).

	Classé en *c* (jugé positif)	**Non classé en *c* (jugé négatif)**
De classe *c*	Vrai positif	Faux négatif
Pas de classe *c*	Faux positif	Vrai négatif

Cette nomenclature devient naturelle en réfléchissant en termes de système de contrôle d'accès : si le système laisse entrer un étranger, c'est un individu jugé positif qui ne l'est pas, autrement dit un faux positif. S'il refuse l'accès à un membre, l'individu est jugé négatif à tort, c'est un faux négatif.

Le f-score est la moyenne harmonique entre précision et rappel :

$$fscore_c = \frac{2 \times précision_c \times rappel_c}{précision_c + rappel_c}$$

La moyenne harmonique permet de pénaliser les systèmes qui ont une trop forte précision par rapport au rappel et vice-versa.

On peut également calculer le f-score sur toutes les classes $\{c_1, \ldots c_N\}$. Il existe plusieurs façons de le calculer :

- En parlant de f-score moyen égal à la moyenne arithmétique des f-scores des classes :

$$fscore = \frac{1}{N} \sum_c fscore_c$$

- Ou égal à la moyenne harmonique des moyennes arithmétiques des précisions et rappels de toutes les classes :

$$fscore = \frac{2 \times \sum_c précision_c \times \sum_c rappel_c}{N \times (\sum_c précision_c + \sum_c rappel_c)}$$

- Ou bien en ne distinguant pas les classes, mais en ne reportant que le nombre de bonnes ou de mauvaises décisions. Dans le cas particulier où tout exemple n'est associé qu'à une et une seule classe, la notion d'étranger disparaît. Précision, rappel et f-score sont alors simplement égaux au taux de bien classés.

Le taux de bien classés tient compte de la distribution des classes, privilégiant celles qui sont majoritairement représentées, alors que le f-score moyen ne tient pas compte de cette distribution et nécessite donc d'avoir un système performant sur l'ensemble des classes modélisées, même les plus minoritaires.

Par la suite, lorsque je parlerai de f-score multi-classe, sans précision, ce sera simplement le taux de bien classés, et lorsqu'il sera question de f-score moyen, ce sera la moyenne arithmétique des f-scores.

4.4 Application

4.4.1 Protocole expérimental

L'expérience a été réalisée sur le corpus DEFT08. Pour simuler une pénurie de données annotées nous avons circonscrit les données d'apprentissage à seulement quelques milliers d'exemples.

Nous retirons les labels du reste des données d'apprentissage, qui sera alors considéré comme la partition non annotée (U).

Nous utilisons Icsiboost et LIBLINEAR comme classifieurs. L'apprentissage et l'évaluation sont réalisés avec la configuration suivante :

- A contient un sous ensemble d'exemples, U contient le reste de l'apprentissage et T est la partition de test originale
- L'ensemble des paramètres P est ici le sac de mots : un vecteur de la taille du lexique qui comporte des 1 dans les dimensions correspondant aux mots présents dans l'exemple, 0 sinon.
- 10 étapes de co-training ($N = 10$)
- 1000 itérations d'Icsiboost pour l'apprentissage des modèles
- Le seuil de confiance d'Icsiboost est fixé à 0.78 et celui de LIBLINEAR à 0.98

Les seuils de confiance sont déterminés afin d'avoir 99% des exemples annotés automatiquement avec la bonne étiquette. Ils ont été choisis empiriquement à partir des modèles appris sur A et évalués sur T. Il aurait été plus rigoureux de les calculer à partir d'un corpus de développement.

Les deux classifieurs servent de témoins dans l'évaluation, en comparant leurs performances avant et après le co-training.

4.4.2 Résultats

Tout d'abord, une baseline est produite à partir des résultats d'Icsiboost et Liblinear lorsqu'ils sont entraînés sur l'intégralité du corpus d'apprentissage pour chaque tâche (Table 4 et Table 5).

Classifieur	Icsiboost			Liblinear		
Classe	Précision	Rappel	F-score	Précision	Rappel	F-score
ART	81,7%	90,4%	85,8%	84,7%	90,2%	87,4%
ECO	84,3%	91,7%	87,9%	82,6%	96,2%	88,9%
SPO	95,2%	89,9%	92,5%	96,8%	91,6%	94,1%
TEL	83,5%	49,3%	62,0%	90,3%	48,0%	62,7%
Toutes			85,4%			86,9%

Table 4 : Performance des classifieurs Icsiboost et Liblinear sur la première tâche de DEFT08.

Classifieur	Icsiboost			Liblinear		
Classe	**Précision**	**Rappel**	**F-score**	**Précision**	**Rappel**	**F-score**
FRA	79,9%	76,1%	78,0%	84,1%	78,9%	81,4%
INT	89,2%	89,9%	89,6%	90,9%	93,2%	92,0%
LIV	92,3%	89,7%	91,0%	92,5%	93,4%	92,9%
SCI	84,6%	89,5%	87,0%	89,1%	89,8%	89,5%
SOC	67,2%	64,9%	66,1%	72,0%	71,6%	71,8%
Toutes			83,8%			86,8%

Table 5 : Performance des classifieurs Icsiboost et Liblinear sur la deuxième tâche de DEFT08.

Lors de la campagne DEFT'08 les deux meilleurs systèmes ont obtenu un f-score de 87.8% sur la première tâche (Trinh et al., 2008) et de 87.9% sur la deuxième (Charton et al., 2008).

En comparaison, ces résultats montrent que l'on obtient facilement des scores très satisfaisants avec des méthodes statistiques, et sans prétraitements, dès lors que l'on a à disposition une quantité suffisante de données d'apprentissage.

Une série d'expériences faisant varier $|A|$ a été menée. Ci-dessous ceux obtenu pour $|A| = 2000$ dans la 1ère tâche et $|A| = 3000$ pour la 2e tâche.

Sur la première tâche proposée par DEFT (Table 6), avec 2000 articles en apprentissage, on constate une progression du f-score global de Icsiboost de 1,1 points, et 1,2 points pour Liblinear grâce au co-training en tirant partie des 13223 autres articles non annotés.

Classifieur	Icsiboost			Liblinear		
Classe	**avant**	**après**	**Gain**	**avant**	**après**	**gain**
ART	81,4%	82,6%	1,2%	81,2%	82,3%	1,1%
ECO	83,5%	84,7%	1,2%	84,7%	85,9%	1,2%
SPO	90,2%	90,9%	0,7%	88,6%	90,4%	1,8%
TEL	50,9%	53,2%	2,3%	40,3%	39,3%	-1,0%
Toutes	81,1%	82,2%	1,1%	80,5%	81,7%	1,2%

Table 6 : F-scores avant, avec 2000 articles en apprentissage, et après co-training en tirant partie des 13223 autres articles de la première tâche de DEFT08.

De même sur la deuxième tâche (Table 7), avec 3000 articles en apprentissage et 20550 non annotés, on note un gain pour quasiment toutes les classes, le f-score d'Icsiboost progresse de 0,8 points, et celle de Liblinear de 0,5 points.

Classifieur	Icsiboost			Liblinear		
Classe	**avant**	**après**	**Gain**	**avant**	**après**	**gain**
FRA	73,7%	75,7%	2,0%	76,9%	77,3%	0,4%
INT	85,6%	85,5%	-0,1%	88,0%	88,3%	0,3%
LIV	88,4%	88,3%	-0,1%	89,1%	89,4%	0,3%
SCI	82,6%	84,1%	1,5%	84,9%	85,4%	0,5%
SOC	58,1%	59,6%	1,5%	61,6%	61,8%	0,2%
Toutes	79,4%	80,2%	0,8%	81,9%	82,4%	0,5%

Table 7 : F-scores avant, avec 3000 articles en apprentissage, et après co-training en tirant partie des 20550 autres articles de la seconde tâche de DEFT08.

En faisant varier |*A*| on obtient des courbes rendant compte du gain apporté par le co-training en fonction de la part de données initialement annotées (Figure 3 et Figure 4).

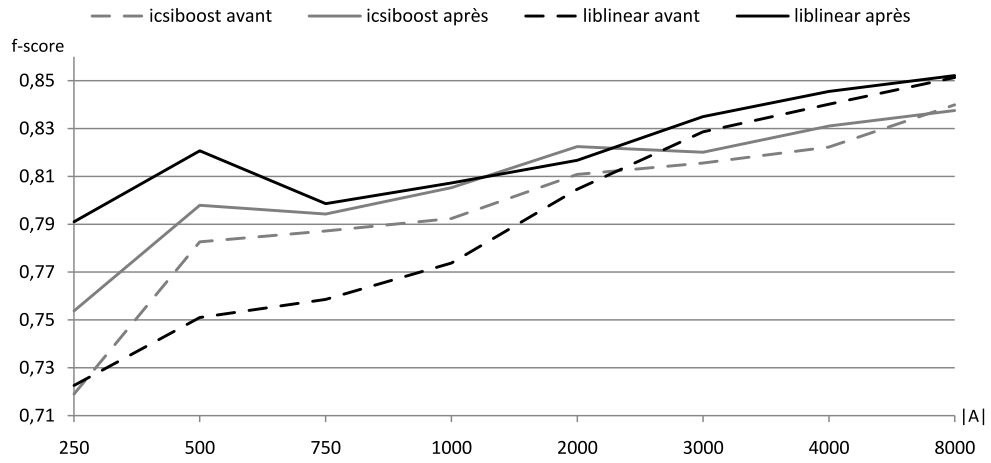


Figure 3 : Gain du co-training en fonction de |A| sur la tâche 1 de DEFT08

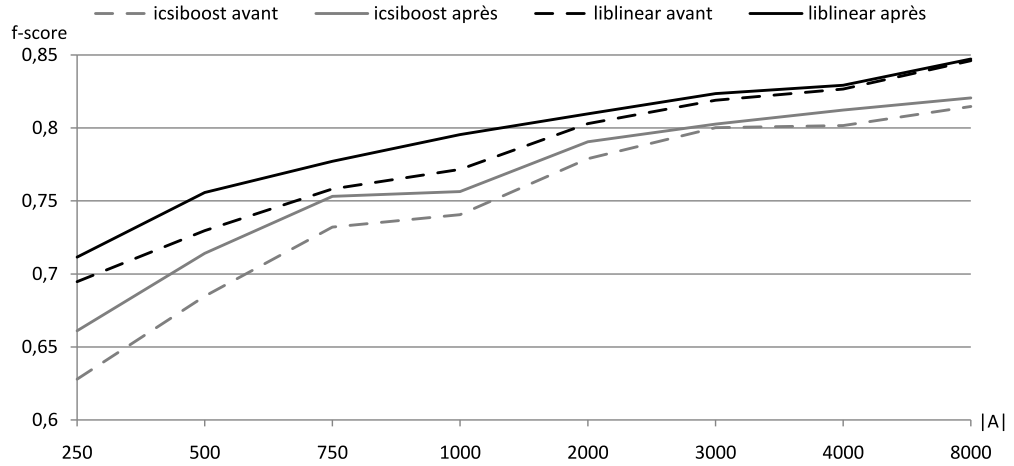


Figure 4 : Gain du co-training en fonction de |A| sur la tâche 2 de DEFT08

Sur la Figure 3, avec 1000 articles en apprentissage par exemple, même si on reste loin des performances obtenues en ayant des annotations parfaites (en prenant l'intégralité du corpus annoté comme dans la baseline), grâce au co-training les deux classifieurs atteignent le score qu'on aurait eu en les entraînant sur plus de 2000 articles, soit avec moitié moins d'annotations manuelles.

On remarque que le gain apporté par le co-training diminue lorsque |*A*| augmente, tendant à devenir nul. Pour faire ressortir ce phénomène, la Figure 5 présente la progression de f-score induite par le co-training, c'est-à-dire la différence entre les courbes des deux figures précédentes.

On voit très nettement que le co-training permet une amélioration des performances des classifieurs pour des tailles de corpus annoté initiales relativement faibles (inférieures à 2000 exemples).

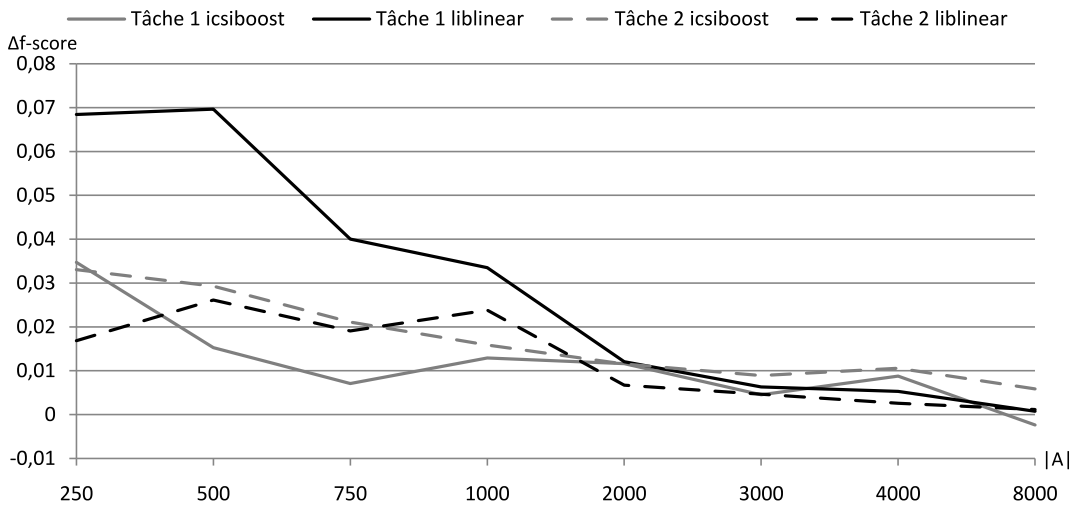


Figure 5 : progression du f-score induite par le co-training en fonction de |A| sur DEFT08

4.4.3 **Application au corpus FT1013**

Cette expérience de *co-training* à été reproduite sur le corpus FT1013 (voir 3.3.1 page 29), afin de la comparer à l'expérience de *self-training* – présentée au Chapitre 3 – et à une expérience *Full Oracle* qui consiste à utiliser tous les exemples avec leur classe de référence en apprentissage.

