Université de Montréal

Génération de texte dans le cadre d'un système de réponse automatique à des courriels

par

Stephen Beauregard

Département d'informatique et de recherche opérationnelle
Faculté des études supérieures

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître ès sciences (M. Sc.)
en informatique

Mars 2001

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé:

Génération de texte dans le cadre d'un système de réponse automatique à des courriels

présenté par:

Stephen Beauregard

a été évalué par un jury composé
des personnes suivantes:

président–rapporteur:	Jian-Yun Nie
directeur:	Guy Lapalme
membre du jury:	Max Mignotte

Mémoire accepté le:.....

Sommaire

À mesure que le courrier électronique deviendra chose courante, un système de réponse automatique au courrier électronique sera tout aussi nécessaire pour les entreprises qu'un système automatique de réponse téléphonique. Un projet du RALI financé par les Laboratoires universitaires Bell (LUB) vise le développement d'un tel système pour le service d'aide à la clientèle où la précision et la qualité des réponses générées sont des facteurs très importants. Ce mémoire s'intéresse à la génération de texte en langue naturelle dans le cadre de ce système.

Nous avons adopté une approche d'ingénierie des connaissances, c'est–à–dire fortement dépendante des spécificités du domaine d'application. Après une analyse manuelle de notre corpus de départ, un domaine de discours restreint a été choisi: des questions sur les imprimantes du département d'informatique de l'Université de Montréal. Une analyse de ce corpus a permis de classifier les courriers électroniques pour identifier des traits communs intéressants pour la construction d'un sélecteur de réponse et pour la génération du texte des messages.

Une recherche bibliographique sur les différents niveaux, ou styles, de génération nous a permis de relever une grande gamme de solutions en termes de capacité de génération et complexité d'implantation. Nous avons étudié le style et le contenu du corpus afin de concevoir des schémas de réponse et nous avons constaté que les patrons de phrases avec trous ainsi qu'une structure rhétorique fixe étaient adéquats pour notre application. Nous avons construit des patrons qui produisent une sortie grammaticalement correcte, peu importe la valeur des paramètres d'instanciation des patrons. Nous n'avons repéré aucune circonstance où les techniques de génération profonde ou linguistique auraient été utiles. La taille restreinte de notre corpus (191 messages) et le texte stéréotypé des réponses originales ont aussi grandement limité les choix de techniques.

Nous avons implanté un sélecteur de réponse à base de cas et un moteur de génération de surface en Prolog. Les résultats de nos expériences démontrent que les deux tiers des messages du corpus de test sont traités correctement. Plus de la moitié des messages résultent en des réponses à toutes fins pratiques identiques aux réponses originales. La majorité des erreurs

résultent en des messages génériques d'accusé de réception, d'autres sont dues à la couverture limitée de la base de cas et certaines sont des erreurs de sélection de réponse.

La génération de surface semble un peu simpliste, mais notre solution serait pratique et utile dans l'industrie. Notre approche peut satisfaire à des besoins commerciaux de façon simple, rapide et compréhensible. Des techniques plus sophistiquées de traitement de la langue naturelle seraient des avenues de recherche intéressantes, surtout lorsque le volume de messages augmente.

Table des matières

Table des figures

Liste des tableaux

Remerciements

Je tiens à remercier mon directeur de recherche, Monsieur Guy Lapalme, pour ses conseils et pour sa patience. Je tiens aussi à remercier ma co-équipière dans le projet de réponse automatique, Leila Kosseim, pour son soutien.

Finalement, je tiens à remercier les Laboratoires universitaires Bell (LUB) pour avoir financé ma maîtrise via une bourse, sans quoi mes études auraient été plus coûteuses et sans doute plus difficiles.

Chapitre 1

Introduction

1.1 Le cadre du projet

Le nombre de documents électroniques disponibles aujourd'hui a atteint un niveau si élevé que la manipulation automatique des textes libres est devenue une nécessité. En effet, le traitement manuel de textes est long et dispendieux, rendant ainsi des techniques d'informatique-linguistique assez attrayantes. Les messages électroniques forment une grande portion de ces documents en textes libres. Pourtant, dans les contextes du commerce électronique, la majorité des entreprises ne sont pas en mesure de faire face adéquatement à cette réalité. En fait, certaines études démontrent qu'un grand nombre d'entreprises sont ensevelies par des volumes faramineux de courriels. À mesure que le courrier électronique deviendra chose courante, un système de réponse automatique au courrier électronique sera tout aussi nécessaire pour les entreprises qu'un système automatique de réponse téléphonique.

Un projet du RALI, financé par le Laboratoires universitaires Bell (LUB), vise le développement d'un système automatique de réponse à des courriers électroniques dans le cadre d'un service d'aide à la clientèle[1]. Ce prototype devra accomplir les tâches suivantes:

- analyser des courriels pour en déterminer le contenu et en extraire les éléments importants
- déterminer la réponse appropriée
- rédiger une réponse personnalisée
- avoir une capacité de traitement tant en anglais qu'en français

1. Dorénavant nous utiliserons l'expression "projet de réponse automatique" pour désigner ce project cadre.

Le sujet principal de ce mémoire est le deuxième élément, c'est-à-dire la rédaction des réponses personnalisées via des techniques de génération de texte. Ce mémoire touche aussi aux questions de la validation des données extraites des messages ainsi qu'aux techniques de sélection de patrons de réponse.

Avant de donner une description détaillée du projet de réponse automatique et d'expliquer sa relation avec ce mémoire (ce que nous ferons à la section 1.3), nous présentons d'abord le contexte commercial et technologique pour l'ensemble du projet de réponse automatique.

1.2 Systèmes de réponse automatique - État présent

Le courrier électronique est le mode de communication grandissant le plus rapidement et qui deviendra le plus important du prochain siècle[2]. Pour les entreprises, la popularité du courrier électronique aura un impact majeur sur le service à la clientèle. Pour aider les entreprises à gérer efficacement le service à la clientèle par courrier électronique, de nombreux systèmes de réponse automatique ont été développées. Actuellement il existe trois types de systèmes de réponses automatique: l'auto-réponse, qui n'analyse pas le contenu du message, les systèmes de courrier électronique et les systèmes de réponse à des messages libres.

1.2.1 L'auto-réponse

L'auto-réponse, aussi connue sous les noms *autoresponder*, *AR* , *infobots*, *mailbots* ou *email-on-demand*, est le type le plus simple de réponse automatique. Il s'agit en fait de système du genre *Majordomo* qui envoie un document pré-rédigé en réponse à un courrier électronique. Le choix du document à envoyer est basé sur la présence de mots-clés placés dans le sujet ou dans le corps du message.

Il existe quelques variantes au modèle de base. Avec certains systèmes, l'utilisateur n'a pas besoin de spécifier de mot-clé. Dans ce cas, une adresse e-mail est associée à chaque type de question (par exemple, un produit particulier). Tout message envoyé à cette adresse, peu importe son contenu, déclenche l'envoi du document associé (par exemple, une brochure sur le produit). Dans d'autres systèmes, l'utilisateur peut remplir un formulaire sur une page Web pour indiquer ses coordonnées (nom, adresse e-mail, etc.). Grâce à cette information, un message *personnalisé* lui sera envoyé.

Ces types de systèmes restent assez simples et existent depuis quelques années pour la gestion des listes d'envoi (*mailing lists*). Ils ne prennent pas en considération le contenu du

2. Le contenu de cette section est tiré en grande partie d'un rapport interne donnant l'état présent des systèmes de réponse automatique [Kosseim, 1999].

message. En effet, si un message contient une question bien précise, celle-ci ne recevra pas de réponse directe. Si le message contient un mot-clé, un document associé à ce mot sera envoyé; sinon le message restera non-traité.

Bien que les systèmes d'auto-réponse ne tentent pas de *comprendre* le contenu des messages, ils peuvent être très utiles pour envoyer des accusés de réception (non-personalisés) en attendant qu'un préposé soit disponible pour traiter les messages manuellement.

1.2.2 La gestion de courrier électronique

Les systèmes de gestion de courrier électronique offrent un environnement intégré pour répondre plus efficacement et de façon uniforme aux messages des clients. Les systèmes de ce type offrent des fonctions différentes, mais parmi les caractéristiques de base, on retrouve généralement:

- la réception de messages
- la redirection automatique par mots clés vers un préposé ou une file d'attente
- l'utilisation de patrons de réponses figés pour les questions les plus communes
- l'accès à des bases de données (clients, FAQ ou autres)
- un correcteur d'orthographe
- l'intégration du service à la clientèle (courriel, courrier, télécopie et téléphone)
- l'historique et le stockage des messages

La majorité des systèmes offrent une option de *réponse automatique* suivant trois niveaux d'automatisation:

Personnalisation de patrons de réponse Certains systèmes sont capables de personnaliser un patron de réponse automatiquement. Un préposé lit le message et rédige une réponse (ou choisit un patron de réponse), et le système inclut automatiquement un entête personnalisé.

Réponse automatique par mots-clés dans le sujet D'autres systèmes proposent de répondre eux-mêmes aux messages sans intervention humaine. L'analyse du message, le choix du patron de réponse et l'envoi sont faits automatiquement. Dans ce cas, le message est analysé en vérifiant la présence de mots-clés dans l'entête du message.

Réponse automatique de formulaires Web Les systèmes de gestion de courrier les plus sophistiqués, eux, analysent le corps du message. Ces systèmes supposent que le message a été envoyé en remplissant un formulaire Web plutôt qu'en rédigeant un texte libre en langue naturelle. Ce type de courrier peut être répondu facilement de façon

automatique car les données sont bien structurées et toutes les possibilités de questions ou de situations peuvent être planifiées par le concepteur des formulaires.

1.2.3 Les systèmes de réponse au texte libre

Les systèmes d'auto-réponse au texte libre, eux, tentent de répondre à un message en fonction de son contenu, c'est-à-dire, en fonction du texte libre dans le corps du message. Ces systèmes, plus sophistiqués, sont nettement moins répandus que les systèmes précédents et sont souvent des application faites sur mesure. Parmi ces systèmes, certains ont été développés dans un contexte de recherche mais la majorité sont des applications commerciales.

1.2.4 La préparation des réponses

Comme nous venons de voir, dans les systèmes commerciaux, il existe plusieurs niveaux d'automatisation possibles pour la préparation et la génération de réponses:

Document figé La copie intégrale d'un document pré-établi dans le courriel de réponse est une solution simple utilisée par les systèmes d'auto-réponse.

Textes figés dans un schéma Un schéma de message simple peut être instancié de façon variable avec plusieurs phrases ou paragraphes figés.

Personnalisation simple Des patrons d'entête et de salutation de réponse peuvent être instanciés avec le nom du client, par exemple.

Patrons à trous Des patrons génériques peuvent être instanciés avec des données provenant de l'analyse du contenu du message ou d'une autre source. Par exemple, une phrase générique confirmant des montants ou des dates pourrait être construite de cette façon.

Certains systèmes offrent un environnement convivial pour la spécification et modification de schémas de réponses à base de texte figé et de patrons à trous. D'autres offrent la possibilité à un préposé de modifier, avant son envoi au client et via un éditeur spécialisé, une réponse automatique déjà instanciée. Ceci est intéressant dans le cas où un classifieur automatique de messages peut attribuer un niveau de confiance (faible dans le cas présent) à ses décisions et donc aux messages instanciés.

En pratique, les patrons à trous ont certaines lacunes. Afin de produire des messages grammaticalement corrects, peu importe l'instanciation exacte de la partie variable du pa-

tron, il faut parfois avoir recours à des tournures peu naturelles[3]. Ceci est même vrai pour l'anglais où il y a très peu d'accords grammaticaux à respecter. Des systèmes de recherche en génération de texte ont tenté de surmonter cette lacune en utilisant la *génération linguistique*. Brièvement, ceci consiste en l'instanciation des patrons lexico-syntaxiques pré-définis avec des éléments variables pour obtenir une description syntactique et lexicalisée du texte à générer. Cette description est ensuite transformée en texte de sortie final via une grammaire génératrice.[4] Avec cette technique on peut, par exemple, spécifier des patrons qui nécessitent des accords de genre et de nombre. Ceci facilite considérablement la génération de phrases naturelles, un élément important de la qualité ultime des réponses.

1.3 Projet de réponse automatique et relation avec ce mémoire

Comme le projet de réponse automatique était développé dans le cadre du service à la clientèle, la précision et la qualité des réponses générées sont des facteurs très importants. En effet, il est plus important de répondre correctement à un petit nombre de questions que de répondre approximativement à un ensemble plus large de questions. Une réponse inappropriée peut avoir des répercussions assez négatives en ce qui concerne la satisfaction de la clientèle. Pour s'assurer d'une précision plus élevée, l'équipe du RALI a adopté une approche fortement dépendante des connaissances spécifiques au domaine de discours.