Le protocole expérimental est le même qu'en 3.3.1, à savoir un *bootstrap A* de 3200 exemples et une partition de test *T* de 4800 exemples. Les résultats obtenus sont présentés en Figure 6.

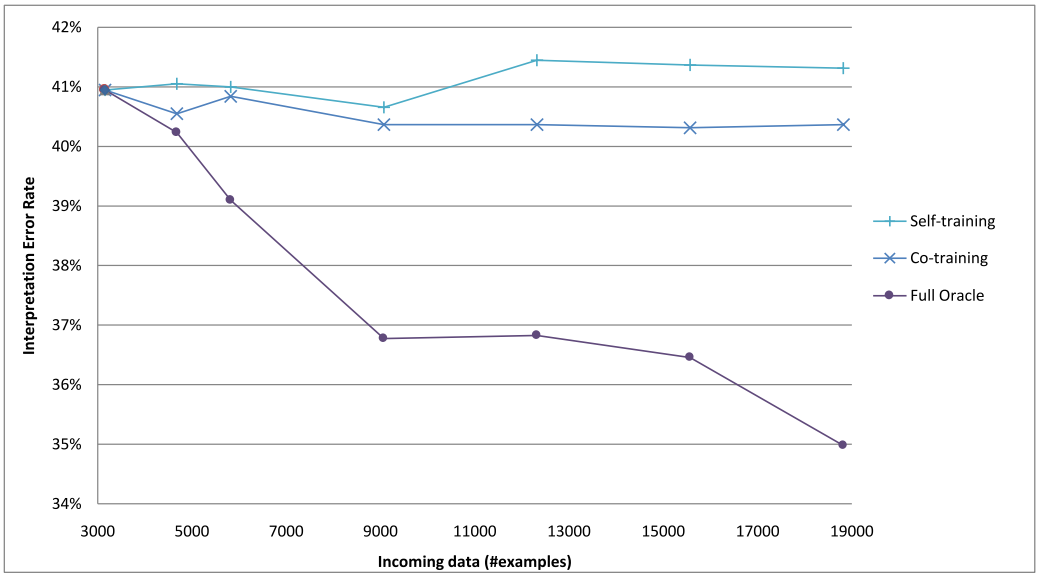


Figure 6 :comparaison de la baisse d'IER induite par le self-training, le co-training et le full Oracle sur le corpus FT1013

On remarque que le gain de 0,6 points du *co-training* sur cette tâche n'est pas statistiquement significatif, et très faible devant les 5,9 points potentiels gagnés par le *Full Oracle*.

4.5 **Conclusion**

Même si l'apport du co-training peut paraître faible, il a été obtenu de façon quasi-gratuite, sans besoin d'intervention humaine. Le gain espéré dépend fortement de la qualité du modèle initial. Alors qu'on pourrait penser qu'un mauvais modèle de départ appris sur très peu de données conduirait à de nombreuses erreurs de classification et donc à un enrichissement maladroit des modèles, les expériences montrent que c'est la situation pour laquelle le co-training est le plus performant : lorsque les données annotées sont disponibles en très faible quantité. C'est à cette même conclusion qu'est arrivé (Pierce et Cardie 2001) sur de la segmentation de groupes nominaux dans le Wall Street Journal, prenant 500 exemples annotés pour en traiter plus de 200 000. On retrouve cette même limite dans notre cas pour de la classification thématique tout comme dans (Wu, et al. 2009) : lorsque les modèles initiaux sont suffisamment performants, les exemples automatiquement annotés par les classifieurs sont moins utiles pour améliorer les performances, voire le co-training devient contre-productif.

Ainsi le co-training n'est pas magique, et n'a pas pu être appliqué avec autant de succès à des tâches de classification plus complexes faisant entrer en jeu des données aussi bruitées que celles des corpus de conversations homme-machine tels celui de France Télécom FT1013. De plus, les systèmes déployés possèdent nécessairement des modèles déjà suffisamment performants, sinon ils ne seraient pas utilisables par le grand public.

Les algorithmes états de l'art prenant en compte les données non annotées dans un processus entièrement non supervisé échouant sur cette tâche. Nous proposons alors dans la partie suivant des algorithmes originaux qui répondent à ce problème.

Quand le nombre de données initiales croit avec la performance des classifieurs, le *self-training* et le *co-training* peinent à apporter de l'information supplémentaire, alors qu'elle existe dans le corpus. On peut penser que cette information ne peut être déduite par les classifieurs car elle n'est pas modélisée. Il est nécessaire de faire appel à un annotateur humain pour l'apporter.

Nous verrons dans le chapitre suivant comment utiliser des annotateurs humains tout en minimisant le coût d'annotation en proposant à l'annotateur des exemples sélectionnés automatiquement parmi un ensemble de données non annotées disponibles : c'est l'*active-learning*.

Deuxième partie

Proposition de méthodes originales de classification automatique pour la compréhension de la parole

Chapitre 5 – **Active-learning**

L'active-learning n'est pas à proprement parler une méthode d'apprentissage semi-supervisé. Elle est intéressante lorsque le coût de collecte du corpus d'apprentissage est faible – donc la quantité de corpus non annoté est potentiellement importante – mais le coût d'annotation est élevé, ce qui est le cas dans la plupart des applications de compréhension de la parole déployées.

Elle intervient lors de la phase de production ou d'enrichissement du modèle en laissant le soin au système de choisir les exemples à annoter pour constituer le corpus d'apprentissage. Classiquement, afin d'avoir un corpus d'apprentissage représentatif, on collecte un certain nombre de *traces* parmi lesquelles on tire aléatoirement des exemples que l'on annote manuellement. Si cette méthode permet d'avoir une bonne représentativité en matière de fréquence d'apparition du phénomène à modéliser, elle ne permet pas d'avoir une couverture optimale (Suendermann, et al. 2008). C'est-à-dire que les phénomènes rares (comme la moto munie d'un side-car) risquent de ne pas apparaître ou d'apparaître en si faible quantité qu'une bonne modélisation de ceux-ci serait impossible. Un autre écueil serait de faire annoter plusieurs fois des traces très similaires voire rigoureusement identiques, ce qui n'enrichirait pas le modèle.

Pour garantir une meilleure couverture et éviter des annotations inutiles, on laisse le système choisir d'annoter les exemples qui apporteront le plus d'information : les plus diversifiés et – s'il y a lieu – les plus éloignés des exemples déjà annotés.

L'active-learning a été utilisé pour enrichir différents types de modèles entrant en jeu dans des systèmes de compréhension de la parole : des modèles acoustiques (Kamm et Meyer 2002) aux modèles de compréhension (Tur, Schapire et Hakkani-Tur 2003) en passant par les modèles de langage (Riccardi et Hakkani-Tur 2005). Dans ces précédents travaux, deux objectifs sont recherchés :

- réduire l'effort d'annotation manuelle des données sans avoir d'incidence négative sur les performances des modèles,
- et trouver l'ensemble optimal d'exemples à annoter parmi la totalité des exemples non annotés disponibles.

L'algorithme général consiste à produire un premier modèle à partir d'un petit ensemble initial d'exemples annotés (un *bootstrap*). Ce modèle est utilisé pour évaluer l'ensemble des exemples non annotés. Un critère d'active-learning permet alors d'extraire un sous-ensemble de candidats à l'annotation manuelle qui seront adjoints au *bootstrap*. L'opération est itérée jusqu'à ce qu'il n'y ait plus de données à annoter ou plus souvent jusqu'à ce qu'un critère d'optimalité soit atteint ou qu'il n'y ait plus assez de ressources pour annoter le corpus.

La plupart des applications de compréhension de la parole déployées peuvent collecter un nombre conséquent de *traces* par jour. L'active-learning est alors intéressant pour

adapter les modèles aux changements de comportement des utilisateurs ou à la création de nouveaux services. Le modèle initial est alors celui disponible lorsque le système a été déployé, et les exemples non annotés sont ceux collectés pendant que le service était en fonctionnement.

5.1 **Algorithme**

L'algorithme que j'ai utilisé est basé sur un critère de sélection par comité de classifieurs(Dagan et Engelson 1995). Les exemples non annotés sont classés par plusieurs classifieurs, et ceux retenus sont ceux qui soulèvent le plus haut taux de désaccord.

En effet, un exemple bien modélisé devrait trouver un consensus unanime auprès des classifieurs. Leur désaccord indique soit que cet exemple n'est pas modélisé, auquel cas il porte une information nouvelle à intégrer aux modèles ; soit qu'il est mal modélisé et donc qu'il faudrait probablement renforcer les modèles pour éviter les erreurs ultérieures.

J'ai introduit le score de confiance des classifieurs dans le processus de sélection, pour que deux exemples ayant le même taux de désaccord soient départagés en sélectionnant en priorité les exemples sur lesquels les classifieurs sont peu confiants.

- Soit un corpus S dont une sous partie $L \subset S$ est annotée et l'autre non $U = S \setminus L$
- Soit deux classifieurs C_1 et C_2 capables de prédire la classe $\hat{y}_i$ d'un exemple x_i avec une confiance s_i : $C_k(x_i) = (\hat{y}_i^k, s_i^k)$.
- Entraîner C_1 et C_2 sur L
- Tant que $U \neq \emptyset$ ou que le nombre d'exemples annotés est suffisant[1]
    - Choisir un sous ensemble $X \subseteq U$
    - Faire classer chaque exemple $x_i \in X$ par chaque classifieur : $C_1(x_i) = (\hat{y}_i^1, s_i^1)$ et $C_2(x_i) = (\hat{y}_i^2, s_i^2)$
    - Ordonner l'ensemble X par désaccord puis par score ascendant : $x_i < x_j$ si $\left(\hat{y}_i^1 \neq \hat{y}_i^2\right) \wedge \left(\hat{y}_j^1 = \hat{y}_j^2\right) \vee \left(s_i^1 * s_i^2 < s_j^1 * s_j^2\right)$
    - Soumettre les δ premiers exemples de X à un annotateur humain, les ajouter à L et les retirer de U
    - Entraîner C_1 et C_2 sur L

L'intérêt de soumettre le non annoté U par sous ensemble X est que chaque itération de l'algorithme va enrichir ou renforcer les modèles et donc permettre une meilleure efficacité dans le ciblage des exemples à annoter lors de l'itération suivante.

La quantité de données à annoter manuellement à chaque itération peut être choisie arbitrairement, en fonction des ressources humaines disponibles par exemple. Le paramètre δ correspondant – qui détermine la taille du sous ensemble utilisé – peut être aussi petit que désiré, mais plus il diminue, plus le nombre d'itérations augmente et donc plus les modèles sont réappris, ce qui peut conduire à des problèmes de performance.

[1]Pour les besoins de l'expérience, l'intégralité de U est consommée, mais dans un contexte réel on s'arrêtera lorsque les ressources allouées à l'annotation sont épuisées.

5.2 **Application**

5.2.1 **Protocole expérimental**

Les expériences ont été menées sur le corpus collecté par le service FT3000 présenté en 2.3 (page 24) : un corpus d'apprentissage de 16600 dialogues et un corpus de test de 2460 dialogues.

Je rappelle que seules les transcriptions manuelles du corpus d'apprentissage ont été réalisées par des humains. Les annotations sémantiques prédicat-argument servant de référence ont été produites par le système à base de règles en utilisant comme entrée les transcriptions manuelles de référence. Elles sont donc plus précises que les sorties initiales du système mais sont tributaires des performances des règles manuelles. En revanche les annotations sémantiques du corpus de test sont manuelles.

Nous considérons que nous avons un Oracle, correspondant à l'annotateur humain, capable de donner la classe y_i d'un exemple x_i. En pratique, la classe donnée par l'Oracle est obtenue par l'application de règles manuelles sur la transcription manuelle. Dans les expériences suivantes, étant donné que nous simulons un tel processus puisque en réalité toutes nos données sont annotées, nous avons arbitrairement choisi la quantité δ_k de corpus à ajouter à l'ensemble L à chaque itération k : $\delta =$ (0,5k[2] ; 0,5k ; 1k ; 2k ; 2k ; 3k ; 4k ; 8k ; 10k ; 10k).

Le critère utilisé pour sélectionner les exemples étant basé sur le désaccord entre différents classifieurs j'ai choisi Icsiboost utilisant AdaBoost, et Liblinear basé sur la régression logistique pour jouer ce rôle. Leurs paramètres d'entrée sont les 2-gram de la sortie de la RAP, les tags et l'état du dialogue pour Icsiboost, et Liblinear utilise le sac de mots de la RAP et du taggueur ainsi que l'état du dialogue.

Dix expériences de sélection aléatoire d'exemples servent de baseline et la moyenne, les valeurs extrêmes et les premier et troisième quartiles pour chaque point de la courbe sont relevés. De même quatre expériences d'active-learning utilisant un bootstrap L de 500 exemples pris aléatoirement dans le corpus d'apprentissage S, et la moyenne et les extrêmes sont reportées sur la courbe.