Au début de la collaboration avec Bell, nous avons proposé d'utiliser un corpus de messages directement relié à un service disponible à ces abonnés, l'Afficheur Internet. Ce service de répondeur intelligent notifie l'abonné qu'il a un appel téléphonique quand sa ligne est occupé par une connexion à l'Internet. Pour des raisons internes, Bell n'a jamais pu nous fournir un corpus de messages pour cette application. Nous avons donc entamé les recherches à partir d'un corpus de qualité équivalente et traitant d'un domaine similaire. C'est pourquoi nous avons choisi, après une analyse manuelle, un domaine de discours restreint: des questions sur les imprimantes du département d'informatique de l'Université de Montréal. Dans la section suivante, nous décrivons ce travail préliminaire d'analyse de ce corpus (nécessaire à toute appproche basée sur l'ingénierie des connaissances).

3. Par exemple, "La/les file(s) d'attente lpq1 lpq2 lpq5 est/sont pleine(s)."

4. Une variante de cette technique est la génération linguistique d'expressions nominales, insérées par la suite dans des patrons à trous.

1.3.1 Le corpus

Le corpus utilisé pour le projet est composé de questions-réponses en français auprès de l'équipe de soutien technique de notre département (DIRO). Le domaine de discours est assez précis et le corpus comporte un grand nombre de questions associées à un petit nombre de réponses de façon à garantir un large éventail de formulations d'une même question.

L'analyse de corpus a principalement tenté de classifier les courriers électroniques pour identifier des traits communs intéressants pour l'extraction d'information et la rédaction de patrons de réponse. Nous avons utilisé un ensemble de messages couvrant 3 années d'activité de l'équipe de soutien technique. Les messages des deux premières années ont été retenus pour les analyses initiales et pour de fins de configuration du système; les messages de la troisième année ont été réservés pour les fins de test.

Classification initiale

Un survol initial des messages a permis de distinguer quatres types de messages selon leur but communicatif:

1. **Messages des usagers:** Il s'agit ici des messages envoyés par les utilisateurs des ressources informatiques pour poser une question, pour formuler une requête ou pour rapporter une situation anormale. Ces messages sont ceux qui nous intéressent le plus.
2. **Annonces:** Les annonces sont des messages envoyés par un membre de l'équipe de soutien technique pour annoncer un événement aux autres membres de l'équipe, par exemple, l'installation d'un nouveau logiciel. Ces messages sont moins intéressants dans ce projet, car en général ils n'obtiennent pas de réponse de la part des membres de l'équipe.
3. **Messages systèmes:** Les messages systèmes sont envoyés automatiquement par des processus, plutôt que des humains. C'est le cas, par exemple, des retours automatiques de courriels non-livrés ou des processus de sauvegarde qui ont échoués. Ces messages sont nettement moins intéressants pour ce projet, d'abord parce qu'ils n'obtiennent pas de réponse, ensuite, parce que le message original n'a pas été rédigé spontanément par un humain. Le message ne contient souvent pas de corps (seul le sujet est informatif) ou bien le corps est constitué d'un message d'erreur du système d'exploitation ou du processus.

4. **Rejets:** Les messages qui, pour diverses raisons, ne sont pas analysables. Parmi ces messages, on retrouve des messages en anglais et des messages vides[5].

Pour classifier les messages dans ces quatre classes, nous avons tout d'abord classifié manuellement les 200 premiers textes pour avoir une idée des traits communs ou distinctifs, puis nous avons classé les 200 derniers pour pouvoir évaluer la classification automatique. Celle-ci était basée sur la cooccurence de termes dans des fenêtres de texte d'une longueur donnée. Nous avons essayé deux variantes de cette technique de classification: avec analyse linguistique du contenu textuel (c'est–à–dire, avec l'étiquetage grammatical des mots suivi de leur lemmatisation) et sans analyse linguistique (c'est–à–dire, les jetons bruts du texte). La classification avec analyse linguistique a démontré une très faible performance au niveau du rappel et de la précision[6]. Ce fait peut s'expliquer par deux raisons:

– La quantité de textes dans chaque classe est assez faible (il y a peu de messages dans chaque classe et les messages sont courts), donc lors de l'entraînement, il y a peu de chance de retrouver des affinités lexicales (cooccurrences de mots) représentatives.
– Les domaines de discours sont assez variés et les auteurs sont tous différents. Donc il est difficile de faire ressortir des cooccurrences fréquentes de mots.

Considérant ces résultats, cette première classification a été effectuée en utilisant des règles sans analyse linguistique du contenu du message. Le corpus d'analyse a été découpé tel que présenté à la figure 1.1.

Classe	**Nb de messages**	**% du corpus**
Usagers	2532	67.1%
Annonces	97	2.6%
Systemes	238	6.3%
Rejets	904	24.0%
Total	3771	100%

FIG. 1.1 – *Premier découpage du corpus*

5. Le corpus original fourni par le soutien technique est en fait un seul gros fichier contenant l'ensemble des paires questions-réponses. Ce fichier a été découpé automatiquement en multiples fichiers de questions et de réponses par un script basé sur les entêtes et les citations des messages. Lorsque les citations sont imbriquées ou ne sont pas indiquées par des marques typographiques répertoriées (ex. $' >'$), le script peut ne pas découper le message correctement et peut créer un fichier de question vide.

6. Ces termes viennent du domaine de la recherche d'information.

Sélection du domaine de discours

Des 2532 messages d'usagers, une analyse générale a révélé qu'ils portaient sur un grand éventail de domaines de discours; trop large pour y faire de l'extraction d'information précise. Nous nous sommes concentrés alors sur un domaine de discours précis: les questions d'usagers portant sur les imprimantes.

Pour identifier les messages portant sur des questions d'imprimantes, nous avons tout simplement fait un *grep intelligent*. C'est-à-dire que nous avons répertorié un ensemble de mots clés pertinents au sujet des imprimantes et avons fait un *grep* sur la forme canonique des mots après avoir enlevé les mots grammaticaux (c'est à dire, sémantiquement vides). Sur 2532 messages d'usagers, 205 portent sur les imprimantes.

Ensuite, sur 205 messages portant sur des questions d'impression, nous avons enlevé:

- les *conversations*; c'est-à-dire, les messages d'usagers dont l'interprétation nécessite des connaissances sur un message antérieur. Nous avons donc enlevé les messages où la notion de conversation était marquée typographiquement par une citation ou un "`Re:`" dans le sujet du message.
- les *doublons*; c'est-à-dire, les questions d'usagers qui recoivent plusieurs réponses à différentes occasions.

Une fois ce dernier nettoyage effectué, le corpus a été réduit à 126 messages[7]. Ce corpus est devenu notre corpus d'entraînement. Le corpus test, préparé de façon identique avec les messages de la troisième année d'activité du support technique, comprend 69 messages. Les signatures et les entêtes ont été enlevées de tous les messages.

Caractérisation des questions et des réponses

Les messages du corpus d'entraînement contiennent une moyenne de 47 mots par question. Cette moyenne est plus petite que celle du corpus Reuter-21578[8], utilisé dans des études de classification de textes, mais plus grande que la question moyenne du QA track de la conférence TREC-8 (9 mots) [TREC, 1999].

Quelques exemples de questions typiques et de leurs réponses originales se trouvent aux figures 1.2 et 1.3. Notez que pour des raisons de confidentialité, les noms de personnes ont été modifiés. La majorité des questions du corpus sont de cette forme et les réponses envoyées par les techniciens sont courtes, claires et factuelles. Les réponses sont bien souvent de simples

7. Au fur et à mesure de nos analyses et vérifications subséquentes, nous avons constaté qu'il restait encore quelques doublons, conversations et autres messages non-utilisables dans le corpus. Les chiffres donnés plus loin dans ce texte sont des corrections mineures qui ne changent pas le fond des expériences.

8. http://www.research.att.com/~lewis

confirmations qu'un problème sera réglé bientôt ou qu'il l'est déjà. Au niveau du style, nous avons constaté qu'il y a de petites variations aux niveaux des salutations et des fermetures des réponses.

En ce qui concerne les questions dites "pathologiques" données à la figure 1.4, elles montrent l'utilisation d'un mélange de l'anglais et du français, d'un ton informel, et d'une orthographe plutôt relachée. Ce genre de question ferait échouer un analyseur grammatical conventionnel et est très difficile à traiter automatiquement. La première question est ambiguë au niveau sémantique et son sens ne peut certainement pas être déterminé automatiquement. La deuxième question présente un problème au niveau du mot "impression". Des règles d'extraction d'information très spécifiques au domaine de discours pourraient peut–être repérer cette utilisation de ce mot et en enlever l'ambiguïté sémantique. Finalement, certaines questions contiennent beaucoup d'éléments non textuels. Par exemple, un usager pourrait inclure la sortie de la commande "lpq", une liste du contenu d'une file d'attente, pour illustrer un problème. Des encadrements en caractères ASCII pour les signatures sont un autre exemple. Tandis que les sorties de commandes peuvent contenir des informations importantes pour l'interprétation du message, les signatures sont une source de "bruit textuel" et sont non-significatives sur le plan sémantique.

Classification plus fine en fonction de la réponse

Les messages du corpus d'impression ont ensuite été classés manuellement dans les classes de la figure 1.5 en fonction du sujet de la réponse. Cette classification a été développée pour identifier des stratégies d'extraction particulières à chaque type de requête et les réponses typiques associées à une question précise. Nous avons effectué une classification manuelle, car la précision était très importante.

Pour déterminer les informations pertinentes à extraire pour chaque classe de messages, deux évaluateurs humains les ont classifiés à la main. Pour classifier un texte, un point a été distribué sur les classes les plus pertinentes. Nous avons utilisé un système de points, plutôt que de simplement compter le nombre de textes dans chaque classe. Nous avons constaté qu'un texte peut faire partie de plusieurs catégories, soit parce que les classes ne sont pas mutuellement exclusives, soit parce que le message touche plusieurs sujets ou bien parce que le message est ambigu. La figure 1.5 illustre le découpage des classes alors que la table 1.1 montre la fréquence de chaque classe. Les classes de messages les plus fréquentes sont: un problème générique (13%), une imprimante inaccessible (10,48%), la file d'attente est bloquée (7,58%), comment changer le format d'impression (6,85%), problème d'encre (6,25%), ma tâche ne sort pas (6,15%) et aucune tâche ne sort (5,24%). Celles-ci sont indiquées par un

Rapport de problème:

From: Jane Black <black@iro.umontreal.ca>
Subject: toner imprimante

Bonjour,

Est-ce que le toner de l'imprimante hp2248 peut etre remplace par un neuf? L'impression est de mauvaise qualite et j'ai besoin d'imprimer des documents pour une conference vendredi.

Question comment faire:

From: David Smith <smith@iro.umontreal.ca>
Subject:

Bonjour,

J'aimerais savoir comment je peux imprimer seulement sur un côté de la page sur l'imprimante hp2248.
Merci.
David

Demande d'information générale:

From: John Doe <doe@iro.umontreal.ca>
Subject:

J'ai envoye un message sur l'imprimante (a partir de elm sur champlain) apres lpq j'ai vu que l'imprimante s'appelle hp4sialdus. Pouvez vous me dire ou se trouve cette imprimante?
JD

Suggestion:

From: Jane Black <black@iro.umontreal.ca>
Subject: Ajout a IRO.FAQ

Cher support,

L'option -i-1, pour imprimer recto seulement sur les imprimantes HP, tel que décrit dans <<IRO.FAQ>>, est sans effet depuis capchat, en fait, je n'ai obtenu le recto que depuis saguenay. Il faudrait, à mon avis, en faire mention dans le FAQ ou encore faire en sorte que l'option fonctionne de toutes les stations.

FIG. 1.2 – *Exemples de questions typiques du corpus d'impression*

Rapport de problème:

From: support@iro.umontreal.ca
Subject: Re: toner imprimante

Quelqu'un s'en viens changer la cassette d'encre.

Support.
Universite de Montreal, I.R.O.

Question comment faire:

From: support@iro.umontreal.ca
Subject: Re: imprimante hp

De la facon dont l'imprimante hp est configuree, il est presentement impossible de n'imprimer que sur une page. Desole. Si tu desires imprimer que sur une page, il te faudra utiliser une autre imprimante.

Demande d'information générale:

From: support@iro.umontreal.ca
Subject: Re: imprimante hp4sialdus

Elle est dans le local 3252. Un autre nom est aussi hp3252.

...Support
support@IRO.UMontreal.CA
Universite de Montreal, I.R.O.

Suggestion:

From: support@iro.umontreal.ca
Subject: Re: Ajout a IRO.FAQ

Tu as raison, cette commande fonctionne uniquement sur les SunOs 4.1.X
J'ai ajouté une note dans le FAQ.

Support.