Une expérience similaire a été réalisée sur le corpus FT1013 (voir 3.3.1 page 29) avec un bootstrap de 3154 exemples et 15666 exemples non annotés séparés en 6 partitions $\delta =$ (1523 ; 1143 ; 3249 ; 3250 ; 3250 ; 3251), et un corpus de test de 4860 exemples (nous avons retiré les tours de parole vide).

Le corpus FT1013 est lui entièrement annoté et transcrit manuellement, nous utilisons cependant les transcriptions automatiques dans les exemples U considérés comme non annotés ainsi que dans le test, afin de coller le plus possible aux conditions réelles.

Douze expériences *Random* servent de baseline à une expérience d'*Active-learning*. Toutes utilisent les 3154 premiers exemples comme bootstrap L.

[2] Pour alléger l'écriture k désigne ici le préfixe kilo (e.g. 0,5k = 500)

5.2.2 **Résultats**

Les résultats présentés en Figure 7 montrent la diminution de l'IER (voir 1.2 page 17) au fur et à mesure que le corpus d'apprentissage L augmente. La taille de L représente l'effort d'annotation manuelle. On voit que l'expérience d'*active-learning* basée sur le désaccord entre les classifieurs et sur les scores de confiance de la classification surpasse rapidement et largement une sélection aléatoire des exemples à annoter. Avec 12 000 exemples (30% du corpus d'entrainement), l'*active-learning* atteint un IER comparable à celui obtenu en utilisant l'intégralité du corpus d'entrainement.

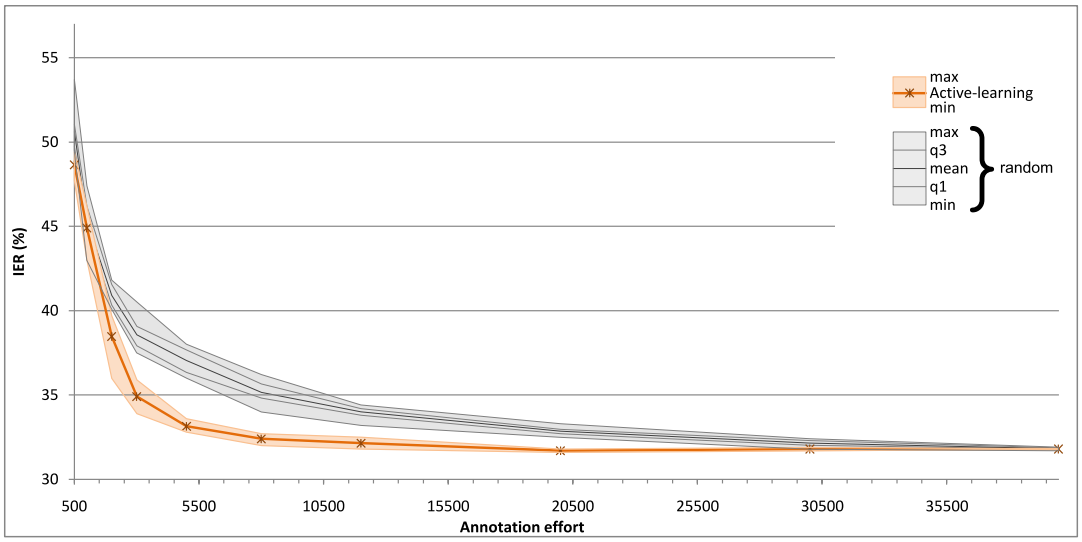


Figure 7 : Interpretation Error Rate en fonction de l'effort d'annotation pour la baseline Random et l'expérience d'Active-learning sur le corpus FT3000

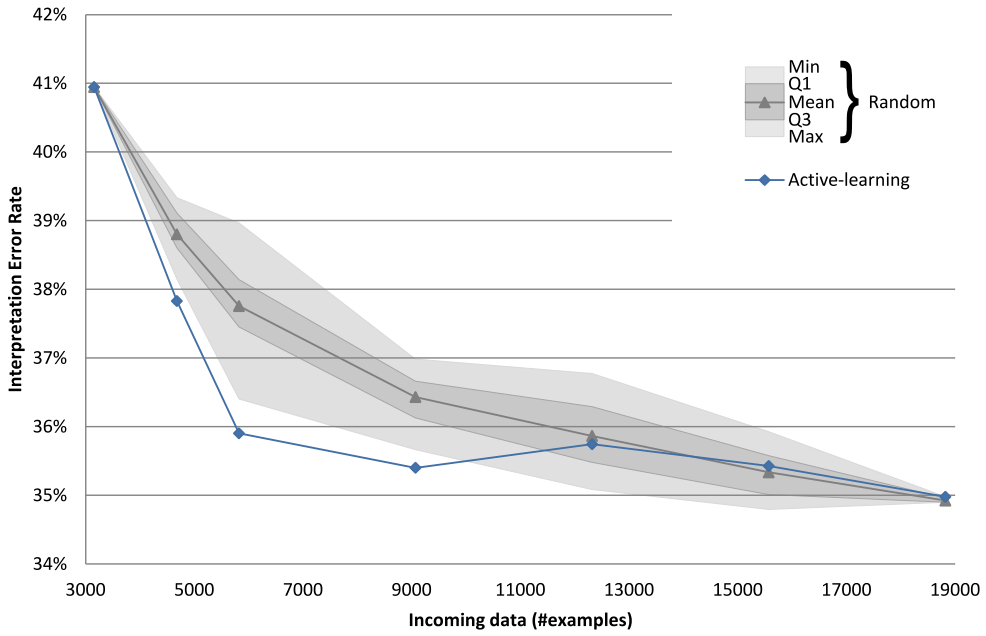


Figure 8 : Interpretation Error Rate en fonction de l'effort d'annotation pour la baseline Random et l'expérience d'Active-learning sur le corpus FT1013

La Figure 8 rend compte des résultats de l'expérience sur le corpus FT1013. On arrive sensiblement au même constat, bien que sur la fin il semble peiner à trouver les exemples les plus discriminants comparé à certains jeux de sélection aléatoire. Ceci peut s'expliquer

par l'éloignement temporel accru des jeux de données comparé au corpus FT3000, ce qui amènerait le processus d'active-learning à sélectionner des exemples apportant effectivement de l'information, mais une information inutile vis-à-vis du test car trop ancienne. Quoi qu'il en soit, l'active-learning étant intéressant lorsque la masse de données disponibles est bien plus importante que ce qu'il est possible d'annoter. On constate par exemple que cette stratégie d'active-learning permet d'obtenir avec 9 000 exemples annotés un modèle de même qualité que si l'on avait sélectionné 16 000 exemples aléatoirement, donc de diviser par deux l'effort d'annotation pour une même qualité de modèle.

5.3 De la variabilité temporelle des données

Dans tous les corpus réalistes sur lesquels j'ai travaillé, que ce soit de la classification d'e-mails, du routage d'appels ou des tâches de compréhension plus complexes, et quelque soit la durée sur laquelle le corpus a été collecté, de quelques semaines à plusieurs mois, les expériences ont révélé une importante variabilité temporelle due aux modifications de comportement des utilisateurs. Cette particularité présupposée a été révélée lors des expériences d'active-learning car l'ordre dans lequel les exemples étaient présentés était crucial.

Les données de test servant pour l'évaluation étant systématiquement postérieures aux données d'apprentissage, nous supposions que les données d'apprentissage temporellement plus proches du test seraient les plus ressemblantes au test. C'est-à-dire qu'il y avait une corrélation entre proximité temporelle et ressemblance.

5.3.1 Protocole expérimental

L'expérience a porté sur le corpus FT3000 (voir 2.3 page 24). L'idée était de présenter au classifieur Liblinear (voir 2.2.1 page 22) les exemples du corpus d'apprentissage dans un ordre prédéterminé dépendant du temps. Liblinear utilisait le sac de mots dans sa configuration par défaut.

Deux expériences dépendantes du temps et une aléatoire ont été menées : l'expérience *Chronological* consistait à prendre les exemples dans l'ordre chronologique, c'est-à-dire les plus éloignés du test en premier ; à l'inverse l'expérience *Reverse chronological* commençait par la fin, c'est-à-dire par les exemples les plus proches du test ; et enfin une série de 10 expériences *Random* présentaient les exemples dans un ordre aléatoire pour servir de *baseline*.

5.3.2 Résultats

Les résultats de la Figure 9 montrent clairement que le pire ordre de présentation des exemples est l'ordre chronologique, pire encore que toute sélection aléatoire d'exemples. A l'inverse, dans l'expérience *Reverse chronological*, les premiers exemples présentés induisent une réduction rapide de l'IER, qui reste meilleur que la sélection aléatoire.

Ces résultats montrent que la proximité temporelle des données d'entraînement et de test implique une similarité entre les exemples qui augmente la performance du classifieur.

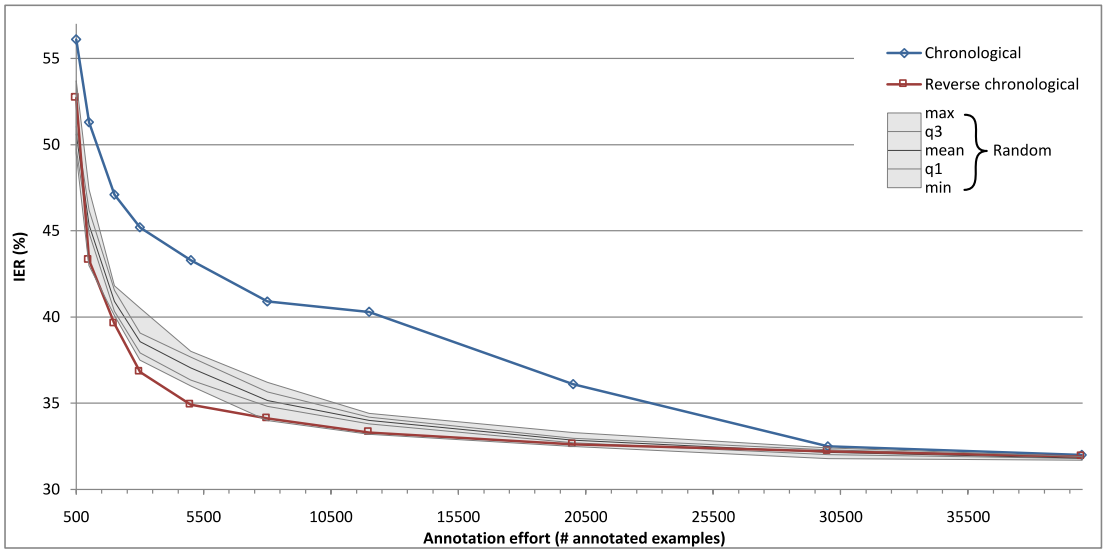


Figure 9 : Influence de l'ordre d'apprentissage des exemples sur le corpus FT3000

À partir de ce résultat assez intuitif et confortant, il serait intéressant de proposer un critère d'active-learning dépendant non seulement de l'apport d'information supposé d'un exemple, mais aussi de sa proximité temporelle avec les données à classer. Autrement dit : privilégier les exemples les plus récents.

5.4 Conclusion

L'active-learning est une méthode très populaire au vu de son intérêt pour réduire le taux de supervision nécessaire à obtenir des modèles de qualité. J'ai montré à l'aide d'un critère basé sur le consensus entre classifieurs et sur la probabilité de classification à posteriori que cette méthode était efficace sur des tâches de classification notamment sur deux corpus réalistes extraits de systèmes de compréhension de la parole déployés où l'active-learning permettait de diviser par 2 voire 4 l'effort d'annotation nécessaire à la production d'un modèle de compréhension de qualité équivalente et surpassait les méthodes de sélection des exemples aléatoires ou basées sur un critère temporel.

J'ai également montré que dans le cadre de systèmes déployés en constante évolution, la distance temporelle entre deux exemples était corrélée à leur similitude. Par conséquent, intégrer l'aspect temporel dans le critère de sélection de l'active-learning, en privilégiant les exemples les plus récents, serait une perspective à explorer.

Le chapitre suivant présentera une contribution originale consistant à appliquer le critère d'active-learning présenté ci-dessus au processus de constitution des règles manuelles par des experts.

Chapitre 6 – **Active-learning appliqué aux modèles à base de règles manuelles**

L'active-learning réussissant à sélectionner des exemples porteurs d'information nouvelle afin que les classifieurs puissent s'en servir pour améliorer leurs modèles, cette information serait-elle aussi profitable à un expert humain produisant des règles manuelles ? Autrement dit, au lieu de sélectionner des exemples pour les soumettre à un classifieur, pourrait-on soumettre ces mêmes exemples à un expert afin d'augmenter la rapidité et l'efficacité avec laquelle il produit des modèles à base de règles manuelles ?

6.1 **Protocole expérimental**

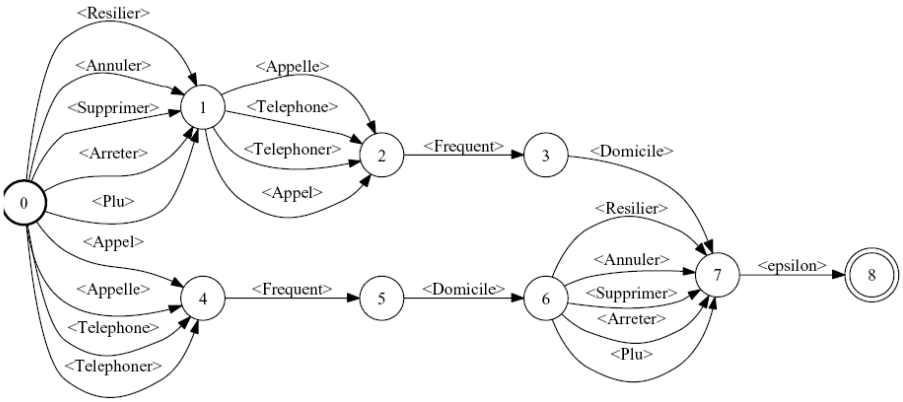
Nous reprenons la stratégie d'active-learning présentée dans le chapitre précédent en simulant l'action d'un expert rédigeant des règles à partir des exemples sélectionnés.