FIG. 1.3 – *Exemples de réponses typiques du corpus d'impression*

Description incompréhensible:

From: John Doe <doe@iro.umontreal.ca>
Subject: imprimante

Bonjour!

Comment on fait pour imprimer? (la porte est barre!)

Mot ambigu:

From: David Smith <smith@iro.umontreal.ca>
Subject: impression

J'ai bien l'impression qu'il manque d'encre dans cette imprimante. Ma derniere copie est plutot blanche...

Orthographe non-standard:

From: Jane Black <black@iro.umontreal.ca>
Subject: mail

Salut support.

C'est le mail que t'as demande. Apropos de quoi? Ben voyons, la u211hp qui veut plus imprimer, t'ecoute don' pas c'qu'on dit?

Ton et vocabulaire informels:

From: Jane Black <black@iro.umontreal.ca>
Subject: pbs imprimante

Salut,

Juste pour te dire qu'il y a encore des pbs avec la criss d'imprimante. Pourquoi c'est toujours la notre qu'est fuckée!

Anyway, elle a des pbs à imprimer pls pages. La secrétaire en a envoyé 10 et elle n'en a imprimé que 5, moi j'en ai envoyé 6 et elle en a imprimé 3 (ou qqchose comme ça).

FIG. 1.4 – *Exemples de questions pathologiques du corpus d'impression*

encadrement autour du nom de la classe à la figure 1.5

1.3.2 Emphase du projet cadre

La réponse au courriel peut être divisée en 3 étapes: la reconnaissance du problème (c'est à dire, lire et comprendre le message); la recherche d'une solution (trouver des éléments de la réponse); et la fourniture d'une réponse (rassembler les éléments de la réponse dans un texte et l'envoyer). Nous avons implanté ces fonctionalités par une suite de traitements, illustrés à la figure 1.6:

- Lors de la réception d'un message, un module de classification détermine le type de message. Plus spécifiquement, nous déterminons s'il s'agit ou non d'une question du domaine des imprimantes.
- En fonction de la classe du message, un module d'extraction d'information tente d'extraire l'information importante et de la structurer dans des formulaires sémantiques pré-définis. Les méthodes d'extraction d'information sont bien adaptées au texte possiblement très informel des courriels des clients. Le parsage syntaxique classique ne fonctionne pas toujours adéquatement dans ce genre de situation.
- Les formulaires remplis sont ensuite envoyés à un vérificateur sémantique qui utilise des bases de connaissances statiques et dynamiques sur le domaine de discours pour vérifier la validité de l'information extraite.
- Les formulaires valides sont étoffés en faisant des inférences sur le discours. Ce module infère des connaissances nouvelles à partir d'une base de connaissances et à partir de l'information déjà extraite.
- En fonction du contenu des formulaires d'extraction, un module de sélection choisit le patron de réponse le plus approprié.
- Finalement, le patron de réponse est rempli et coordonné pour produire une réponse naturelle et personalisée par la génération de texte.

L'emphase du projet de réponse automatique est mise sur les deux modules de linguistique-informatique: l'extraction d'information et la génération de texte. Ces deux modules sont identifiés par les encadrés foncés à la figure 1.6.

1.4 But de ce mémoire

Pour ce projet de maîtrise, nous nous concentrons sur une partie des traitements décrits à la section précédente. Nous visons plus spécifiquement les couches de sélection d'un patron de

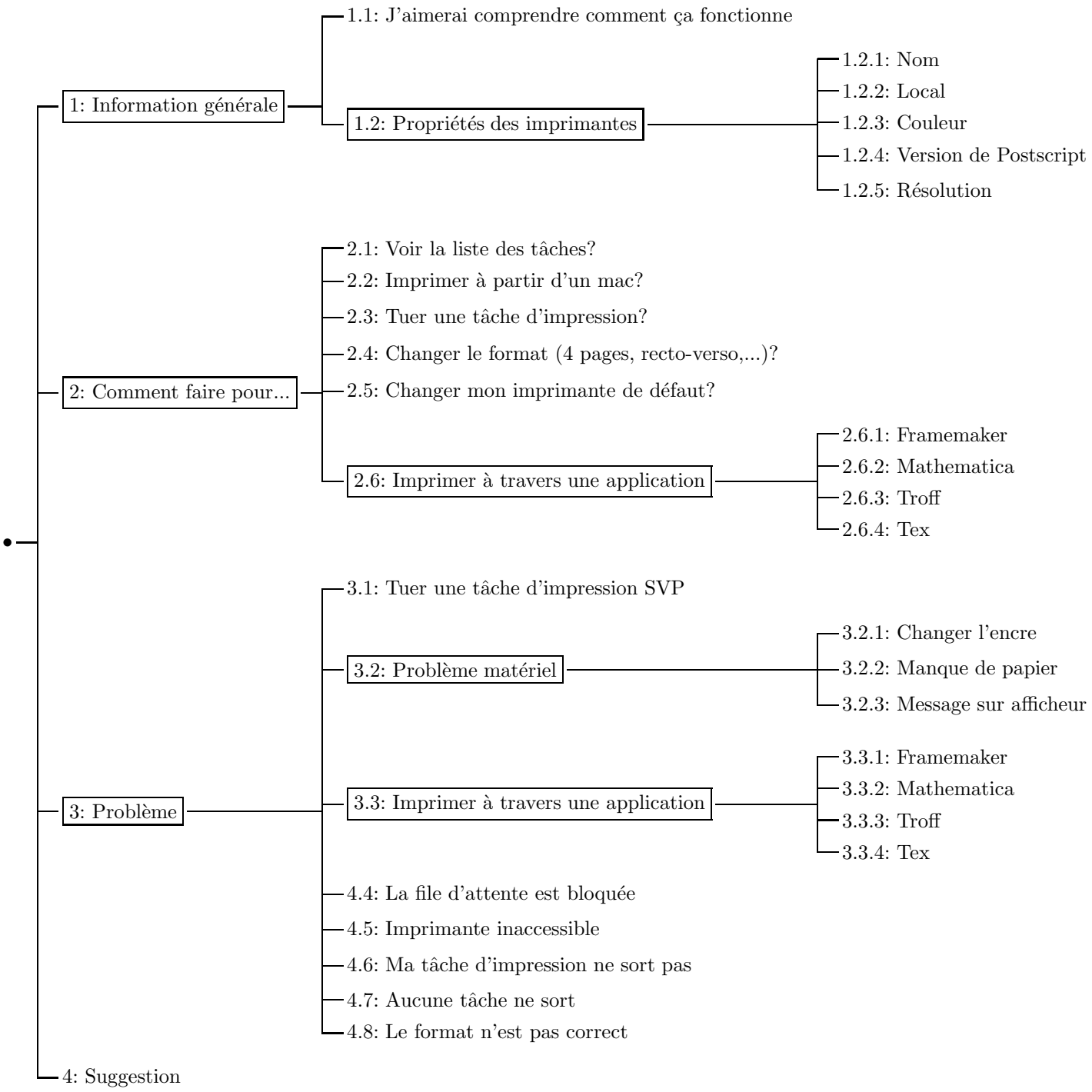

FIG. 1.5 – *Classification finale du corpus d'impression*

Classe	moyenne (évaluateurs 1 et 2) points	%
1 info	2,88	2,32
1.1 comprendre	1,00	0,81
1.2 propriétés	2,75	2,22
1.2.1 nom	2,38	1,92
1.2.2 local	4,00	3,23
1.2.3 couleur	2,63	2,12
1.2.4 postcript	0,00	0,00
1.2.5 resolution	1,00	0,81
2 comment	2,63	2,12
2.1 liste	0,50	0,40
2.2 mac	1,00	0,81
2.3 tuer tâche	2,63	2,12
2.4 changer format	8,50	6,85
2.5 changer défaut	1,00	0,81
2.6 application	2,38	1,92
2.6.1 framemaker	2,38	1,92
2.6.2 mathematica	0,00	0,00
2.6.3 troff	1,00	0,81
2.6.4 tex	1,00	0,81
3 problème	16,13	13,00
3.1 tuer tâche SVP	4,98	4,01
3.2 matériel	1,00	0,81
3.2.1 encre	7,75	6,25
3.2.2 papier	5,00	4,03
3.2.3 afficheur	2,25	1,81
3.3 application	0,50	0,40
3.3.1 framemaker	4,75	3,83
3.3.2 mathematica	2,00	1,61
3.3.3 tex	1,00	0,81
3.4 file bloquée	9,40	7,58
3.5 inaccessible	13,00	10,48
3.6 ma tâche ne sort pas	7,63	6,15
3.7 aucune tâche ne sort	6,50	5,24
3.8 format	1,00	0,81
4 suggestion	1,50	1,21
total	**124**	**100**

TAB. 1.1 – *Fréquence de chaque classe dans le corpus d'impression*

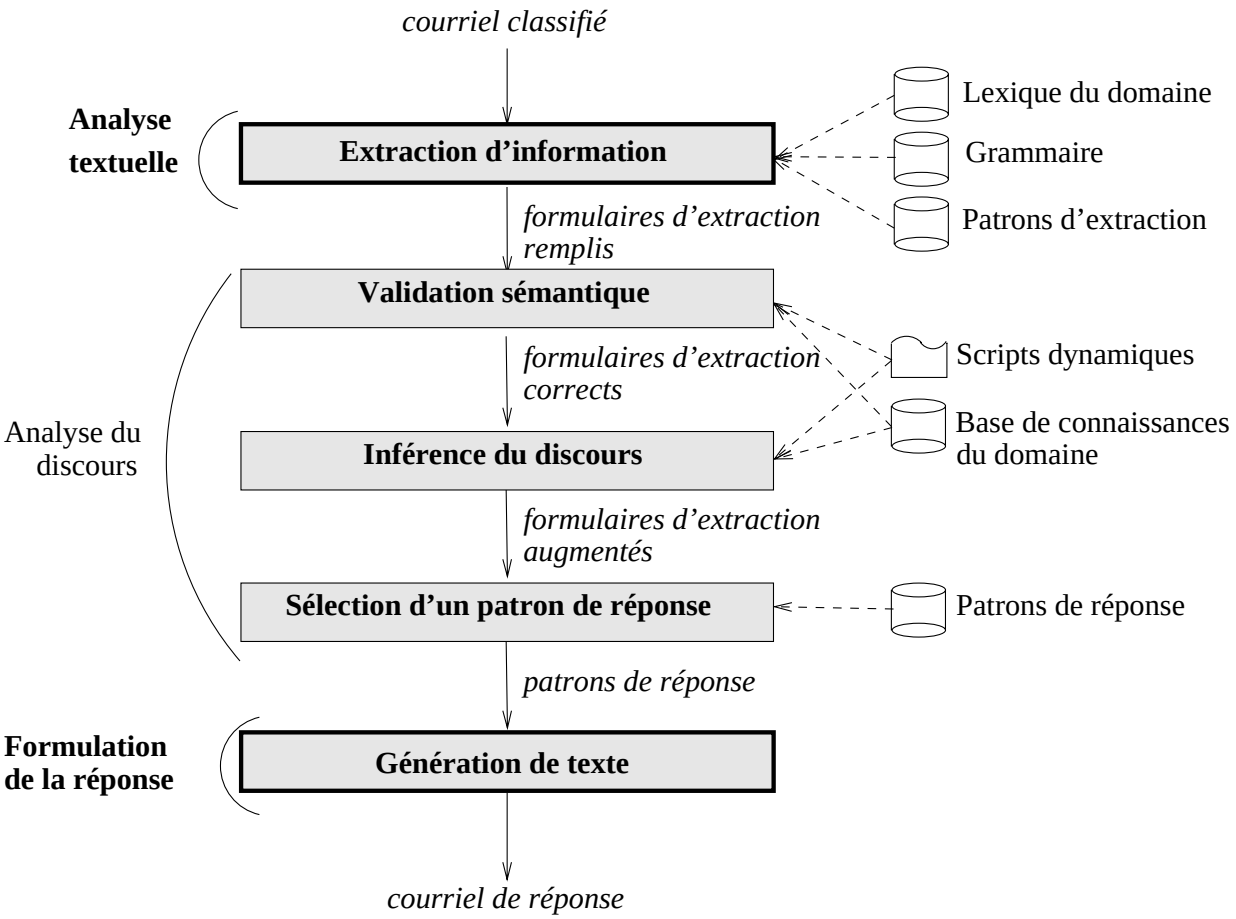

FIG. 1.6 – *Architecture du système*

réponse et de génération de texte. Nous mettrons peu d'emphase sur les étapes de validation et d'inférence des données pour les raisons que nous allons expliquer au chapitre 3.

Les buts de ces travaux de maîtrise sont donc:

- d'implanter et d'évaluer un système fonctionnel
- de démontrer l'utilité de la génération de texte
- de suggérer des pistes pour rendre l'utilisation des patrons conviviale dans une mise en œuvre commerciale
- d'orienter les recherches futures en réponse automatique.

1.5 Structure du mémoire

Ce mémoire développe des techniques pour la sélection de patrons de réponse et pour la génération de texte. Pour présenter nos résultats, nous procédons comme suit. Au chapitre 2, nous nous familiarisons avec le domaine de la génération de texte et aborderons la question du niveau de génération qui est la plus approprié pour notre application. Au chapitre 3, nous examinerons la forme des données de sortie du système d'extraction d'information ainsi que les étapes de validation, d'inférence du discours et de sélection de patron réponse. Le chapitre 4 décrit l'approche qui sera adoptée pour la génération de texte et l'application du modèle abstrait de génération à notre domaine de discours, c'est-à-dire la réponse aux question d'impression. Au chapitre 5, nous décrivons l'implantation du système en Prolog et au chapitre 6, nous évaluons les réponses générées. Finalement, le chapitre 7 résume les idées maîtresses du mémoire et présente des avenues de recherches futures.

Chapitre 2

Aperçu de la génération de texte

Dans ce chapitre, nous allons donner un aperçu du domaine de la génération et de quelques travaux pertinents. Nous verrons plusieurs niveaux de complexité dans l'implantation d'un générateur de texte. Nous soulignerons le consensus autour du niveau approprié qui semble s'imposer pour les applications industrielles typiques d'aujourd'hui. Puisque nous visons une implantation pratique, cette question du niveau approprié nous concerne directement. Mais comme point de départ de ce chapitre, nous allons dégager certaines motivations pour la génération.

2.1 Motivations pour la génération

Pourquoi la génération et non la simple utilisation de texte figé et de quelques patrons? Pourquoi faire la génération de langage? Comme discipline scientifique, on aimerait que la génération de texte fasse avancer la science cognitive et l'intelligence artificielle et nous aide à mieux comprendre la faculté humaine de production de language. Ces justifications plutôt théoriques touchent peu aux principales motivations pour la génération, c'est à dire résoudre des problèmes pratiques. Les chercheurs reconnaissent des avantages concrets de la génération qui ne sont pas présents dans le texte figé [Coch, 1998] [Dale, 1995] [Reiter, 1995]:

Meilleure productivité Avec les mêmes coûts, on peut produire plus de documents automatiquement, ou inversement, on peut produire la même quantité de documents avec des coûts moindres.

Qualité améliorée On peut générer des documents de plus grande qualité, plus fluides, plus naturel avec la génération.

Multilinguisme On peut produire des documents dans plusieurs langues à la fois ou dans une langue différente de celle du rédacteur. La génération automatique peut donc deve-

nir une alternative intéressante à la traduction automatique, et serait moins coûteuse que la traduction assistés ou humaine. Le multilinguisme est déjà un enjeu important pour les multinationales actives dans le commerce électronique.

Conformité aux normes Dans certains cas, il est important pour un organisme utilisateur de respecter certains critères de qualité ou de contenu, par exemple, les normes militaires américaines telles que DOD-2167A [US Department of Defense, 1988], et même de satisfaire à des normes de rédaction pour simplifier les textes ou pour en faciliter la compréhension (par exemple, l'utilisation du "Simplified English" [AECMA, 1986]). Cette vérification de conformité peut faire partie intégrante d'un système de génération.

Expressivité À mesure que les machines et les applications deviennent plus complexes, elles nécessitent des moyens d'expression de plus en plus sophistiqués. Or, seulement la génération pourra répondre à ce besoin, par exemple dans de futurs systèmes de dialogue en ligne.

Faisabilité Lorsqu'il serait (presque) impossible de faire autrement. Par exemple, tout ce qui est généré dynamiquement (comme les pages Web dynamiques), les systèmes de question/réponse dans un domaine ouvert, les systèmes conversationnels en ligne, la génération de rapports diagnostiques en-ligne (par exemple IDAS [Reiter et al., 1995]) seraient impraticables sans la génération de texte. On peut noter que cette motivation peut être vue comme variations des trois précédentes: un rédacteur ne pourrait produire ou modifier des centaines de documents par jour, ou des documents dans des langues qu'il ne connait pas, ou des documents satisfaisant à certaines normes.

Explosion combinatoire des variantes Sans la génération, il faut prévoir toutes les variantes possibles de texte de sortie et en préparer les instanciations d'avance. Par exemple, dans le champ d'application du système PEBA-II [Milosavljevic et al., 1996], la zoologie descriptive, il aurait été impossible de générer d'avance tout les textes comparant toutes les paires d'animaux dans la base de connaissances.

Multimodalité Il peut être plus naturel d'ajouter des éléments graphiques, comme des hyperliens ou des formats indiquant des emphases, avec un générateur de texte qu'avec du texte figé.

Consistance logiciel Un générateur de texte bien intégré aux autres éléments d'un système peut garantir un certain contrôle de qualité et de configuration et une certaine uniformité de présentation. Par exemple, un méchanisme uniforme de génération linguistique de messages ou d'avertissements pourrait éviter des messages incorrects aux usagers lors de l'évolution d'un système.

Adaptabilité Un générateur bien intégré peut faciliter la propagation de changements à

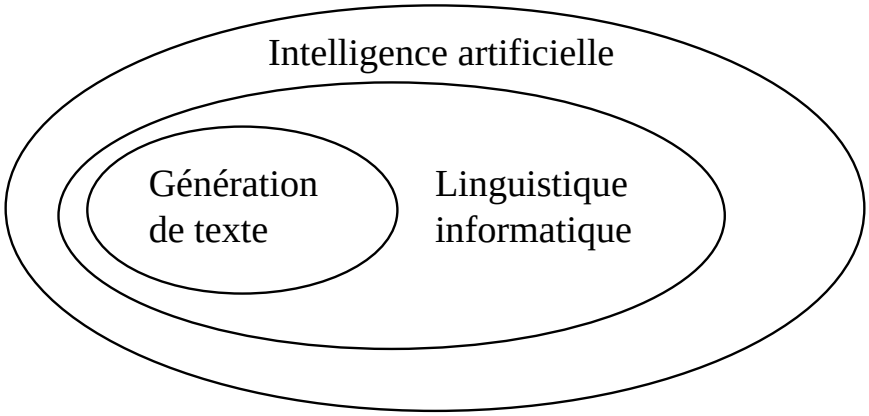

FIG. 2.1 – *Relation entre la génération de texte et l'intelligence artificielle*

un système (par exemple, d'un modèle entité-relation de base de données) vers les éléments linguistiques des interfaces usagers (le texte dans les formulaires d'une base de données).

2.2 Modèle abstrait de la génération

La génération du langage naturel est un sous-domaine de la linguistique informatique et de l'intelligence artificielle (voir la figure 2.1) portant sur l'étude et l'implantation de la production automatique du langage écrit et parlé [Buseman, 1999]. L'étude de la production du langage par les humains est une entreprise pluri-disciplinaire, demandant de l'expertise d'autres branches de la science, telles que la linguistique, la psychologie, l'ingénierie, l'informatique et même la philosophie. Un des buts de ce domaine est d'étudier comment les programmes d'ordinateur peuvent être conçus pour générer du texte de bonne qualité à partir de représentations (abstraites) des informations à communiquer.

La génération de texte est souvent caractérisée comme un processus ayant comme point de départ les buts communicatifs visés et utilisant des techniques de planification pour convertir progressivement ces buts vers du langage parlé ou écrit (figure 2.2). Dans cette optique, les buts communicatifs généraux sont raffinés en sous-buts de nature de plus en plus linguistique, culminant en buts linguistiques de bas niveau, c'est-à-dire des suites de mots. Habituellement, une modularisation divise le processus de génération en une partie stratégique ("décider quoi dire") et une partie tactique ("décider comment le dire"). Dans le cas de génération de texte, il y a aussi une partie moteur ("décider comment l'écrire") ou, dans le cas de génération de parole, une partie équivalente qu'on pourrait appeler articulatoire ("décider comment le prononcer"). Cette division est souvent reproduite dans une implantation logiciel par des

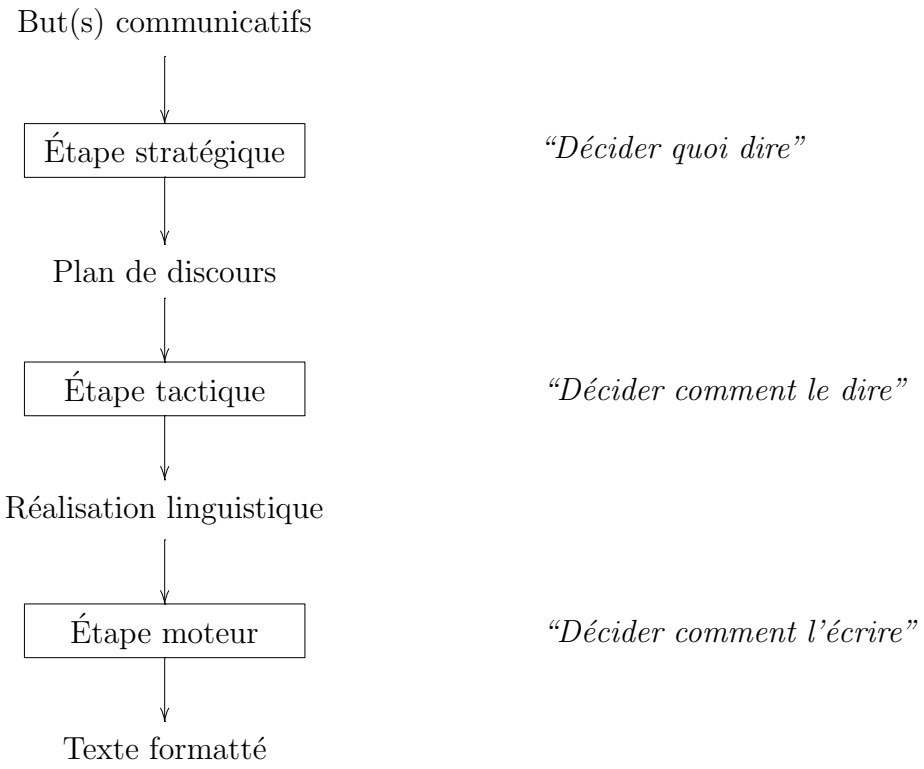

FIG. 2.2 – *Étapes de la génération: stratégique, tactique et moteur*

fonctions de planification de texte et de planification de phrases.

Ce modèle architectural, présenté la figure 2.2, est assez répandu [Reiter, 1994] [Busemann, 1993]. Même si la terminologie pour chacune des étapes de traitement ainsi que leur découpage exact varie d'un chercheur à l'autre, la configuration linéaire "pipeline" semble faire concensus. En réalité, les étapes de génération ne sont pas nécessairement indépendantes puisque les informations de haut niveau peuvent avoir des impacts sur des transformations de bas niveau. Par exemple, un plan rhétorique peut avoir un impact sur le formatage final d'un texte pour indiquer une emphase avec des caractères en italique.

Il y a eu quelques suggestions d'une architecture moins hiérarchisée. On voit mentionné par exemple celle du "blackboard" [Milosavljevic et al., 1996], et celles incorporant de la rétroaction ("feedback") entre les niveaux. Un exemple est le système KAMP [Appelt, 1985] qui utilise un seul moteur global de planification IA pour une bonne partie des tâches de génération. Du point de vue pratique, par contre, cette variante d'architecture de génération reste l'exception plutôt que la règle car elle est très complexe à contrôler.

2.2.1 Étape stratégique

L'étape stratégique du modèle linéaire de génération concerne la planification de texte, l'organisation des structures à grande échelle. À l'entrée, les buts communicatifs guident un processus de sélection des connaissances, c'est-à-dire la détermination des entités et des relations pertinentes. Un modèle de l'interlocuteur, englobant ce qu'il sait, ce qu'il voudrait savoir, et ses intentions (modèle "Belief, desire and intention"), peut guider cette sélection. Un historique des échanges précédents peut donner des indices sur ce qui serait pertinent à dire; d'autres contraintes pratiques, comme la longueur maximale du texte final, les relations inter-personnelles, et le ton général du dialogue [Hovy, 1988] peuvent aussi être pris en compte à cette étape. Le résultat de ce processus est normalement une structure arborescente représentant la structure rhétorique du texte, où chaque feuille de l'arbre correspond à un énoncé (simple). Cette étape est illustrée à la figure 2.3.

Dans les texte plus formels, les chercheurs en génération ont souvent recours à un modèle formel de discours, le RST ("Rhetorical Structure Theory") [Mann et Thompson, 1986] étant un des mieux connus, pour les fins de planification stratégique. Ce modèle sert à spécifier la suite de présentation des idées dans un texte via la satisfaction de relations rhétoriques spécifiques. Ces relations binaires peuvent être établies entre deux idées ou segments de texte, un noyau et un satellite[1]. Les noms de ces relations sont très descriptifs: Élaboration, Exemplification, Contraste, Motivation, Séquence narrative et ainsi de suite. Il est possible d'implanter les mécanismes d'agencement d'idées selon des règles rhétoriques via un module de planification IA. Par contre, il est souvent plus pratique de faire l'implantation des modèles de discours sous forme de schémas prédéfinis et fixes [McKeown, 1985], possiblement avec des éléments conditionnels. Ces schémas sont utiles dans le cas de textes plus longs, où la complexité d'une planification complète et dynamique est trop grande pour être pratique [Hovy, 1993].