Durant le processus d'active-learning, pour chaque part du corpus annoté *L* produite nous conservons uniquement les règles du modèle manuel s'appliquant sur au moins un exemple de *L*.

Pour cela nous avons généré des *FSM* à partir de l'ensemble des règles manuelles. Par exemple, la règle suivante (**#** représente la disjonction et **&** la conjonction) :

```
((Resilier|Annuler|Supprimer|Arreter|Plu)
# ((Appel|Appelle|Telephone|Telephoner) & Frequent & Domicile))
=>
{Gest(Resilier,Ambi(AtoutsPlus,HeureLocale,ForfaitLocal))}
```

Est représentée par le *FSM* suivant :



Si le résultat de la composition de ce FSM avec les exemples de *L* est non vide, alors on retient cette règle, c'est à dire qu'on estime que l'expert aurait rédigé cette règle à la vue de l'ensemble des exemples *L*.

Ce principe est une approximation puisqu'il est clair que lorsqu'un expert rédige des règles il généralise celles-ci pour embrasser des exemples n'étant pas présents dans le corpus collecté. Cependant nous considérons que c'est une approximation raisonnable et

un bon moyen d'évaluer automatiquement quelles règles peuvent être dérivées des exemples sélectionnés.

Les expériences *Chronological*, *Reverse Chronological* et *Random* ont également été réitérées avec ce mode opératoire.

6.2 Résultats

Le modèle manuel de compréhension produit à partir des grammaires et des règles retenues à chaque itération de l'active-learning ou des autres expériences est toujours évalué sur la sortie de l'ASR du corpus de test et les résultats sont présentés en Figure 10. On constate de grandes similitudes entre ces courbes et celles appliquées au système automatique à base de corpus : la pire situation est l'expérience *Chronological* consistant à prendre les exemples dans l'ordre d'arrivée, l'ordre inverse est bien plus intéressant, même si l'active-learning reste la meilleure méthode qui atteint son IER optimal avec seulement 8000 exemples.

Une autre façon de constater l'efficacité de la production guidée de règles manuelles grâce à cette stratégie d'active-learning est de regarder la quantité de règles dérivées des exemples sélectionnés à chaque itération. Le résultat est présenté en Figure 11. Nous constatons que l'active-learning dépasse largement les autres méthodes de sélection : avec seulement 5000 exemples on obtient presque autant de règles manuelles qu'avec l'intégralité des 39 973 exemples.

On remarque également que l'expérience chronologique est légèrement meilleure que son opposée. On peut en déduire qu'intégrer un critère temporel dans le choix des exemples à annoter permet une amélioration conséquente des performances sur le corpus de test, mais qu'une sélection aléatoire permet une meilleure couverture globale. On imagine donc qu'un certain nombre de règles est devenu inutile vis-à-vis du corpus de test du fait de l'évolution du service.

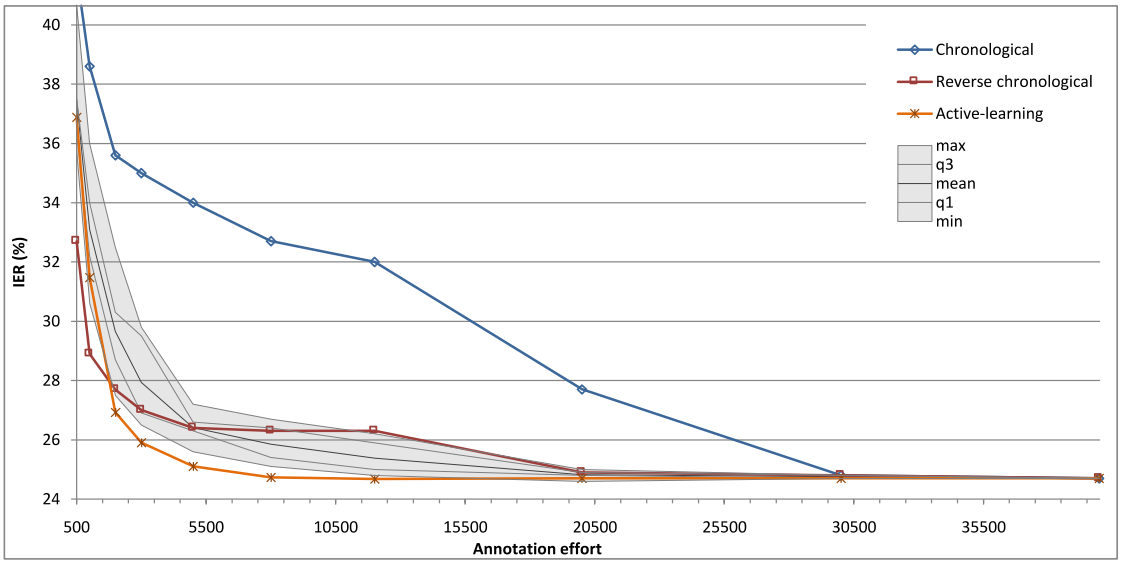


Figure 10 :Qualité de la couverture des modèles manuels du service FT3000 soumis à une simulation de production guidée par une sélection automatique des exemples à examiner.

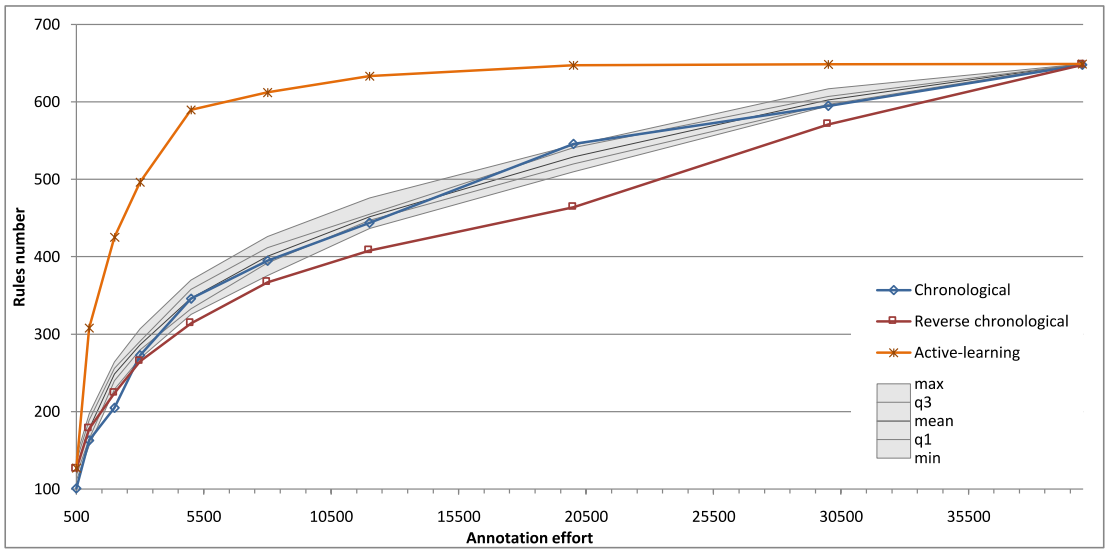


Figure 11 :Evolution du nombre de règles présentes dans les modèles manuels du service FT3000 soumis à une simulation de production guidée par une sélection automatique des exemples à examiner.

6.3 Conclusion

Nous avons montré que l'active-learning pouvait également servir dans le contexte d'un système de compréhension à base de règles manuelles : utiliser un critère d'active-learning afin de sélectionner les exemples à soumettre aux experts produisant les règles manuelles permet une augmentation significative de la rapidité de production des règles et de la qualité des modèles conséquents.

Même si cette expérience n'apporte rien de plus d'un point de vue purement théorique, elle montre que l'active-learning peut s'appliquer à d'autres contextes applicatifs traitant des données issues de la parole, comme, ici, la rédaction de règles manuelles.

Il est à noter que cette stratégie est payante même en ne disposant pas d'une annotation sémantique manuelle, mais en utilisant les annotations produites par les règles. Ceci permet d'utiliser un critère d'active-learning lors de la rédaction de grammaire sans induire un surcoût d'effort d'annotation manuelle. Bien sur ce résultat mériterait d'être confirmé par une expérience réelle avec de véritables experts produisant des règles. Cependant nous espérons que le pouvoir de généralisation des experts pour modéliser des exemples non vus devrait conduire à des performances encore meilleures que celles attendues.

Cependant l'active-learning nécessite toujours l'intervention humaine d'un annotateur, souvent expert. Y aurait-il un moyen de se passer de ce niveau de supervision ? L'idée développée dans le prochain chapitre est la suivante : les systèmes sur lesquels nous travaillons sont déployés, c'est-à-dire qu'une masse importante d'humains les utilisent. Pourquoi ne pas tirer parti de ces utilisateurs et notamment des réactions de ceux-ci pour enrichir et faire évoluer nos modèles de compréhension ? Et comment procéder ?

Chapitre 7 – **Oracle partiel**

Les systèmes de compréhension de la parole – y compris les simples systèmes de routage – basés sur l'apprentissage automatique souffrent de deux problèmes majeurs :

- l'acquisition initiale des données qui au delà de son coût élevé se base sur une représentation des données à priori, de laquelle est dérivée un modèle qui ne sera pas forcément en adéquation avec la réalité ;
- et le problème subséquent qui est l'adaptation de ce premier modèle à la réalité des comportement des utilisateurs, ainsi que son adaptation future lors de l'évolution du service et des habitudes des utilisateurs.

L'expérience que l'on peut tirer des expériences précédentes est l'importance de l'adéquation entre les données servant à l'apprentissage et celles de test, notamment en terme de proximité temporelle. Ainsi, pour qu'un système reste performant il est nécessaire d'adapter ses modèles le plus régulièrement possible pour palier ces variations.

On imagine mal un système d'apprentissage automatique s'adapter comme par magie à de telles variations sans l'aide d'un expert humain pour le guider. L'idée qui se cache derrière l'active-learning vu dans les chapitres précédents consiste à faire appel à un annotateur humain pour amasser de nouvelles connaissances précisément où on en a le plus besoin : sur les exemples qui posent le plus de problèmes à la machine. Cependant un système de compréhension de la parole est en contact permanent avec d'autres humains susceptibles de l'aider à évoluer : les utilisateurs eux-mêmes ! L'utilisateur possède également un avantage face au concepteur du système : il connaît sa propre intention.

En conséquence, les informations que l'on pourrait récolter auprès de l'utilisateur sont dotées d'un important potentiel informatif et sont de plus disponibles en quantité d'autant importante que le système est largement utilisé.

Dans les expériences précédentes nous comptions sur un expert humain pour annoter les exemples que nous lui soumettions. Cet annotateur peut être vu comme un Oracle, c'est-à-dire un être divin capable de donner la véritable réponse à n'importe quelle question telle « la grande question sur la Vie, l'Univers et le Reste », et dans notre cas, de fournir la classe de référence d'un exemple donné.

L'utilisateur du système, lorsqu'on lui propose une compréhension pour un de ses énoncés, est capable de dire si oui ou non nous avons raison. S'il dit que nous avons raison, alors nous connaissons la classe de référence associée à son énoncé : c'est celle que nous avons prédite. En revanche, dans le cas contraire, nous savons que nous nous sommes trompés mais nous ignorons – probablement tout comme lui, qui n'a pas l'expertise suffisante du système pour nous donner la classe de référence – quelle est la bonne classe. La seule chose que nous savons c'est que ce n'est pas celle prédite. En cela, l'utilisateur peut être vu comme un Oracle partiel : parfois capable de donner une réponse non ambigüe du succès de la compréhension, et parfois non.

Afin d'obtenir l'information si oui ou non nous avons trouvé la bonne compréhension nous pourrions le demander à l'utilisateur après chaque tour de parole. Mais bien entendu,

imaginer déployer un système qui demande systématiquement confirmation de ce qu'il a compris serait irréaliste, car l'utilisateur – trop souvent sollicité – serait exaspéré.

Aussi nous pouvons imaginer utiliser d'autres indicateurs de la bonne compréhension comme la réussite apparente de l'objectif : l'utilisateur arrive à la fin du dialogue et est mis en relation avec un humain ou obtient le service qu'il voulait ; ou inversement son échec : l'utilisateur raccroche violemment. Avec un système de dialogue à états, le retour dans un état inférieur peut être vu comme un échec alors que le passage à des états plus profonds est une réussite. D'autres marqueurs tels que le ton de la voix, une rapide enquête de satisfaction, ou tout autre indicateur de la réussite ou de l'échec de la tâche de compréhension pourraient être utilisés.

A ce propos, (Krahmer, et al. 2001) propose des métriques intéressantes comme la longueur des tours de parole, un appui prononcé des mots du discours, une absence de réponse, de multiples corrections et répétitions, l'absence de nouvelle donnée, et de manière générale une variation significative de la complexité des réponses indiquant que l'utilisateur est passé d'un mode de dialogue naturel et efficace à un mode de dialogue dans lequel il utilise d'avantage d'énergie pour se faire comprendre de la machine.

Un principe similaire à l'utilisation de Oracle Partiel été introduit par (Bohus et Rudnicky 2007), baptisé *implicitely supervised learning*. Leur système récolte les réponses positives ou négatives de l'utilisateur pour générer des exemples positifs ou négatifs servant à l'amélioration des modèles. Grâce à cette technique leur système comble de 59 à 78% de la marge maximale de progression donnée par un corpus totalement annoté.

Le principe de l'oracle partiel se rapproche également de l'apprentissage par renforcement (Sutton, 1988) qui consiste à maximiser la récompense de la décision d'un agent étant donné son état au sein d'un processus de décision markovien. On peut définir la récompense comme l'affirmation ou l'infirmation de l'utilisateur en fonction de l'action "prédire la classe C". En revanche l'état courant de l'agent reste difficile à appréhender dans le cadre des systèmes de dialogue oraux, en effet le nombre d'états possibles du dialogue est potentiellement infini.

(Singh, et al. 1999) ont proposé un cadre pour représenter un système de dialogue par un processus de décision markovien. En l'occurrence ils proposent une modélisation partielle des états de l'agent, dépendante du système de dialogue, qu'ils appellent *"états approximatifs"*. Par exemple, pour un système de réservation hôtelière, cela peut correspondre aux différents états de complétude des informations clés collectées par le système (lieu, date, durée, nombre de personnes, etc.). De nombreux systèmes de dialogue possèdent un gestionnaire de dialogue à états qui peut également servir à définir ces *"états approximatifs"*. De ce fait l'apprentissage par renforcement à eu d'avantage de succès dans le domaine des stratégies de dialogue - donc du problème de l'orientation à donner au dialogue - que dans la réelle compréhension de l'intention de l'utilisateur.

7.1 **Algorithme**

On se place dans le cadre théorique pour lequel, pour chaque hypothèse émise par le système, l'utilisateur émet une confirmation ou une infirmation de cette hypothèse. En

réapprenant les exemples dont notre prédiction a été confirmée, on se place dans un cadre proche du self-training, mais avec une différence de taille : Là où le self-training introduit des exemples dont la véracité est uniquement basée sur un score de confiance, ce qui peut introduire un biais important et une « dégénérescence » des modèles, nous avons l'assurance que la classe prédite est là bonne.

Mais l'idée est d'aller encore plus loin et de tirer parti des exemples négatifs, pour lesquels l'utilisateur à infirmé notre hypothèse. Pour cela nous avons besoin d'insérer une connaissance du type « cet exemple n'est pas de classe y ». Nous avons donc choisi d'utiliser des modèles binaires qui supportent naturellement ce type de connaissance.

Le module de compréhension doit être capable de prédire une classe $\hat{y}$ parmi N classes possibles grâce à une série de N classifieurs binaires Θ_y avec $1 \leq y \leq N$. Chaque classifieur Θ_y traitant un énoncée produit un score $\Theta_y(e) = s_y$ représentant sa confiance dans la prédiction de la classe y pour e. La classe finalement prédite $\Theta(e) = \hat{y}$ est celle qui obtient le meilleur score parmi les N classifieurs : $\hat{y} = \text{argmax}_y\, \Theta_y(e)$.

- Soit B un *bootstrap* initial d'énoncés transcrits et annotés. Un modèle de langage de RAP LM_B est entraîné sur ce corpus.
- Soit $U = \{U_1, \dots, U_k\}$ un ensemble de k corpus contenant chacun γ énoncés. Les $k \times \gamma$ énoncés sont transcrits automatiquement en utilisant LM_B.
- Soit $\Phi(e) = y$ un *Oracle complet* retournant le label de référence y de l'énoncé e.
- Soit $\Psi(e, y) \in \{vrai, faux\}$ un *Oracle partiel* retournant une réponse booléenne définie ainsi :

$$\Psi(e, y) = \begin{cases} vrai \text{ si } y = \Phi(e) \\ faux \text{ sinon} \end{cases}$$

Tout d'abord, les N classifieurs Θ_y sont entrainés sur le corpus B préalablement séparé en $2 \times N$ sous ensembles : pour chaque classe y il existe un ensemble S_y^+ d'exemples *positifs* et un ensemble S_y^- d'exemples *négatifs* :

- Chaque exemple $x = (e, t, y)$ est un triplet contenant l'énoncé e, sa transcription t (manuelle t_m ou automatique t_a) et son label sémantique associé y (qui peut être négatif $\bar{y}$ pour « n'est pas de classe y » lorsqu'il est dans S_y^-).
- Pour chaque énoncé $e \in B$ avec $\Phi(e) = y$, l'exemple $x = (e, t, y)$ est ajouté à S_y^+ et des exemples $x = (e, t, \bar{z})$ avec $z \neq y$ sont ajoutés aux ensembles S_z^-. Par soucis de simplicité nous noterons $\neg y$ tous les labels différents de y.
- Chaque classifieur Θ_y est entrainé sur les ensembles S_y^+ et S_y^- (les exemples positifs étant étiquetés y et les exemples négatifs $\bar{y}$).

L'algorithme de mise à jour du système de compréhension grâce à l'Oracle partiel opère en *batch* sur chaque ensemble d'exemples U_i (afin de ne pas réentrainer les modèles à chaque nouvel exemple). Il soumet la prédiction combinée de ses classifieurs $\hat{y}$ à l'oracle partiel Ψ qui la valide ou l'invalide. Dans le cas d'une validation il enrichit son ensemble $S_{\hat{y}}^+$ de ce nouvel exemple et insère les exemples négatifs correspondants dans les autres

ensembles $S^-_{\neg\hat{y}}$. Dans le cas d'une invalidation, il enrichit son seul ensemble $S^-_{\hat{y}}$ de l'exemple en question. Une fois le groupe d'exemple épuisé, les modèles sont réetnrainée.

De manière plus formelle :

```
Pour chaque U_i :
      Pour chaque e ∈ U_i :
            Prédiction de la classe Θ(e) = ŷ
            Si Ψ(e, ŷ) = vrai : ajouter (e, t_a, ŷ) à S⁺_ŷ et (e, t_a, ¬ŷ) aux S⁻_¬ŷ
            Sinon si Ψ(e, ŷ) = faux : ajouter (e, t_a, ȳ) à S⁻_ŷ                (1)
      Réentrainer chaque classifieur Θ_y sur S⁺_y et S⁻_y
```

Les transcriptions automatiques t_a sont obtenues grâce au module de RAP en utilisant LM_B, ainsi la seule supervision nécessaire dans cet algorithme est l'Oracle partiel Ψ, aucune autre intervention humaine n'est nécessaire.

7.2 Application

7.2.1 Protocole expérimental

Les expériences ont été menées sur le corpus FT1013 (voir 3.3.1 page 29). L'une appelée *Oracle partiel A* ne considérant que les réponses positives de l'utilisateur (ne considérant pas la ligne (1) de l'algorithme), et l'autre *Oracle partiel B* prenant en compte les réponses négatives. Icsiboost est utilisé pour jouer le rôle des classifieurs binaires Θ_y. Chaque classifieur utilise les bigrammes de la transcription et réalise 100 itérations de boosting.

Une expérience de référence appelée *Oracle complet* consiste à toujours annoter les données avec la classe de référence, c'est-à-dire à remplacer dans l'algorithme la classe prédite $\Theta(e)$ par la classe de référence $\Phi(e)$. Cette expérience représente la limite supérieure qui peut être atteinte par l'Oracle partiel.

Le *bootstrap B* est constitué de 3154 exemples et les 6 partitions U_i contiennent respectivement 1523, 1143, 3249, 3250,3250 et 3251 exemples qui correspondent à un découpage des différentes campagnes de collecte du corpus.

7.2.2 Résultats

Les résultats présentés en Figure 12 montrent tout d'abord que la stratégie consistant à utiliser les exemples positifs permet un gain final de 1,5 points d'IER, et que la prise en compte des exemples négatifs augmente ce gain de 1,8 points supplémentaires, soit un gain total de 3,3 points ce qui correspond à plus de la moitié du gain maximal représenté par l'*Oracle complet*.

La seule prise en compte des exemples positifs ne permet au classifieur que de progresser sur des exemples où il est déjà bon (car il a prédit la bonne classe), mais en prenant en compte les exemples négatifs, on lui permet d'apprendre de ses erreurs.

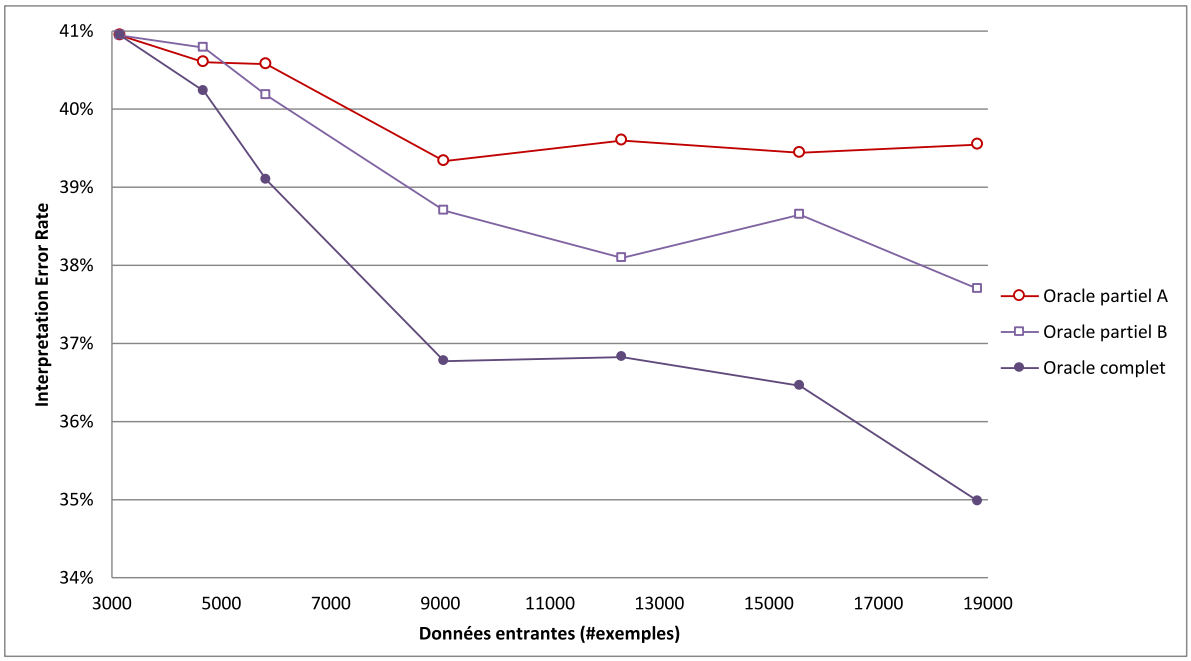


Figure 12 : Progression de l'IER au cours des expériences d'Oracle partiel A et B et d'Oracle complet en fonction de la quantité de données du corpus FT1013.

7.3 Adaptation du modèle de RAP grâce à l'Oracle partiel

Dans la même ligne que l'adaptation du modèle de compréhension à l'aide de l'Oracle Partiel pour réduire l'intervention d'experts, nous pourrions utiliser l'Oracle Partiel afin d'adapter le modèle de langage utilisé pour la RAP. Cela revient à faire du self-training avec le modèle de langage.

L'Oracle Partiel pouvant nous donner une idée de la qualité d'un exemple, puisqu'une bonne compréhension a probablement été émise à partir d'un exemple de plus grande qualité qu'une mauvaise compréhension. En d'autres termes, il y a une corrélation entre la qualité d'un exemple (en termes de WER) et la réponse de l'Oracle Partiel. Cette corrélation a par ailleurs été vérifiée expérimentalement.

L'idée générale est donc d'adapter le modèle de langage de l'ASR par un simple algorithme de self-training utilisant les exemples dont la compréhension a été validée par l'Oracle Partiel.

7.3.1 Protocole expérimental

Notre première idée était d'adapter le modèle de langage tout au long de l'algorithme : à chaque itération i, les exemples U_i sont utilisés pour adapter le modèle qui sera utilisé pour transcrire le prochain ensemble U_{i+1} ainsi que le corpus de test utilisé pour évaluer le système. Cependant cette expérience à conduit à une amélioration non significative des performances du système (de 0.02 points) par rapport à l'expérience *Oracle partiel B*. D'un autre coté, les résultats ont montré que réaliser du self-training sur le modèle de langage à fortement dégradé la qualité de ce modèle, passant d'un WER de 47,7% à 53,24%. Cette augmentation de 5,54 points à elle seule peut expliquer l'échec de l'expérience, la dégradation du modèle de langage compensant l'amélioration de celui de compréhension.

Etant donné que de multiples adaptations successives du modèle tendaient à le dégrader, nous avons choisi d'adapter le modèle en une seule passe, à la fin de l'algorithme. Comme précédemment, seules les sorties de RAP utilisant LM_B sont utilisées pour enrichir le modèle, afin de ne pas introduire de nouvelle supervision.

Deux expériences sont réalisées, l'une produisant un modèle de langage LM_{ALL} à l'aide de tous les exemples non annotés, et l'autre produisant LM_{RIGHT} uniquement à partir des exemples pour lesquels l'Oracle Partiel à répondu *vrai*.

7.3.2 **Résultats**

Modèle de langage	**IER**	**WER**
LM_B	37,7%	47,7%
LM_{ALL}	37,8%	49,6%
LM_{RIGHT}	37,0%	49,8%

Table 8 : Adaptation du modèle de langage avec les exemples issus de l'Oracle Partiel

Les résultats sont présentés en Table 8. Nous observons là encore une augmentation du WER (de 2 points) due à l'itération de self-training, mais cette fois compensée par l'amélioration des modèles de compréhension.

LM_{ALL} n'apporte pas de gain significatif, en revanche on observe un gain substantiel avec LM_{RIGHT} (0,7 points soit 1,8% de gain). Comme prévu, les exemples retenus par l'Oracle Partiel portent une information plus utile et mois bruitée conduisant à une amélioration globale du système.