2.2.2 Étape tactique

À l'étape tactique, l'arbre représentant la stucture rhétorique instanciée est transformé par un générateur de phrase, implantant la sous-tâche de microplanification ou l'organisation du contenu de chaque phrase. Ensuite, la représentation abstraite de chaque phrase passe par un processus de réalisation linguistique, le résultat étant un arbre syntaxique avec tous les attributs grammaticaux des mots complètement spécifiés. La figure 2.4 donne un aperçu de cette étape.

1. En anglais, la théorie utilise les termes "nucleus" et "satellite".

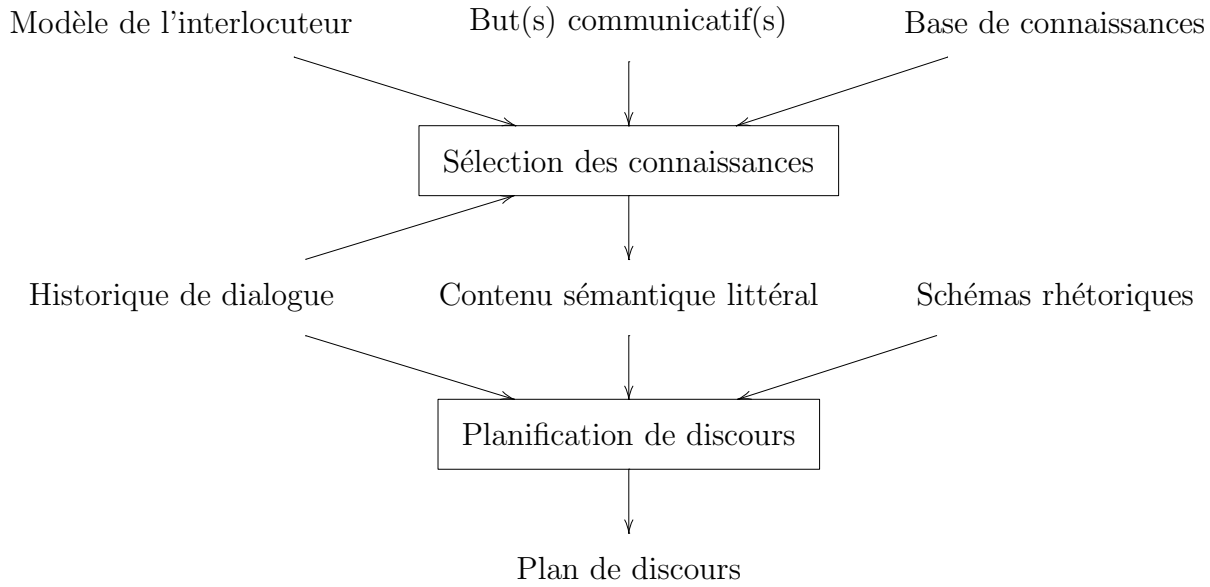

FIG. 2.3 – *Étape stratégique de la génération*

Lors de la microplanification, l'agrégation sert à rendre le texte généré plus lisible, fluide, et compréhensible [Dalianis et Hovy, 1996]. Elle est nécessaire quand une partie de l'information à présenter est redondante ou répétitive, ou bien quand le lecteur peut facilement inférer les informations à partir des autres faits dans le texte [Horacek]. Considérez par exemple le texte suivant (adapté de [Reiter, 1995]):

> *La maison est blanche. La maison est grande. Jean est le propriétaire de la maison. La maison est sur la rue Simard. La maison est à coté d'une école primaire.*

Il peut être transformé via des règles d'agrégation vers le texte suivant:

> *Jean est propriétaire d'une grande maison blanche sur la rue Simard. Elle est proche d'une école primaire.*

Dans l'exemple ci-haut, nous voyons le résultat de l'application de règles d'agrégation sur les énoncés simples (ou des descriptions fonctionnelles, "functional descriptions" [Cancedda, 1999]). On voit qu'il y a eu l'application de règles d'ordonnancement; de regroupement par la subordination ou par l'écrasement; d'élimination de sous-structures communes à plusieurs énoncés; et d'élimination de répétitions de structure communes complètes

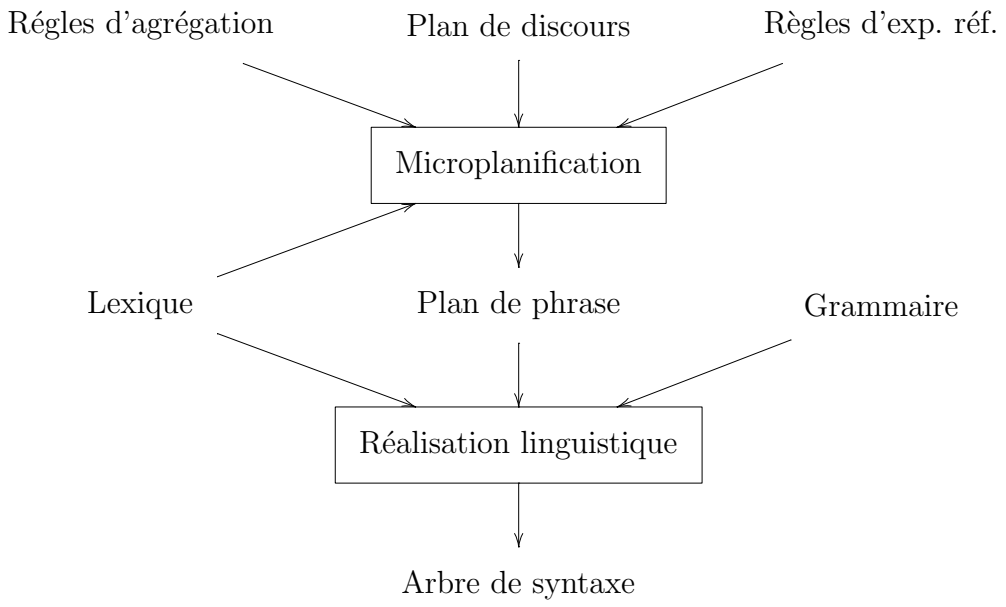

FIG. 2.4 – *Étape tactique de la génération*

[Dalianis et Hovy, 1996]. D'autres termes pour ces trois dernières opérations sont la combinaison de clauses, l'ellipse, et la réduction des conjonctions [Reiter, 1995].

L'application des règles d'expressions référentielles, par exemple l'utilisation de pronoms, réduit la redondance des entités nommées. La substitution de l'expression "La maison" par "Elle" dans l'exemple ci-haut en est un cas typique.

Certains auteurs identifient aussi la sélection du thème et du focus du texte à cette étape tactique.

2.2.3 Étape moteur

Les buts de l'étape moteur sont de faire une réalisation de surface, c'est-à-dire la conversion de l'arbre de syntaxe en une suite de mots fléchis avec une orthographe correcte, ainsi que d'appliquer les étiquettes de formatage ("markup tags") pour les fins d'affichage. Un découpage possible de cette étape est présenté à la figure 2.5. La conversion de l'arbre en suite de mots s'appelle la linéarisation. Elle est obtenue en parcourant l'arbre selon un ordre fixe de nœuds. Par flexion de mots, on entend l'application de règles morphologiques aux racines de mots (par exemple, conjuguer les verbes et former les pluriels). Par orthographe

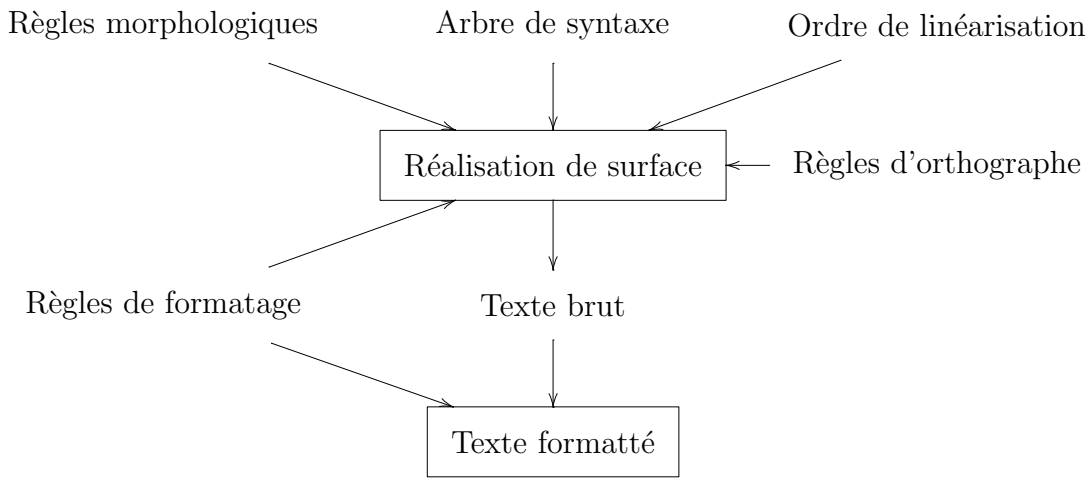

FIG. 2.5 – *Étapes moteur de la génération*

on veut dire le traitement des questions d'élision[2], de contraction[3], de capitalisation et de ponctuation. Le formatage pourrait consister simplement en la séparation des paragraphes et des sections par des lignes blanches pour des applications en texte brut ou, pour des besoins plus sophistiqués, l'ajout d'étiquettes HTML. Avec des étiquettes HTML, on pourrait choisir d'indiquer une emphase avec une police de caractères gras ou italiques.

2.3 Niveaux de complexité de la génération

La génération à base linguistique n'est pas la seule façon de produire du texte dans un programme d'ordinateur. Dans les paragraphes suivants, nous allons examiner les techniques plutôt superficielles (le texte figé, les patrons à trous, les phrases lexicales), la technique profonde, la génération statistique et finalement les techniques hybrides. À la fin de cette section, nous allons discuter des avantages et inconvénients de la génération profonde et celle de surface.

2. e.g. *le + avion ⇒ l'avion*
3. e.g. *de + les ⇒ des*

2.3.1 Le texte figé

La technique de génération la plus simple est celle du texte figé. Ceci consiste en l'utilisation du texte fixe, prédéfini et complètement spécifié des points de vue de la morphologie, de l'orthographe et du formatage. Quoiqu'abordable, rapide, et implantable par tout programmeur dans tout langage, le texte figé a des désavantages majeurs [Dale, 1995]:

- Le texte figé est ... figé. Il offre très peu de souplesse d'expression et est invariable.
- Il faut prévoir d'avance toutes les situations d'utilisation possibles; par exemple, anticiper toutes les questions possibles et en préparer les réponses.
- Les modifications pour des raisons autres que celles du plus simple contenu, par exemple, des changements de langue ou de style, peuvent être très couteuses. La localisation des logiciels en est un exemple.
- Il n'y a pas de garantie que le texte figé sera conforme aux besoins des autres éléments d'un système puisqu'il est créé de toute pièce par un humain et le texte n'est en aucune façon paramétrisé par le système.

On a justement inventé le domaine de la génération de texte en grande partie pour résoudre ces problèmes pratiques de base. Le texte figé a très peu d'intérêt scientifique intrinsèque, mais ses descendants en possèdent sensiblement plus, comme nous allons voir.

2.3.2 Patrons à trous

La plupart des logiciels de traitement de texte moderne (Microsoft Word, Frame, etc.) offrent des facilités de spécification de trous à remplir avec du texte variable. Comme exemple, on peut s'en servir pour générer de la correspondance d'affaires standardisée à partir d'un fichier d'addresses de clients[4]. Ces mêmes logiciels incorporent des langages de scripting pour la spécification d'éléments contextuels et, en fait, pour la modification arbitraire du texte de sortie [Reiter et Dale, 2000]. Évidemment, toute cette fonctionnalité peut aussi être facilement créé par un programmeur dans un langage de programmation conventionnel tel que C++, avec, au besoin, l'apport d'une librairie d'expression régulières. Par contre, les langages de programmation ainsi que les environnements de patrons à trous dans les logiciels de traitement de texte n'offrent aucun support pour les notions proprement linguistiques, que ça soit au niveau syntaxique ou au niveau de la planification de texte. Le support se limite, au mieux, à la mise en majuscules de la première lettre d'une phrase. Certains systèmes de traitement de texte pourraient, en théorie du moins, se servir des fonctions de vérification

4. D'où l'expression anglaise "mail merge".

grammaticale pour faire des accords simples (genre, nombre) dans des patrons mais ça ne semble jamais avoir été fait [Reiter, 1995].