Cependant le WER de LM_{ALL} et de LM_{RIGHT} est quasiment le même ! Si l'IER du second est plus élevé ce n'est donc pas du fait d'une simple amélioration des qualités des transcriptions mais bien d'une amélioration du système complet. Ce résultat confirme que la performance brute de la RAP n'est pas directement reliée à celle du SLU, et que l'optimisation du WER sans prise en compte des données sur lesquelles il est adapté n'est pas un critère suffisant pour améliorer un système de compréhension. En réalité, une adaptation conjointe (sur les mêmes données) des modèles de langage et de compréhension est la clef pour obtenir de meilleures performances. En clair, le modèle de langage doit être entraîné sur les transcriptions de référence pour s'améliorer, mais doit être adapté à l'aide des transcriptions automatiques pour fonctionner de concert avec le modèle de compréhension.

7.4 **Conclusion**

L'utilisation de l'Oracle partiel dans un contexte d'apprentissage faiblement supervisé permet une amélioration et une adaptation conséquentes des modèles de compréhension sans aucun besoin de supervision humaine si ce n'est celle de l'utilisateur.

Nous utilisons la capacité de l'utilisateur à valider ou invalider l'interprétation que nous lui soumettons. Dans le cas d'une validation, il nous donne la compréhension de référence et nous pouvons enrichir tous les modèles binaires grâce à cet exemple. En revanche dans le cas d'une invalidation, seul le modèle de la classe mal prédite est mis à jour grâce à la

prise en compte de ce contre-exemple. Cependant, l'adjonction de cette information dans les modèles double le gain obtenu ! Cette capacité au système d'apprendre de ses erreurs peut notamment être utile dans le cas d'énoncés comportant des erreurs de RAP, en rendant les classifieurs plus robustes aux erreurs de reconnaissance des mots clés (e.g. déterminant le choix des classifieurs).

Il faut toutefois garder à l'esprit que, dans des conditions réelles, les détections de validation ou d'invalidation de la part de l'utilisateur sont faillibles. (Bohus et Rudnicky 2005) montre par exemple que son système (ou l'utilisateur) peut se tromper sur la validation ou l'invalidation d'une confirmation dans environ 1% des cas pour les confirmations explicites et 13% des cas pour des confirmations implicites.

A un autre niveau, l'Oracle Partiel permet également de sélectionner des exemples de qualité qui peuvent servir à adapter le système au niveau du modèle de langage servant à la RAP. Dans ce cadre, nous avons présenté une expérience permettant d'adapter et d'améliorer le système de compréhension dans son ensemble, encore une fois sans besoin d'autre supervision que celle de l'utilisateur. Une telle adaptation ne dispense pas de faire des campagnes d'annotation manuelle, mais elle permettrait d'en réduire la fréquence.

Même si l'apport de contre-exemples conduit à des résultats très intéressants, comment faire profiter toutes les classes lorsque l'utilisateur donne une réponse négative ?

Le chapitre suivant montre comment utiliser de manière plus efficace les contre-exemples pour améliorer un système de compréhension.

Chapitre 8 – **Apprendre de l'Oracle Partiel**

La particularité de l'*Oracle Partiel* est de fournir un ensemble d'exemples associés à une classe (celle que notre système à prédit) et à une information sur la justesse de cette prédiction. Le corpus ainsi constitué est ce qu'on appelle partiellement annoté. Selon la qualité du système de compréhension, il peut contenir une bonne partie d'exemples négatifs (pour lesquels l'*Oracle partiel* à répondu *faux*). Un système d'apprentissage automatique multi-classe utiliserait cette information pour enrichir le modèle de la classe prédite à tort, sans toutefois en faire directement bénéficier les autres classes.

Le corpus partiellement annoté contiendra également de nombreux exemples positifs. Et bien que de nombreux systèmes peuvent bénéficier d'exemples qu'ils savent déjà classer, certains algorithmes d'apprentissage ne pourront pas utiliser cette information (c'est le cas du perceptron, qui a besoin d'exemples qu'il ne sait pas déjà classer pour corriger son modèle).

Il était alors nécessaire de concevoir un algorithme capable à la fois d'apprendre à partir d'exemples positifs annotés par lui-même, et à partir de tels exemples négatifs – sachant que la classe prédite est erronée – de façon efficace.

De même dans l'*Implicitely supervised learning* proposé par (Bohus et Rudnicky 2007) il est fait usage de modèles binaires obtenus par "*stepwise logistic regression*" (régression logistique capable de sélectionner ses paramètres d'entrée par un processus itératif automatique) et ne sont donc pas réellement des modèles multiclasse. L'idée est donc de proposer un algorithme intrinsèquement multiclasse capable d'apprendre à partir de telles données partiellement annotées.

Une autre motivation est le fait que nos modèles binaires apprenant de l'*Oracle partiel* ne découvrent jamais de nouvelles connaissances, ils se contentent d'améliorer les modèles existants à l'aide des exemples positifs ou négatifs, mais sont incapables de modéliser une classe qu'ils n'ont jamais prédite. L'idée est donc, à la manière du Banditron (Kakade, Shwartz et Tewari 2008), de produire un algorithme capable de faire de l'exploration en plus de l'exploitation.

Avec la collaboration de Liva Ralaivola du Laboratoire d'Informatique Fondamentale de Marseille, nous avons développé le *Lazy Perceptron*, une version modifiée du perceptron, inspirée du Banditron, capable d'apprendre dans un contexte où les exemples d'apprentissage sont partiellement annotés, comme sur un corpus obtenu grâce à l'application de l'*Oracle partiel* présenté au chapitre précédent. Il fonctionne dans sa version actuelle en mode purement exploratoire.

Après avoir détaillé ces différents algorithmes d'apprentissage (*Perceptron*, *Lazy Perceptron* et *Banditron*), je présenterai une expérimentation de ces trois algorithmes sur les données conversationnelles tirées des corpus FT1013 et FT3000.

8.1 **Perceptron**

Le perceptron est un classifieur binaire très simple initialement conçu pour modéliser l'apprentissage neuronal (Rosenblatt 1958).

Les exemples étiquetés positifs ou négatifs, sont projetés dans un espace dont chaque dimension correspond à un des D paramètres. L'algorithme cherche un hyperplan qui séparera l'espace laissant d'un coté les exemples positifs et de l'autre les négatifs. Les données sont réputées linéairement séparables, c'est-à-dire qu'il existe un tel hyperplan optimal w^*.

Soit un corpus d'apprentissage $S = \{(x^n, y^n)\}_{1 \leq n \leq N}$ où $x^n \in \mathbb{R}^d$ est un vecteur à d dimensions représentant les paramètres d'un exemple. $y^n \in \{0; 1\}$ est la classe binaire associée à cet exemple, c'est-à-dire la sortie attendue du perceptron pour l'entrée x^n.

Le modèle du perceptron est représenté par un vecteur w de dimension $d + 1$ (la dimension supplémentaire représente le biais, on adjoint alors une dimension aux vecteurs d'entrée $(x^n)_{D+1} = 1$) qui va évoluer au fil de l'apprentissage. Ce vecteur est le vecteur support de l'hyperplan séparateur. On note w^t l'état de ce vecteur à l'itération t.

Le vecteur w^0 est initialisé à 0, avec des valeurs aléatoires ou avec des valeurs réputées proches de w^* si possible.

A chaque itération, un exemple $x^n \in S$ est soumis au perceptron et $\hat{y}^n = \text{sign}(x_i \cdot w^t)$ est la sortie produite. Le signe du produit scalaire détermine de quel côté de l'hyperplan w^t se situe x^n.

Si $\hat{y}^n \neq y^n$ le perceptron fait une erreur de classification, il modifie alors w^t pour corriger cette erreur et se rapprocher de l'hyperplan optimal w^* :

$$w^{t+1} = w^t + \tau . x^n \text{ avec } \tau = \alpha(y^n - \hat{y}^n)$$

où α est le pas d'apprentissage déterminant la vitesse de convergence de w^t vers w^*.

Le processus est répété jusqu'à ce que tous les exemples de S soient bien classés par w, lorsqu'un nombre d'itérations arbitraire a été atteint, ou lorsque le perceptron passe sous un seuil d'erreur ∂ déterminé ($\forall i, y^n - \hat{y}^n < \partial$).

Ces deux derniers cas de terminaison sont utilisés lorsque l'ensemble de départ n'est pas linéairement séparable, mais lorsqu'on souhaite tout de même obtenir un hyperplan séparant au mieux l'ensemble d'apprentissage.

Dans le cas de problèmes multi-classes on utilise un perceptron multi-classe (Duda et Hart 1973) ou bien plusieurs perceptrons, chacun modélisant une classe contre les autres. Le perceptron a été utilisé dans le domaine du TALN (taggueur de *part of speech* et découpage en phrases nominales) (M. Collins, Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms 2002) et sa simplicité en fait un bon algorithme d'étude (Crammer et Singer 2003).

L'algorithme du perceptron multi-classe pour N exemples appartenant à $Q = \{1, \dots, Q\}$ classes est formalisé en Table 9. τ_q y représente la correction d'erreur appliquée au vecteur w_q modélisant la classe q. Elle est laissée libre dans l'algorithme bien qu'on choisisse souvent de pénaliser équitablement les classes fautives ($\tau_q = -1/|E|$ pour $q \neq y$).

Entrée : $S = \{(x^n, y^n)\}_{n=1}^N$
Sortie : $W = \{w_1, \dots, w_Q\}$
Initialisation : $w_1 = \dots = w_Q = 0$
Répéter T fois :
    Prendre une paire annotée (x, y) dans S
    Définir $E \coloneqq \{q \mid q \neq y \wedge \langle w_q, x\rangle \geq \langle w_y, x\rangle\}$
    Si $E \neq \emptyset$ alors choisir $\tau_1, \dots, \tau_Q$ tels que :

$$\begin{cases} \tau_y = 1 \\ \tau_q = 0 \text{ pour } q \notin E \cup \{y\} \\ \tau_q \leq 0 \text{ pour } q \neq y \\ \sum_{q \in Q} \tau_q = 0 \end{cases}$$

    Pour toute classe $q \in Q$ mettre à jour le modèle :
        $w_q \leftarrow w_q + \tau_q . x$

Table 9 : Algorithme générique du perceptron multi-classe

La procédure d'apprentissage du perceptron est donc une procédure par correction d'erreur puisque les poids ne sont modifiés que si la sortie attendue diffère de la sortie prédite. C'est ce qui conduit à développer un nouvel algorithme pour pouvoir intégrer les exemples positifs fournis par l'utilisateur. En effet, si l'exemple est positif, c'est parce que le classifieur à correctement prédit sa classe. Cet exemple génère donc une sortie conforme à la référence et ne conduit pas à une modification du modèle du perceptron classique.

8.2 **Lazy Perceptron**

Afin de tirer parti d'exemples bien classés, l'idée du Lazy Perceptron est de générer un exemple virtuel à partir d'exemples existants, qui à la particularité de ne pas être bien classé par le perceptron, et qui va donc conduire à une modification du modèle. Cet exemple virtuel est généré à partir d'exemples positifs issus de l'Oracle partiel.

Nous travaillons maintenant avec un ensemble d'exemples partiellement annotés $\tilde{S} = \{(x^n, \tilde{y}^n)\}_{n=1}^N$ où $\tilde{y}^n$ est un Q-uplet avec $\tilde{y}_q^n \in \{0,1,\perp\}$ pour tout $q \in Q$. L'information portée par un exemple est la suivante :

- Si $\tilde{y}_q^n = 1$, alors x^n est de classe q,
- Si $\tilde{y}_q^n = 0$, alors x^n n'est *pas* de classe q,
- Si $\tilde{y}_q^n = \perp$, alors nous ne savons pas si x^n est de classe q ou non.

L'information est partielle seulement lorsqu'aucun des $\tilde{y}_q^n$ n'est égal à 1, on parle alors d'exemple négatif. Autrement c'est un exemple positif de classe q.

Un exemple virtuel $\hat{\mu}_{pq}$ est en réalité la moyenne des exemples positifs x^n de classe q (avec $\tilde{y}_q^n = 1$) se trouvant dans le sous-espace $A_{pq} = \{x \in \mathbb{R}^d \mid \langle w_p, x\rangle \geq \langle w_q, x\rangle\}$ qui contient les exemples mieux classés (à tort) par w_p que par w_q. La Figure 13 résume graphiquement

la situation. Cette génération est faite pour chaque classe $p \neq q$ ce qui conduit à la génération de $N-1$ exemples virtuels par itération.

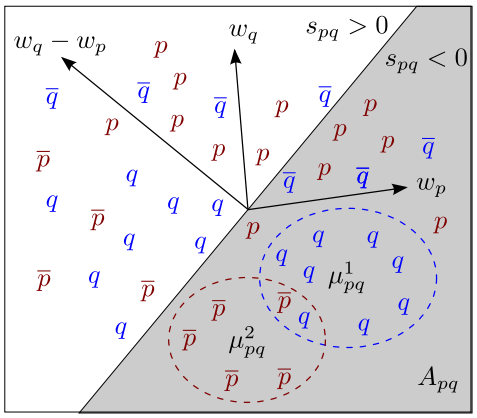


Figure 13 : Illustration de la génération d'exemples virtuel mal classés μ^1_{pq} et μ^2_{pq} à partir d'exemples partiellement annotés q et $\bar{p}$ issus d'un Oracle partiel. Ici $s_{pq} = \langle w_q - w_p, x \rangle$ avec $x \in \mathbb{R}^2$

Les exemples générés étant de fait mal classés par w_p et devant être des représentants de la classe q (donc devant être bien classés par w_q), la procédure de mise à jour des modèles consiste simplement à avoir $\tau_q = 1$, $\tau_p = -1$ et $\tau_r = 0$ pour $r \neq p$ et $r \neq q$. L'algorithme résultant est présenté en Table 10.