Le système de recherche CLINT [Gedalia et Elhadad, 1996] offre un bon exemple de langage de patrons moderne pour la génération. Ce système, écrit en C++, est un programme autonome et ne semble pas avoir comme but une intégration étroite avec un tiers logiciel de traitement de texte. Il permet l'expansion de patrons par le simple remplacement de chaînes de caractères, des alternances alléatoires (pour ajouter un peu de variété au texte produit), l'expansion conditionnelle et l'imbrication récursive de patrons (permettant ainsi la description compositionnelle des patrons). Un exemple où il y a une intégration étroite avec un système de traitement de texte est Soda[5]. L'idée de ce produit est d'opérer avec le logiciel FrameMaker+SGML[6] dans la préparation semi-automatique de documents techniques structurés dans le domaine du génie logiciel. Soda offre un langage de patrons à trous puissant, surtout en ce qui concerne l'accès aux éléments du dépôt de données de conception de logiciel, mais il n'offre aucune fonctionalité proprement linguistique.

Cette méthode de génération peut être pratique dans certaines situations: elle est peu coûteuse et rapide à déployer. Dans le contexte de systèmes interactifs, elle a un avantage en termes de temps de réponse par rapport aux systèmes de génération plus sophistiquées, ces derniers pouvant être très lents. Le système Écran [Geldof et Van de Velde, 1997] réussit bien à cet égard dans la génération en-ligne de documents hypertextes multimédias. Par contre, il est probablement difficile de configurer un système de patrons à trous pour générer du texte varié avec une bonne qualité de rédaction lorsque l'application devient plus grande et lorsque les informations nécessaires pour décrire le domaine de discours sont variées et complexes. En d'autres mots, il peut s'avérer difficile de gérer simultanément un très grand nombre de patrons interdépendants. Il est probable que les systèmes de ce genre ont des coûts d'entretien plus élevés que les systèmes de génération plus sophistiqués et bien structurés. Malheureusement, il ne semble pas avoir eu d'étude quantitative sur cette question.

2.3.3 Systèmes à base de phrases lexicales

Une grande partie de la génération consiste en la mise en correspondance d'éléments sémantiques d'une base de connaissance avec des phénomènes syntaxiques et lexiques de surface précis. On peut voir cette correspondance comme une correspondance d'atomes dans un langage à des atomes dans un autre et comme une correspondance des relations dans le domaine sémantique à des patrons syntaxiques simples. Une idée proposée à l'origine par

5. Soda est un produit de la firme Rational Software Inc.
6. FrameMaker+SGML est un produit de la firme Adobe Inc.

$$\begin{bmatrix} \text{SEM:} & \text{30CM-DE-LONGUEUR} \\ \text{ORTH:} & \text{“EST 30CM DE LONGUEUR”} \\ \text{SYN:} & \begin{bmatrix} \text{CAT:} & \text{VP} \\ \text{ACC:} & \begin{bmatrix} \text{NOMBRE:} & \text{singulier} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

FIG. 2.6 – *Matrice attribut-valeur simple pour la phrase lexicale*

Becker [Becker, 1975] voulant qu'un lexique puisse contenir des phrases et non seulement des éléments lexiques atomiques (i.e. des mots) a été utilisée avec succès lorsqu'elle a été transposée à ce style de génération.

Souvent les correspondances atomiques sont laborieuses à construire et il s'avère en pratique beaucoup plus efficace d'utiliser des “morceaux” plus larges qu'un seul mot (un concept) ou un patron simple (une relation). Dans le système PEBA-II [Milosavljevic et al., 1996], les chercheurs ont fait une correspondance directe entre le concept sémantique de “30cm_de_longueur” et l'orthographie “est 30cm de longueur”. On voit à la figure 2.6 une représentation de cette idée dans une matrice valeur-attributs. Une matrice valeur-attribut est simplement une représentation graphique pratique d'un de ces morceaux par un groupement logique de paires attribut-valeur [7]. La variante plus complexe de ce même patron, offrant un mécanisme d'accord de nombre, est donnée à la figure 2.7.

Les divers morceaux d'une phrase peuvent être combinés par des mécanismes relativement simples (i.e. l'unification) implantant des règles syntaxiques [McKeown, 1985]. On peut voir cette technique de phrases lexicales comme une précompilation ou évaluation partielle de patrons linguistiques plus fondamentaux mais où un bon nombre de paramètres de génération sont invariants. Certains auteurs appellent cette technique celle des patrons sémantiques [Bateman et Henschel, 1999] [Buseman, 1999].

L'utilisation de ce genre de phrase lexicale a plusieurs avantages:

Performance Il est évidemment beaucoup plus rapide de “cacher” des phrases lexicales précompilées, donnant une vitesse de “génération” beaucoup plus élevée, que de recréer ces mêmes phrases à répétition avec un processus de génération profonde.

Traitment de idiomes Il est très facile de stocker et de générer des expressions idiomatiques fixes.

7. Cette notion de paire valeur-attribut est un élément important de la plupart des formalismes grammaticaux. Les attributs spécifient des catégories (syntaxique, fonctionnel, ou sémantique) et les valeurs spécifient des affectations légales des catégories. Les attributs sont des symboles qui dénotent des catégories comme phrase nominale, nom, protagoniste, but, sujet, etc. Les valeurs peuvent être des symboles ou des sous-grammaires, ceux-ci composées aussi de paires attribut-valeur.

$$\begin{bmatrix} \text{SEM:} & 30\text{CM-DE-LONGUEUR} \\ \text{SYN:} & \begin{bmatrix} \text{CAT:} & \text{VP} \\ \text{ACC:} & \begin{bmatrix} \text{NOMBRE:} & \text{Nombre} \end{bmatrix} \\ \text{VERBE:} & \begin{bmatrix} \text{SYN:} & \begin{bmatrix} \text{CAT:} & \text{V} \\ \text{LEX:} & \text{“ÊTRE”} \\ \text{NOMBRE:} & \text{Nombre} \end{bmatrix} \end{bmatrix} \\ \text{OBJECT:} & \begin{bmatrix} \text{SYN:} & \begin{bmatrix} \text{CAT:} & \text{NP} \end{bmatrix} \\ \text{ORTH:} & \text{“30CM DE LONGUEUR”} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

FIG. 2.7 – *Matrice attribut-valeur profonde pour la même phrase lexicale*

Extraction d'expressions Il est possible de faire de la prospection de texte dans le domaine de discours afin de repérer les expressions les plus communes et de s'en servir, après un minimum de transformation, pour la génération. Cancedda [Cancedda, 1999] se sert cette technique pour la (re-)génération de messages dans le cadre des MUC[8] .

2.3.4 Génération linguistique profonde

Le niveau de génération le plus complexe possible est celui de la génération profonde, situé à l'autre extremité du spectre de complexité par rapport au texte figé. Avec cette forme de génération, on implante tous les niveaux du modèle abstrait de la section 2.2. En théorie, on devrait être en mesure de représenter et de manipuler toutes les granularités de phénomènes linguistiques: communicationnel, sémantique, lexique, syntactique, et même prosodique.

Le point de départ dans la conception de générateurs profonds[9] est le choix d'un cadre théorique linguistique. Il a deux grande familles de cadres, la transformationnelle (Chomsky, USA, années '50) et celle à base de dépendances (Tesnières, années '20, Europe). Ces cadres théoriques sont essentiellement descriptifs et ne disent pas comment rendre leurs notions opérationnelles[10]. L'implantation d'un générateur profond requiert normalement le choix d'une théorie ou d'un formalisme linguistique complet, par exemple la théorie Sens-Texte de Mel'čuk, ou bien le choix d'une combinaison de formalismes indépendants mais compa-

8. “Message Understanding Conference”.

9. Pour un systéme d'analyse de la langue, on passe par essentiellement les mêmes étapes. Il y a des différences importantes entre la génération et l'analyse, mais on ne les discute pas ici.

10. Certaines écoles linguistiques, notamment des branches de la famille transformationnelle et l'école de linguistique cognitive, prétendent pouvoir expliquer les fondements *neuro-psychologique* du langage humain.

tibles, par exemple, la théorie RST pour la partie rhétorique et la linguistique systémique fonctionnelle [Matthiessen et Bateman, 1991] pour le reste. Il faut au moins que les choix résultent en une couverture très large et complète de la langue en termes de phénomènes de communication, de syntaxe, de ton, etc. Il faut ensuite transformer la théorie en grammaire et autre mécanisme formel pour pouvoir l'implanter dans un programme et l'exécuter sur un ordinateur[11]. On pense par exemple aux grammaires SFG, TAG, DG, et HPSG[12], idéales pour l'implantation informatique[13].

Une des conséquences de la large couverture de ces grammaires est qu'il y a un très grand nombre de paramètres ajustables et de règles interdépendantes contrôlant tous les phénomènes de la théorie sous-jacente. Jusqu'à très récemment, le paradigme d'implantation était fortement biaisé vers une approche à base de règles. Cette approche engendre une multiplication de règles pour traiter tous les cas possibles de combinaisons de valeurs de paramètres et donc tout les phénomènes. Certaines grammaires de l'anglais contiennent de dizaines de milliers de règles. Pour faciliter l'implantation logicielle, on a eu souvent recours aux langages de recherche puissants comme Prolog ou LISP, peu utilisés dans le monde commercial[14].

La plupart des avantages de la génération en général énumérés la section 2.1, comme une qualité améliorée et une garantie de conformité à des normes, s'appliquent à la génération profonde. Par contre, il y a d'importants désavantages spécifiques à la génération profonde ([Coch, 1998] et autres):

- Peu de ressources réutilisables sont disponibles et il est présentement impossible d'acheter un artéfact logiciel bien conçu, documenté, robuste et complet pour faire la génération profonde.
- Peu de personnes/organisations peuvent créer de toutes pièces des systèmes de génération profonde.
- Pour faire un générateur, il est nécessaire de disposer de connaissances en intelligence artificielle (pour la sélection et la planification du contenu) et en linguistique formelle

11. L'auteur ne voit pas vraiment comment des grammaires transformationnelles ou cognitives pourraient exécuter dans le supposé lobe de "grammaire universelle" du cerveau, comme prétendent les adeptes de ces théories.

12. Ces sigles signifient "Systemic Functional Grammar", "Tree Adjoining Grammar", "Dependency Grammar", et "Head-driven Phrase-Structure Grammar".

13. En théorie, ces grammaires peuvent exécuter dans un sens pour l'analyse ainsi que dans l'autre pour la génération. En pratique, elles sont très inefficaces et lentes car les deux tâches sont fondamentalement différentes. La génération de texte a une grande composante de planification et une petite composante d'application de règles de grammaire. L'analyse de texte est essentiellement du parsage.

14. Il est possible de satisfaire les mêmes besoins algorithmiques avec des langages conventionnels comme C++ mais le code résultant peut être moins naturel et lisible et donc plus cher de conception et d'entretien.

(pour la planification du texte et de sa réalisation).

- La génération profonde coûte cher si le domaine de discours n'est pas déjà représenté dans l'application de base par une ontologie formelle[15].
- Elle coûte cher si les sorties textuelles n'ont pas de représentation comme phrases dans l'application de base[16].
- Les générateurs profonds requièrent des entrées très complexes. Les concepteurs des systèmes en amont des générateurs doivent générer des sorties complexes, une tâche coûteuse en temps et en effort.
- La génération profonde est toujours expérimentale et donc risquée pour les applications commerciales.

2.3.5 Génération statistique

Une nouvelle approche de génération est apparue dans le domaine depuis environ 1996. L'approche dite statistique à eu du succès dans d'autres domaines du TALN: la reconnaissance de la parole, la recherche d'information, l'étiquetage grammatical, le parsage et la traduction assistée par ordinateur (voir sommaire de discussions menées par Hovy [Impacts–2000, 2000]). Un préalable à toute approche statistique est la disponibilité de corpus d'entraînement, ce qui était jusqu'à récemment un problème majeur mais qui s'estompe avec les années et les efforts d'étiquetage manuel de corpus.

L'auteur croit qu'il existe un potentiel intéressant à cette approche, particulièrement si elle est utilisée conjointement avec les tâches d'extraction d'information et de parsage. Neumann et Flickinger [Neumann et Flickinger, 1999] croient qu'une version stochastique de la grammaire HPSG, spécialisée au niveau lexical par entraînement sur un corpus, pourrait s'ajouter à un méchanisme bi-directionnel uniforme [Neumann, 1998] et accéléré [Neumann, 1997]. Le système résultant serait applicable tant à la génération qu'au parsage. Ces mêmes auteurs croient que les informations linguistiques profondes disponibles via ces SLTG ("Stochastic Lexicalized Tree Grammar") seraient de grande utilité dans l'extraction d'information.

Par rapport à la génération basée sur l'IA classique à base de règles, l'approche de

15. Ou, au minimum, par une structure semi-formelle comme un modèle entité-relation de base de données.

16. Le texte généré par une application typique d'aujourd'hui est typiquement formé par la concaténation de variables (des chaînes de caractères) représentant des segments de phrase. Il est difficile de repèrer des phrases complètes (ou des patrons de celles-ci) du code source d'une telle application. Automatiquement re-implanter les phrases dans une nouvelle version de l'application qui fait de la génération profonde serait donc un tour de force.

génération statistique pourrait offrir des avantages théoriques importants:

- Elle est moins fragile ("brittle") dans le sens où elle peut toujours générer des solutions qui sont approximativement correctes.
- Elle est moins dépendante du domaine de discours puisqu'on peut re-entraîner le système automatiquement à partir d'un corpus; elle est donc plus adaptative.
- Pour les même raisons, elle ne requiert pas un travail intensif de création d'une base de règles (grammaire et autre phénomènes) comme c'est le cas dans l'approche classique; elle requiert seulement (!) un corpus d'entraînement et des techniques d'apprentissage machine adaptées au domaine du langage naturel.
- Elle offre un cadre théorique clair et uniforme (l'apprentissage machine statistique) pour la représentation et l'induction des phénomènes de génération, de parsage et d'autre phénomènes linguistiques.

Pour l'instant, il y a très peu d'exemples de ce genre de système de génération. Aussi, il n'est pas clair avec quel niveau de génération ou avec quelle théorie linguistique une approche statistique aurait le plus de succès.

2.3.6 Les techniques hybrides

Les techniques hybrides sont un mélange judicieux de génération profonde et de surface. Elles peuvent être vues comme un compromis entre la robustesse et la portabilité entre domaines de discours. Elles peuvent offrir une bonne fluidité dans le texte de sortie [Cancedda et al., 1997b]. Examinons les caractéristiques de quelques systèmes hybrides typiques.

Dans le système SAX [Cancedda et al., 1997a] [Cancedda et al., 1997b], on peut combiner des éléments invariables et prédéfinis, des éléments variables instanciés dynamiquement juste avant la génération et des éléments prédéfinis et statiques mais qui s'ajustent au contexte linguistique immédiat. En pratique, ceci veut dire que les éléments variables peuvent être l'objet d'ajustements phonétiques, morphologiques (pour les accords) ou orthographiques. Des schémas statiques de communication, des variantes des schémas rhétoriques mentionnés à la section 2.2.1, définissent le plan de discours et après l'instanciation d'éléments variables, le plan de texte.

Le système GeM [Cancedda, 1999] génère un genre de sommaire à partir de données obtenues par des techniques de surface d'extraction d'information, c'est-à-dire avec des expressions régulières linguistiques. Cancedda a combiné un engin de réalisation profonde, FUF [Elhadad, 1993], avec une librairie de tels patrons d'extraction. Ces patrons ont été créés pour le système FASTUS [Appelt et al., 1993], un système d'extraction d'information

stratifié fonctionnant à base d'expressions régulières. Ces patrons ont servi comme point de départ pour la création automatique de patrons de génération, réduisant ainsi grandement l'effort de conception du générateur.

Le système CLINT [Gedalia et Elhadad, 1996] combine les patrons à trous et la génération de groupes nominaux. La génération de ces derniers tient compte du contexte dans lequel chaque instanciation est utilisée et l'identité de la tête des groupes nominaux. Le système peut générer un pronom, un groupe nominal avec tous ses modificateurs, un groupe nominal avec seulement quelques modificateurs ou même une anaphore en fonction de critères spécifiés par l'usager.

Le système TG/2 [Busemann, 1998] combine la génération profonde, le texte figé et les patrons à trous[17]. Il a été utilisé pour implanter un service de gestion de rendez-vous[18][Busemann et al., 1994] [Busemann et al., 1997]. Seulement quelques actes de discours, environ vingt patrons de phrases et une grammaire complète d'expressions de dates et d'heure étaient nécessaires. Le reste du système est composé de texte figé.

2.4 Génération de surface ou Génération profonde?

Après l'exposé des sections précédentes, il est clair qu'il y a des avantages et désavantages importants à chaque style de génération, profonde ou de surface. Il est aussi clair qu'il y a des différences entre les approches des systèmes de recherche et les approches avec enjeux commerciaux. Mais comment choisir le bon style de génération? Heureusement, certaines personnes ayant déjà créé des systèmes de recherche ont aussi une expérience pratique et commerciale avec la génération de texte. Leur conclusions, parfois étonnantes, peuvent nous être utiles. Coch, œuvrant chez Lexiquest[19] et connu dans le domaine du traitement automatique de la langue, est sans équivoque concernant les avantages relatifs des techniques profondes et de surface:

> Si un organisme décide de développer une application de génération linguistique, il faut soit que cet organisme possède une équipe très spécialisée, soit qu'il sous-traite ce travail à une équipe externe ayant ces compétences, ce qui n'est

17. Le cœur de TG/2 implante, en LISP, l'unification à la Prolog et un système de production. Les aspects linguistiques profonds sont traités par le module grammatical du système DISCO [Uszkoreit et al., 1994]. Cette grammaire est de la famille HPSG épicée avec des éléments d'autres formalismes grammaticaux (notamment de la grammaire catégorielle).

18. Dans cette application, l'heure et la date de rendez-vous pouvaient être proposées, rejetées et modifiées via des communications en langue naturelle par courriel.

19. Cette firme situé à Paris opérait auparavant sous le nom ERLI SA.

> pas le cas de la génération par patrons. À égalité de conditions, une application par *patrons* serait donc préférable à une application linguistique [Coch, 1998].

Reiter [Reiter et Mellish, 1993] tire sensiblement la même conclusion suite à ses expériences avec le système IDAS. Selon cet auteur, les techniques "intermédiaires" de génération pourraient fournir en bonne partie (mais pas toutes) les avantages de la génération profonde, plus rigoureuse sur le plan théorique, mais à des coûts beaucoup moindres. Horacek, un chercheur bien connu dans le domaine de la génération, dit dans un article récent [Horacek et Busemann, 1998]:

> While in-depth approaches to NLG are indispensable for scientific progress in the field, we suggest that certain types of applications can, at present, be built more successfully by systematically adopting more *shallow* techniques, which emphasize domain- and user-specific preferences over general-purpose cummunicative principles.

Buseman croit qu'un large éventail d'applications peut être traité par des techniques relativement simples comme celles utilisées dans ses projets de gestion de rendez–vous [Busemann et al., 1997] et de génération de rapports atmosphériques [Busemann et Horacek, 1997]. Avec Becker [Becker et Busemann, 1999], ce même auteur dit

> Generally the techniques used in NLG applications and application-oriented NLG systems differ from those utilised in research systems. While the latter typically aim at general, in-depth solutions, the former are geared towards solving particular classes of NLG problems. This involves *shallow* generation such as dealing with canned texts or templates rather than choosing freely from the coverage of complex linguistic grammars.

2.5 Sommaire

Dans ce chapitre nous avons fait un aperçu du domaine de la génération de texte. Nous avons évoqué les raisons pour lesquelles la génération de texte, plutôt que le simple texte figé, est intéressante. Nous avons vu les grandes étapes de la génération ainsi que les tâches servant à implanter des sous-phénomènes comme l'agrégation ou la linéarisation.

Notre exposé sur les différents niveaux, ou styles, de génération a démontré une large gamme de solutions en termes de capacité de génération et complexité d'implantation. Ces informations nous permettront de choisir la méthode la plus appropriée à notre application dans les prochains chapitres.

Chapitre 3

Préparation à la génération

Nous quittons maintenant la discussion des travaux antérieurs et des modèles abstraits de génération pour entamer la description de nos travaux. Nous entrons dans le concret du système décrit au premier chapitre. Dans ce chapitre, nous allons expliquer comment les données et les paramètres de génération sont obtenus et comment la vérification des données est faite. Nous mettrons l'emphase sur la sélection de la réponse.

Faisant référence à la figure 1.6, nous supposons que la classification des messages a été faite et que tous les messages à traiter portent sur les problèmes d'impression. Le système doit ensuite extraire les informations intéressantes des messages, les vérifier et les augmenter au besoin, et les mettre dans une forme standardisée, utilisable par les étapes subséquentes de traitement. Dans la section suivante, nous donnons un sommaire de l'EI afin que le lecteur puisse comprendre la terminologie spécifique à ce domaine de recherche et afin que les lacunes propres à l'EI soit claires.

3.1 Extraction d'information

L'objectif de l'extraction d'information (EI) est de repérer, à partir de textes, de l'information pertinente afin de remplir des structures de données fixes et pré-définies. Cette information, une fois trouvée, peut soit être marquée à même le texte, soit être insérée dans des bases de données pour des recherches ultérieures [Pazienza, 1997][Gaizauskas et Wilks, 1998]. Le but de l'EI n'est pas de faire une compréhension complète et exhaustive d'énoncés textuels mais plutôt d'extraire des faits spécifiques de textes stéréotypés, tel que les dépêches de nouvelles ou les messages militaires. Par exemple, d'un corpus de dépêches sur les attentats terroristes, un système d'EI pourrait trouver les noms et le nombre des victimes, les armes utilisées, la date, l'heure et le lieu de l'attentat, les auteurs de l'attentat, etc. Un avan-

tage majeur des techniques traditionnelles d'EI est qu'elles sont très rapides car elles sont basées sur les automates finis et non sur le parsage avec grammaire sophistiqué couvrant des phénomènes linguistiques complexes.

Il y a au moins quatres approches au problème de la construction de systèmes d'extraction d'information:

- Dans l'approche ingénierie des connaissances, la terminologie du domaine de discours et les règles d'extraction sont construites manuellement suite à l'inspection d'un corpus par un expert. L'expert identifie des agencements de termes qui sont des marqueurs pour les informations recherchées et on implante ces patrons lexico-syntaxiques sous la forme d'expressions régulières.
- Dans l'approche d'induction automatique, des méthodes inductives sont utilisées afin de générer les règles d'extraction (c'est–à–dire des expressions régulières) à partir d'un grand corpus d'entraînement annoté.
- Les approches manuelles et inductives peuvent être combinées dans un contexte interactif. Ici un usager annote les textes du corpus de façon incrémentale pour indiquer des exemples d'informations à extraire. De nouvelles règles d'extraction sont induites à chaque nouvel exemple et la boucle est répètée. Cette collaboration entre l'usager et les algorithmes d'apprentissage peuvent converger rapidement vers des règles d'extraction utiles.
- Dans l'approche statistique, des modèles probabilistes de langage, typiquement à base de "n-grammes", sont construits automatiquement à partir d'un très grand corpus. La majorité des essais portent sur l'utilisation de modèles cachés de Markov[1] [Boufaden, 1999]. Un problème majeur avec cette approche est de réunir assez de données pour l'entraînement et le test du modèle probabiliste.

Dans le cas présent, nous avons suivi l'approche d'ingénierie de connaissances parce que notre corpus de messages était trop petit pour les approches d'induction de règles et statistiques.

Patrons d'extraction

Les informations pertinentes repérées par les systèmes d'EI sont habituellement structurées en forme de formulaires ou "patrons" (*"templates"* dans la terminologie des MUC[2]). Chaque patron est une structure de données avec un nombre fini de champs, chacun ayant une valeur et un type de donnée. Un exemple de patron vide se trouve à la figure 3.1. Les

1. En anglais, ces modèles sont désignés "Hidden Markov Model" ou "HMM".
2. "Message Understanding Conference"

valeurs peuvent être un ensemble, une suite d'items, ou une référence vers un autre patron. Il se peut qu'un champ reste vide après l'extraction des données. Le symbole "∅" est pour un champ vide, indiquant qu'aucune donnée n'y a été insérée. La figure 3.2 donne un exemple d'un patron rempli selon les informations de la question "Comment" de la figure 1.2.

patron message	
Champ	*Valeur*
but	ensemble
sujet	ensemble
expéditeur	patron usager
...	...

patron usager	
Champ	*Valeur*
nom	chaîne
adresse courriel	ensemble
laboratoire	ensemble
...	...

patron évènement-impression	
Champ	*Valeur*
expéditeur	patron usager
destination	patron imprimante
fichier_à_imprimer	patron fichier
source	patron machine
...	...

patron imprimante	
Champ	*Valeur*
nom	ensemble
local	ensemble
état	ensemble
dans_queue	patron fichier
...	...

patron fichier	
Champ	*Valeur*
nom	chaîne
format_actuel	ensemble
desired_format	ensemble
taille_actuelle	ensemble
taille_désirée	ensemble
owner	patron usager
numéro_de_job	chaîne
...	...