Il est également possible de générer des exemples virtuels à partir des exemples négatifs en remplaçant (2) par $\hat{\mu}^2_{pq} = \frac{1}{N}\sum_{n=1}^{N} \mathbb{I}\{x_n \in A_{pq}\} x^n - \frac{Q}{2N}\sum_{n=1}^{N} \mathbb{I}\{\tilde{y}^n_q = 0 \wedge x_n \in A_{pq}\} x^n$.

Voire de combiner ces deux exemples virtuels : $\hat{\mu}^{\alpha}_{pq} = (1-\alpha).\hat{\mu}^1_{pq} + \alpha.\hat{\mu}^2_{pq}$

```
Entrée : S̃ = {(x^n, ỹ^n)}_{n=1}^N
Sortie : W = {w_1, …, w_Q}
Initialisation : w_1 = … = w_Q = 0
Répéter T fois :
      Choisir un label q ∈ Q aléatoirement (label correct)
      Pour tout p ≠ q (label incorrect)
      Générer un exemple virtuel μ̂_pq à partir des exemples positifs :
            μ̂¹_pq = (Q/N) Σ_{n=1}^N 𝕀{ỹ^n_q = 1 ∧ x_n ∈ A_pq} x^n        (2)
      Puis mettre à jour le modèle :
            w_q ← w_q + μ_pq
            w_p ← w_p − μ_pq
```

Table 10 : Algorithme du Lazy Perceptron multi-classe apprenant à partir d'exemples virtuels générés à partir d'exemples positifs issus d'un Oracle partiel. Note : $\mathbb{I}\{cond.\} = \begin{cases} 1 \text{ si } cond. \text{ est vraie} \\ 0 \text{ sinon} \end{cases}$

La convergence de l'algorithme d'apprentissage (Ralaivola, et al. 2011) repose sur le tirage équiprobable d'un label correct q parmi Q. En cela il se rapproche du mode exploratoire du Banditron qui produit également un label aléatoire sans prendre en compte le modèle. Il est apparu intéressant de comparer ces deux algorithmes en ligne.

8.3 Banditron

Le Banditron (Kakade, Shwartz et Tewari 2008) est une version légèrement modifiée du perceptron. Il fonctionne selon deux modes : exploration et exploitation.

- En mode exploitation il se comporte comme un perceptron multi-classe classique.
- En mode exploration il produit des sorties aléatoires afin de générer des exemples mal classés mais pour lesquels la classe est connue : à chaque itération il prédit une classe $\hat{p}$ mais tire une autre classe aléatoire $\tilde{p}$ parmi une distribution P biaisée vers la classe prédite. Il utilise alors l'information de l'Oracle afin de mettre à jour son modèle.

Le paramètre γ est utilisé afin de choisir un compromis entre exploration et exploitation. L'algorithme est détaillé en Table 11.

Il est à noter que le Banditron – contrairement au Lazy Perceptron – ne réutilise pas les exemples passés, il ne traite un exemple qu'une seule fois.

Entrée : $S = \{(x^n)\}_{n=1}^{N}, \gamma \in [0..0,5]$
Sortie : $W = \{w_1, \dots, w_Q\}$
Initialisation : $w_1 = \dots = w_Q = 0$
Pour chaque exemple x :
  Prédire $\hat{p} = \text{argmax}_{q \in Q} \langle w_q, x \rangle$
  $\forall q \in Q$, construire la distribution $P(q) = (1-\gamma)\mathbb{I}\{q = \hat{p}\} + \frac{\gamma}{Q}$
  Tirer $\tilde{p}$ parmi P et prédire $\tilde{y}_{\tilde{p}} \in \{0,1\}$ en conséquence
  Définir $\tau_1, \dots, \tau_Q$ tels que :

$$\begin{cases} \tau_{\hat{p}} = -1 \\ \tau_{\tilde{p}} = \frac{\tilde{y}_{\tilde{p}}}{P(\tilde{p})} \\ \tau_q = 0 \text{ pour } q \notin \{\hat{p}, \tilde{p}\} \end{cases}$$

  Pour toute classe $q \in Q$ mettre à jour le modèle :
    $w_q \leftarrow w_q + \tau_q . x$

Table 11 : Algorithme du Banditron

8.4 Protocole expérimental

Les expériences sont menées sur les corpus FT3000 (voir 2.3 page 24) et FT1013 (voir 3.3.1 page 29). en ne conservant que les 8 classes les plus représentées pour FT3000 et 10 pour FT1013 ; ceci afin d'éviter qu'un déséquilibre trop important des classes n'entraîne de trop nombreuses erreurs de la part du Lazy Oracle qui utilise un tirage équiprobable et de la part du Banditron qui ne tient pas non plus compte de la distribution des classes a priori.

Les corpus sont circonscrits à 10000 exemples en apprentissage et 3800 (pour FT3000) et 1000 (pour FT1013) exemples dans le corpus de test. Le reste constitue un corpus de développement pour estimer γ dans le Banditron.

Chaque expérience est relancée avec un nombre d'exemples N. Pour chaque taille de corpus, 20 expériences sont lancées afin d'en relever la moyenne et l'écart-type. Six expériences sont comparées :

- Le *Banditron* : avec un ratio exploration/exploitation γ estimé à 0,15
- Un perceptron classique appelé *Regular* entraîné uniquement à partir des exemples positifs produits par l'Oracle partiel, faisant office de *baseline*.

- Trois expériences utilisant le *Lazy Perceptron* entraîné sur un ensemble de N exemples $\tilde{S}$, chaque exemple étant soumis à l'Oracle partiel avec une des Q classes tirée de façon équiprobable, conformément à la preuve de convergence de l'algorithme :
    - *Lazy I* : ne génère que des exemples virtuels $\hat{\mu}_{pq}^{1}$ à partir des exemples positifs,
    - Lazy *II* : ne génère que des exemples virtuels $\hat{\mu}_{pq}^{2}$ à partir des exemples négatifs,
    - *Lazy III* : génère des exemples virtuels étant des combinaisons des deux précédents $\hat{\mu}_{pq}^{\alpha}$ (Le paramètre α est estimé afin de minimiser la variance de $\langle w_q - w_p, \hat{\mu}_{pq}^{\alpha} \rangle$).
- Un *Oracle total* est un perceptron entraîné sur un ensemble totalement annoté. Il matérialise le plus haut résultat admissible que nous essayons d'atteindre avec les autres méthodes.

8.5 Résultats

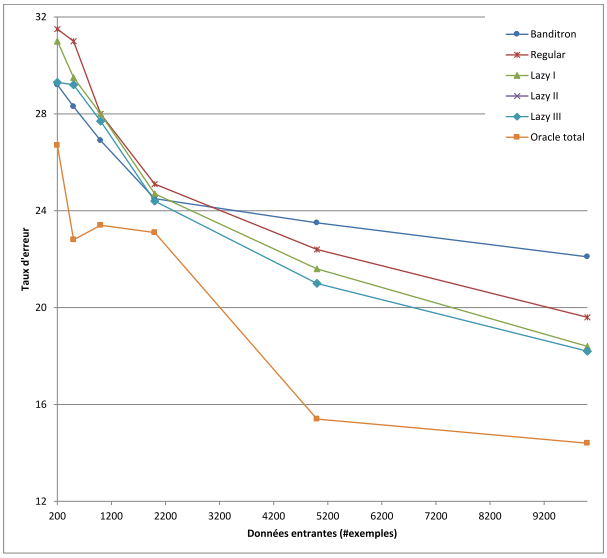


Figure 14 : Résultats des différentes expériences autour du Lazy Perceptron sur le corpus FT3000

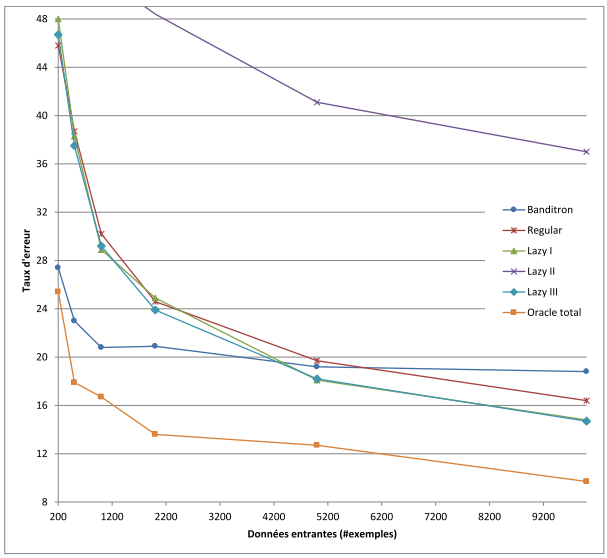


Figure 15 : Résultats des différentes expériences autour du Lazy Perceptron sur le corpus FT1013

Comme prévu, l'*Oracle total* surpasse toutes les autres expériences. Le perceptron *Regular* se comporte comme un Oracle total apprenant à partir des N/Q exemples positifs et constitue donc une *baseline* pour l'expérience *Lazy I*. Ces deux dernières expériences se comporte similairement jusqu'à 2000 exemples environ. Ensuite, *Lazy I* produit de meilleurs résultats tout en utilisant la même quantité d'exemples positifs. Bien que la différence soit faible, cela montre que *Lazy I* profite des exemples virtuels générés pour améliorer ses performances par rapport à un perceptron classique.

Lazy II, qui n'utilise que les exemples négatifs et dont les courbes n'apparaissent pas toujours dans les figures par soucis de clarté passe de 47,7% à 32,4% de taux d'erreur sur le corpus FT3000 soit un gain de 15,3 points, et de 71,8% à 37% sur FT1013 soit 34,8 points de mieux. C'est une performance très honorable vis-à-vis des autres expériences, et cela montre que le Lazy Perceptron peut apprendre des $\frac{Q-1}{Q}N$ exemples négatifs sans avoir

aucune information sur leur classe réelle (!), principalement lorsque le nombre d'exemples disponibles devient important.

L'expérience *Lazy III*, qui utilise à la fois les exemples positifs et négatifs comme combinaison linéaire des exemples virtuels générés par *Lazy I* et *II* obtient les meilleurs résultats des expériences utilisant le Lazy Perceptron.

Enfin, le *Banditron* se montre très efficace pour de petites tailles de corpus mais produit de moins bons résultats que les expériences *Lazy* pour des tailles plus importantes.

Tous ces résultats ont été obtenus à partir des transcriptions automatiques des corpus avec un taux d'erreur mot supérieur à 30%, à la fois lors de l'entraînement des classifieurs que lors du test. Ceci garantit l'efficacité de ces algorithmes sans besoin d'effort de transcription manuelle et démontre qu'ils sont robustes par rapport au bruit généré par les systèmes de RAP.

8.6 **Conclusion**

Nous avons présenté le *Lazy Perceptron*, un algorithme novateur proposant une méthode permettant d'apprendre dans un environnement où les exemples sont partiellement annotés, comme dans le cadre de l'Oracle Partiel.

La stratégie employée est de générer des exemples virtuels en moyennant un sous-ensemble d'exemples existants afin de permettre au classifieur à la fois d'apprendre par renforcement (ce que le perceptron classique ne peut pas faire) mais surtout d'apprendre à partir d'exemples négatifs qui ne portent aucune information quant à la classe à laquelle ils appartiennent.

Cependant, un inconvénient majeur réside dans la nécessité d'avoir une distribution équiprobable des exemples sur les classes prédites. Comme les problèmes de compréhension de la parole ont le plus souvent des distributions non uniformes avec parfois une queue de distribution assez longue, une stratégie pourrait être – comme pour le Banditron – de tirer aléatoirement la classe prédite. Mais ce caractère aléatoire génère des propositions erronées et dégrade donc artificiellement le système, ce qui est difficilement envisageable sur un service en production.

Cette contrainte peut néanmoins être relaxée en modifiant l'algorithme pour tirer partie d'une matrice de confusion apprise à priori, sur le *bootstrap* par exemple, ce qui constitue une perspective de recherche intéressante.

Conclusion et perspectives

L'objectif de cette thèse, dans le cadre de l'apprentissage automatique appliqué à la compréhension de la parole, a été l'exploration et la proposition de techniques peu voire non supervisées visant à réduire l'effort d'annotation de corpus nécessaires à la constitution ou à l'adaptation de systèmes de dialogue oral.

J'ai tout d'abord présenté trois techniques semi-supervisées populaires visant à réduire l'effort d'annotation et les ai appliquées au domaine du traitement automatique de la langue naturelle, notamment pour la compréhension automatique dans des systèmes de dialogue oral déployés.

Tout d'abord le self-training et le co-training qui donnent de bons résultats lorsque les données d'apprentissage sont disponibles en quantité limitée, mais peinent à améliorer des modèles déjà performants. Il sont donc intéressant pour constituer un modèle à moindre coût mais ne sont d'aucune aide pour adapter un système en exploitation (qui est donc déjà suffisamment performant pour être exploité).

Puis j'ai étudié l'active-learning et l'ai évalué en utilisant un critère d'active-learning original basé sur un consensus de classifieurs qui permet une réduction conséquente du coût d'annotation.

Certaines de ces techniques ont montré leurs limites qui soulèvent des questions et ouvrent des perspectives de recherche :

- Les limites du co-training étant principalement l'absence totale de supervision, il serait intéressant de combiner le co-training à une forme d'active-learning comme cela a été fait dans (Mao, et al. 2009) ou (Pierce et Cardie 2001).
- Le problème de performance du choix du nombre d'itérations de l'active-learning peut être contourné en utilisant des modèles d'apprentissage en ligne (on-line learning, (Kivinen et Warmuth 1997)) qui permettent d'intégrer de nouveaux exemples à des modèles déjà constitués sans devoir reconstruire l'intégralité du modèle à partir de l'ensemble des exemples d'apprentissage. C'est le cas du perceptron ou des algorithmes de type MIRA (Kramer 2003) par exemple.
- L'active-learning pourrait intégrer l'information temporelle portée par les exemples dans son critère de sélection afin de privilégier les exemples récents qui sont réputés plus représentatifs des données à classer. Il faudrait également préalablement confirmer cette hypothèse en mesurant l'effective proximité structurelle entre les exemples temporellement proches, et pour ce faire trouver une mesure de distance entre deux exemples.

J'ai ensuite proposé une contribution originale appliquant l'active-learning à la rédaction de règles manuelles : l'objectif est de guider les experts rédigeant des règles manuelles en leur soumettant des exemples porteurs d'information. La conclusion de cette étude a été très positive, montrant un gain potentiel important quant à la rapidité avec laquelle un expert utilisant un tel système écrirait des règles en nombre et en qualité.

Cependant l'étude a été simulée et nécessiterait une évaluation en conditions réelles, avec de vrais experts pour confirmer les résultats expérimentaux.

L'idée qu'un système automatique, aussi perfectionné soit-il, ne pouvait pas créer de connaissance supplémentaire, confirmée par l'échec du co-training lorsque le modèle initial était déjà d'une bonne qualité, nous a amené à chercher à obtenir cette connaissance auprès d'humains mais à moindre coût. L'*Active-learning* cherchait à réduire l'effort fourni par les experts (annotateurs et transcripteurs) mais une autre catégorie d'humains entrait en jeu dans le cycle de production d'un service de compréhension de la parole : l'utilisateur. Même s'ils sont moins expert que les concepteurs du système, ce sont eux qui, au final, utilisent et évaluent le système. C'est ainsi qu'est née l'idée de voir un utilisateur comme un *Oracle partiel* capable de ne donner qu'une confirmation ou une infirmation de la décision du système, et de s'en servir pour entraîner et adapter le système aux changements de comportement de ces mêmes utilisateurs.

J'ai montré qu'en cas de confirmation, les stratégies d'apprentissage par renforcement fonctionnaient bien, cependant, lorsque l'utilisateur infirmait une hypothèse du système, l'intégration de cette information était plus difficile et ne conduisait pas à une amélioration significative.

Ainsi, en partenariat avec le Laboratoire d'Informatique Fondamentale de Marseille nous avons introduit le *Lazy Perceptron*, un algorithme d'apprentissage capable d'apprendre efficacement dans un tel environnement, où les exemples sont partiellement annotés par un Oracle Partiel. Cet algorithme a démontré son efficacité sur deux corpus issus de systèmes de compréhension de la parole déployés par France Télécom, et notamment sa capacité à apprendre des erreurs du système.

Là encore les données étaient simulées, et même si nous étions plus proche d'une situation réelle qu'avec des données générées aléatoirement, une application dans des conditions réelles devra être envisagé pour confirmer ces résultats expérimentaux. L'algorithme actuel nécessite une distribution uniforme sur les classes, cette contrainte devra être relaxée afin de pouvoir l'appliquer à bon nombre d'applications réelles.

Bibliographie

Béchet, Frédéric, Christian Raymond, Frédéric Duvert, et Renato De Mori. «Frame based interpretation of conversational speech.» *Spoken Language Technology Workshop (SLT).* Berkeley, CA, 2010.

Blum, A., et T. Mitchell. «Combining labeled and unlabeled data with co-training.» *Proceedings of the eleventh annual conference on Computational learning theory.* 1998. 92-100.

Bohus, D., et A Rudnicky. «Constructing Accurate Beliefs in Spoken Dialog Systems.» *Proceedings of ASRU.* San Juan, Puerto Rico, 2005.

Bohus, Dan, et Alexander Rudnicky. «Implicitly-Supervised Learning in Spoken Language Interfaces: An Application to the Confidence Annotation Problem.» *Proceedings of 8th SIGdial Workshop.* Antwerp, Belgium, 2007. 256-264.

Bonneau-Maynard, H., et al. «Results of the French Evalda-Media evaluation campaign for literal understanding.» *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC 2006).* Genova, Italy, 2006.

Carpenter, B., et J. Chu-Carroll. «Natural language call routing: a robust, self-organizing approach.» *5th International Conference on Spoken Language Processing (ICSLP'98).* Sydney, Australia, 1998. paper 76.

Chomsky, Noham. *Syntactic Structures.* Mouton & Co, 1957.

Collins, et Singer. «Unsupervised Models for Named Entity Classification.» *Proceedings of EMNLP'99.* 1999.

Collins, M. «Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms.» *Proceedings of the ACL-02 conference on Empirical methods in natural language processing (EMNLP'02).* Morristown, New Jersey: Association for Computational Linguistics, 2002. 1-8.

Collins, M., et Y. Singer. *Proceedings of EMNLP'99.* 1999. 100-110.

Cortes, C., et V. Vapnik. «Support-Vector Networks.» *Machine Learning vol. 20*, 1995: 273-297.

Crammer, Koby, et Yoram Singer. «Ultraconservative online algorithms for multiclass problems.» *The Journal of Machine Learning Research vol.3*, 2003: 951-991.

Dagan, Ido, et Sean P. Engelson. «Committee-Based Sampling for Training Probabilistic Classifiers.» *Proceedings of the International Conference on Machine Learning (ICML'95).* 1995. 150-157.

Damnati, G., F. Béchet, et R. De Mori. «Spoken language understanding strategies on the France Telecom 3000 Voice Agency corpus.» *International Conference on Acoustics, Speech and Signal Processing (ICASSP'07).* Honolulu, 2007.

Denis, F., R. Gilleron, et M. Tommasi. «Classification de textes et co-training à partir de textes positifs et non étiquetés.» *Actes de la Conférence Francophone sur l'Apprentissage.* Orléans, France, 2002. 205-220.

Duda, R. O., et P. E. Hart. *Pattern Classification and Scene Analysis.* New York: John Willey & Sons, 1973.

Evermann, G., et al. «Training LVCSR Systems on Thousands of Hours of Data.» *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'05).* Philadelphia, PA, USA, 2005. 209-212.

Freund, Yoav, et Robert E. Schapire. «Experiments with a new boosting algorithm.» *Machine Learning: Proceedings of the Thirteenth International Conference*, 1996: 148-156.

Goldman, S., et Y. Zhou. «Enhancing supervised learning with unlabeled data.» *Proceedings of the 17th International Conference on Machine Learning.* 2000. 327-334.

Gorin, A.L., G. Riccardi, and J.H. Wright. "How may I help you?" *Speech Communication.* 1997. 113-127.

Gotab, Pierre, Frederic Bechet, et Geraldine Damnati. «Active Learning for rule-based and corpus-based Spoken Language Understanding models.» *Acoustic Speech Recognition and Understanding (ASRU).* Merano, Italie, 2009. 444-449.

Gupta, N., G. Tur, D. Hakkani-Tur, S. Bangalore, G. Riccardi, et M. Gilbert. «The AT&T spoken language understanding system.» *IEEE Transactions on Audio, Speech, and Language Processing*, 2006: 213-222.

Guz, U., S., Hakkani-Tür, D. Cuendet, et G. Tur. «Co-training Using Prosodic and Lexical Information for Sentence Segmentation.» *Proceedings of the 8th Conference of the International Speech Communication Association (Interspeech'07).* Antwerp, Belgium, 2007. 2597-2600.

Haffner, P., G. Tur, et J. Wright. «Optimizing SVMs for complex call classification.» *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'03).* Hong-Kong, 2003.

Hsu, C.-W., C.-C. Chang, et C.-J. Lin. «A Practical Guide to Support Vector Classification.» 2003, rev. 2010.

Hwa, Rebecca, Miles Osborne, Anoop Sarkar, et Mark Steedman. «Corrected co-training for statistical parsers.» *ICML-03 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining.* Washington, DC, 2003. 95-102.

Kakade, S. M., S. S. Shwartz, et A. Tewari. «Efficient bandit algorithms for online multiclass prediction.» *Proceedings of the 25th International Conference on Machine Learning (ICML'08).* 2008. 440-447.

Kamm, T., et G. Meyer. «Selective sampling of training data for speech recognition.» *International Conference on Human Language Technology Research, HLT'02.* San Diego CA, USA, 2002. 20–24.

Kivinen, J., et M. K. Warmuth. «Additive versus exponentiated gradient updates for linear prediction.» *Information and Computation*, 1997: 1-64.

Krahmer, E., M. Swerts, M. Theune, et M. Weegels. «Error Detection in Spoken Human-Machine Interaction.» *International Journal of Speech Technology*, 2001: 19-30.

Lin, C.-J., et C.-C. Chang. «LIBSVM : a library for support vector machines.» *ACM Transactions on Intelligent Systems and Technology*, 2011: 2:27:1--27:27.

Lin, C.-J., R.-E. Fan, K.-W. Chang, C.-J. Hsieh, et X.-R. Wang. «LIBLINEAR: A library for large linear classification.» *Journal of Machine Learning Research*, 2008: 1871-1874.

Mao, Ching-Hao, Hahn-Ming Lee, Devi Parikh, Tsuhan Chen, et Si-Yu Huang. «Semi-supervised co-training and active learning based approach for multi-view intrusion detection.» *Proceedings of the 2009 ACM symposium on Applied Computing.* Honolulu, Hawaii, 2009. 2042-2048.

Müller, Christoph, Stefan Rapp, et Michael Strube. «Applying Co-Training to reference resolution.» *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics.* Philadelphia, Pennsylvania, 2002. 352-359.

Nigam, Kamal, et Rayid Ghani. «Analyzing the Effectiveness and Applicability of Co-training.» *Proceedings of the ninth international conference on Information and knowledge management.* McLean, Virginia, 2000. 86-93.

Pierce, David, et Claire Cardie. «Limitations of Co-Training for Natural Language Learning from Large Datasets.» *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing.* Pittsburgh, PA, 2001. 1-9.

Ralaivola, L., B. Favre, P. Gotab, F. Bechet, et G. Damnati. «Applying Multiclass Bandit algorithms to call-type classification.» *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU'11).* 2011.

Riccardi, G., et D. Hakkani-Tur. «Active learning: theory and applications to automatic speech recognition.» *IEEE Transactions on Speech and Audio Processing, vol. 13, no. 4*, 2005: 504–511.

Rosenblatt, F. «The Perceptron: A probabilistic model for information storage and organization in the brain.» *Psychological Review*, 1958: 386-408.

Rouvier, Mickael, Georges Linarès, et Benjamin Lecouteux. «Query-Driven Strategy for On-the-Fly Term Spotting in Spontaneous Speech.» *EURASIP Journal on Audio, Speech and Music Processing.* 2010.

Sakkar, Anoop. «Applying co-training methods to statistical parsing.» *roceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies.* Pittsburgh, Pennsylvania, 2001. 1-8.

Schapire, Robert E. «The strength of weak learnability.» *Machine Learning*, 1990: 197-227.

Schapire, Robert E., and Yoram Singer. "BoosTexter: A boosting-based system for text categorization." *Machine Learning, 39*, 2000: 135-168.

Seneff, S. «TINA : A natural language system for spoken language applications.» *Computational Linguistics*, 1992: 61–86.

Shannon, C. "A mathematical theory of communcation." *Bell Systems Technical Journal*, 1948.

Singh, S., M. Kearns, D. Litman, et M. Walker. «Reinforcement learning for spoken dialogue systems.» *Proceedings of NIPS'99.* 1999.

Suendermann, D., J. Liscombe, K. Evanini, K. Dayanidhi, et R. Pieraccini. «C5.» *Spoken Language Technology Workshop, SLT'08.* 2008. 125–128.

Tur, G., R. Schapire, et D. Hakkani-Tur. «Active learning for spoken language understanding.» *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'03.* Hong Kong, China, 2003. 276–279.

Woods, W.A. *What's in a Link: Foundations for Semantic Networks.* NTIS, 1975.

Wu, W., R. Lu, J. Duan, H. Liu, F. Gao, et Y. Chen. «Spoken language understanding using weakly supervised learning.» *Computer Speech & Language*, 2009: 358-382.

Publications personnelles

Gotab, P. « Apprentissage automatique et Co-training. » *TALN/RECITAL*. Senlis, France, 2009.

Gotab, P., Bechet F., et Damnati G. « Active Learning for rule-based and corpus-based Spoken Language Understanding models. » *Acoustic Speech Recognition and Understanding (ASRU).* Merano, Italia, 2009.

Gotab, P., Bechet F., et Damnati G. « Stratégie d'apprentissage actif pour l'adaptation de modèles de compréhension dans un Système de Dialogue Oral déployé. » *Journées d'Etude sur la Parole (JEP).* Mons, Belgique, 2010.

Gotab, P., Damnati G., Bechet F., et Delphin-Poulat, L. « Online SLU model adaptation with a partial Oracle. » *INTERSPEECH*. Makuhari, Japan, 2010.

Ralaivola, L., Favre B., Gotab P., Bechet F., et Damnati G. « Applying Multiclass Bandit algorithms to call-type classification » *Acoustic Speech Recognition and Understanding (ASRU). Hawaii, 2011*.

Index des figures

Index des tables