FIG. 3.1 – *Exemple de patrons d'extraction à remplir*

Ressources

Les champs des patrons d'extraction sont remplis en faisant appel à trois types de ressources (construites à la main dans notre cas):

Lexiques Ceci inclut des listes de noms d'usagers, de laboratoires, de numéros de locaux, et de noms d'imprimantes. Dans le cas d'un système en-ligne, un appel à des bases de données disponibles dans les systèmes d'exploitation[3] peut retourner ces informations.

3. Par exemple, le service NIS "Network Information System" et les commandes UNIX "lpq", "who", etc.

message 1	
but	question comment
expéditeur	patron usager 1
...	...

évènement impression 1	
expéditeur	patron usager 1
destination	patron imprimante 1
fichier_à_imprimer	patron fichier 1
source	∅
...	...

usager 1	
nom	David Smith
adresse courriel	smith@iro.umontreal.ca
laboratoire	∅
...	...

imprimante 1	
nom	hp2248
local	∅
état	∅
dans_queue	∅
...	...

fichier 1	
nom	∅
format_actuel	∅
format_désiré	recto
taille_actuelle	∅
taille_désirée	∅
owner	∅
numéro_de_job	∅
...	...

FIG. 3.2 – *Exemple de patrons d'extraction après remplissage*

Grammaires pour entités nommées Des expressions régulières pour les noms d'imprimantes, ou pour les locaux permettent de reconnaitre des instances de ceux-ci, même s'ils ne sont pas dans les lexiques.

Patrons lexico-syntaxiques Cette technique permet de reconnaitre des entités nommées même si elles ne sont pas repérées par les expressions régulières. On utilise les mots dans l'environnement immédiat d'une chaîne pour déterminer son sens. Par exemple, le patron **imprimante::nom**[4]**dans le local X** nous permet d'inférer que X est le numéro de local de l'imprimante même si X ne suit pas la grammaire pour cette classe d'entité nommée.

Certaines ressources (par exemple les prénoms communs, les indexes géographiques et les expressions régulières pour les dates, les heures, et les numéros de téléphone) sont réutilisables mais d'autres sont spécifiques à l'application et au domaine de discours.

4. La notation A::B veut dire le champ B du patron A.

3.2 Validation sémantique et inférence du discours

Les textes à analyser dans le cadre des recherches en EI sont le plus souvent rédigés de façon formelle et révisés; c'est le cas d'articles de journaux. Dans notre cas, les textes sont rédigés spontanément et souvent sans trop d'attention aux détails grammaticaux. Les textes ne sont donc pas nécessairement bien structurés et contiennent des fautes de vocabulaire et d'orthographe. Même les faits relatés peuvent être erronés. Par exemple, un usager pourrait faire référence à l'imprimante "hp2284" pour désigner la "hp2248".

Les techniques d'EI ne font pas d'analyse sémantique intelligente et l'analyse syntaxique reste rudimentaire. Il n'y a pas de bases théoriques quantitatives à la l'EI qui permettraient d'estimer la performance ou la fiabilité du processus d'extraction (la précision et le rappel). Dans notre cas, nous effectuons deux sortes de validation sémantique:

1. Comparaison des champs les uns aux les autres et contre une base de connaissances afin d'en vérifier la cohérence. Par exemple, si le module d'EI a rempli le patron d'extraction avec les informations **imprimante_1::nom = hp2248** et **imprimante_1::local = X–234** mais que la base de connaissances ne contient pas la paire **imprimante_1::nom = hp2248** et **imprimante_1::local = X–234**, alors une incohérence est détectée.
2. Exécution de commandes ou de procédures associées aux champs des patrons. Par exemple, si le module d'EI a rempli les champs **imprimante_1::nom = hp2248** et **imprimante_1::dans_queue::nom = test.txt** mais que le script pour les noms d'imprimantes ne peut pas trouver le fichier spécifié dans sa queue (via la sortie de la commande **lpq -P imprimante_1::nom**), alors une incohérence est détectée.

Les patrons incorrects sont marqués et envoyés directement au module de sélection de réponse, où le message "Informations incorrectes" sera choisi (voir la section 3.4). Par contre, les patrons validés sont assujettis à l'étape d'inférence de discours. Le rôle de cet étape est d'inférer des informations ou des relations intéressantes qui ne sont pas données de façon explicite dans le message mais qui peuvent être déduites à partir de connaissances sur le domaine de discours. Par exemple, le fait que **David Smith** est membre du laboratoire d'**intelligence artificielle** peut être déterminé via une recherche dans une base de données sur les coordonnées des usagers. Cette information ainsi des détails sur le fichier dans la queue de l'imprimante (obtenues via la commande **lpq**) sont indiqués en caractères gras à la figure 3.3. Les informations rajoutées peuvent aussi être des valeurs de défaut pour certains champs.

Dans un système en ligne, il serait très facile de vérifier ou d'inférer des informations dynamiques, comme l'état d'une queue d'impression. Malheureusement, puisque notre prototype

fonctionne à base d'un corpus archivé depuis quelques années, il n'y pas moyen de valider ces informations transitoires et éphemères. Nous nous sommes donc limité à la vérification de certains champs des patrons contre une base de données statique, traitant surtout de l'identité et des attributs des imprimantes. Puisque les données d'entraînement et de test ont été préparées manuellement, cette vérification à été faite par l'auteur.

usager 1	
nom	David Smith
adresse couriel	smith@iro.umontreal.ca
laboratoire	**intelligence artificielle**
...	...

imprimante 1	
nom	hp2248
local	**P-204**
état	**impression en cours**
dans_queue	**fichier 2**
...	...

fichier 1	
nom	**test.txt**
format_actuel	∅
format_désiré	∅
taille_actuelle	**8.5x11**
taille_désiré	**8.5x11**
owner	∅
numéro_de_job	∅
...	...

FIG. 3.3 – *Exemple de patrons d'extraction après analyse du discours*

3.3 Paramètres stylistiques

Dans une situation idéale, il serait avantageux de pouvoir ajuster des paramètres stylistiques avant de lancer le processus de génération. Par exemple, on pourrait choisir des niveaux de langage, de détail, ou de formalité pour chaque réponse afin qu'ils s'accordent bien avec le style du message de l'utilisateur. La difficulté avec les paramètres stylistiques ne réside pas dans leur utilisation dans la génération[5] mais plutôt dans la détermination de leurs valeurs. Pour faire une telle détermination, il faudrait être en mesure d'identifier le ton (par exemple, sérieux ou humoristique) du message ou d'assigner un niveau de connaissance (expert ou néophyte) à l'expéditeur du message. Ceci est clairement impossible dans l'état actuel de l'analyse de la langue naturelle. Malgré l'utilité théorique des paramètres stylistiques, nous ne tentons pas d'en calculer avant le processus de génération car nous disposons de trop peu d'exemples de messages dans notre corpus pour en dégager des règles significatives. Ceci ne nous empêche pas d'ajouter quelques éléments aléatoires et conditionnels

5. Au pire, on pourrait écrire des patrons différents pour chaque combinaison de valeurs de paramètres.

afin de donner un peu de variété aux textes générés sans pour autant en changer le contenu sémantique.

3.4 Sélection de la réponse

Lors de la sélection d'une réponse, on aimerait traiter les quatre situations suivantes:

Réponse trouvée: Les patrons d'extraction contiennent des informations complètes et validées et le module de sélection de réponse peut trouver un patron de réponse convenable. Dans ce cas–ci, les informations variables à mettre dans réponse sont passées à l'étape de génération de texte et le courriel de réponse est envoyé.

Faire suivre au préposé: Les patrons d'extraction contiennent des informations correctes et validées mais le module de sélection ne peut pas trouver une réponse convenable. Une réponse indiquant que le message sera traité par un préposé est renvoyée au client.

Informations incorrectes: Les patrons d'extraction contiennent des informations incorrectes. Cette situation peut être le résultat d'une véritable erreur de la part de l'expéditeur du message ou le résultat d'une erreur lors de l'EI. Puisqu'il est impossible de déterminer la source de l'erreur, un message expliquant la situation est envoyé à l'usager.

Informations incomplètes: Les patrons ne contiennent pas assez d'information pour faire la sélection d'une réponse. Encore une fois, il est impossible de déterminer si l'information est manquante dans le message original ou si elle n'était pas repérée lors de l'extraction. Un message expliquant la situation est expédié.

Dans une implantation commerciale, il serait souhaitable de traiter ces quatres situations, surtout avec un grand volume de messages. Malheureusement, notre corpus contient un nombre restreint de messages et, comme nous allons voir à la chapitre 6.1, toutes les données dans les patrons d'extraction ont été créées à la main. Nous avons donc simplifié ces cas en fusionnant les trois derniers en un seul cas "Faire suivre au préposé".

Mécanisme de sélection

L'implantation la plus simple d'un mécanisme de sélection de réponse est la construction manuelle d'une base de cas. Ceci consiste en une suite de tests sur les valeurs dans les patrons d'extraction. Les tests ainsi que leur ordre d'évaluation est établi par le programmeur à partir d'une banque d'exemples classifiés manuellement. Les cas et les tests sont ordonnancés à partir des cas les plus spécifiques vers les classes les plus génériques. Le programmeur peut combiner plusieurs exemples semblables en une catégorie en fusionnant des combinaisons de

tests. Le cas de défault est choisi quand tous les tests précédents échouent, dans notre cas, “Faire suivre au préposé”.

On peut aussi utiliser des techniques automatiques. Une mémorisation complète des cas d’entraînement peut donner une structure semblable à la construction manuelle décrite précédemment. Une technique d’apprentissage machine, par exemple l’induction d’arbres de décision, comprime une base de cas (contenant possiblement des incohérences) vers une structure de données plus évoluée. Si des grandes quantités de données d’entraînement sont disponibles, il devient aussi possible d’établir des bornes théoriques sur l’erreur de généralisation et de calculer une similitude numérique entre le nouveau cas et la catégorie choisie par l’algorithme. Dans le cas des arbres de décision, il est aussi possible d’utiliser les nœuds internes de l’arbre et non juste les feuilles. Lors de l’évaluation d’un nouveau cas, l’arbre est traversé de sa racine jusqu’à une feuille via des nœuds internes où des tests sur les attributs sont effectués. S’il n’y a pas assez de données pour faire un test à un nœud donné, un transfert vers un patron de réponse “Informations incomplètes” pourrait être fait.

Étant donné la petite taille des corpus (122 et 69 messages) et leur nature bruitée, il est fort improbable que les techniques d’apprentissage machine puissent dégager des régularités fiables et intéressantes. Nous avons donc adopté la stratégie manuelle de construction d’une base de cas.

Chapitre 4

Génération de réponses

Dans ce chapitre, nous allons caractériser les besoins de notre système du point de vue de la génération de texte. Nous allons ensuite justifier notre choix d'approche à la génération ainsi qu'expliciter les caractéristiques de l'implantation.

4.1 Expériences initiales

Comme première étape, nous avons évalué la qualité des réponses générées par un prototype très simple par rapport au besoins de l'application. Le but était de comprendre la difficulté de générer quelques réponses à partir de patrons d'extraction fictifs préparés manuellement (tel qu'expliqué dans la section 6.1). Ce prototype consistait en un nombre limité de patrons de réponses, composés en large partie de texte figé, codé en Prolog. Les résultats qui s'en dégageaient étaient très probants: les patrons de phrases avec trous ainsi qu'une structure rhétorique fixe semblaient parfaitement adéquats pour les fins de notre application.

L'explication de ce phénomène est assez simple:

- Le nombre de types de réponses est très limité car le domaine de discours est très restreint.
- Il est facile de mettre toutes les informations nécessaires pour répondre aux questions fictives dans des réponses courtes.
- Il ne semble pas nécessaire de faire une planification dynamique de document puisque les réponses originales sont stéréotypées.
- Des techniques avancées de génération ne semblent pas nécessaires. Nous n'avons repéré aucune circonstance où elles auraient été utiles. Par exemple, il semble superflu de faire l'agrégation dynamique de phrases courtes puisqu'on peut déterminer à l'avance

le contenu des réponses.

– Notre système ne fait pas de gestion de dialogue – toutes les réponses générées sont directes et elles mettent fin à la "conversation" entre le système de réponse automatique et l'expéditeur. Nous évitons ainsi certaines représentations de connaissances complexes (en autres, les actes de parole) ainsi que les techniques de génération profonde pour paraphraser des éléments des messages précédents.

Comme notre application n'est pas très différente de ces tests, nous avons choisi un système de génération de surface.

4.2 Technique de génération retenue

Comme Horacek [Horacek et Busemann, 1998], nous avons adopté une méthodologie de développement pragmatique comportant:

– l'utilisation d'une ontologie *minimale* pour représenter le domaine de discours (ces connaissances sont réflétées dans les patrons d'extraction et dans les structures de données intermédiaires utilisées durant le traitement)

– des paramètres de génération simples, facilement interprétés par les non-linguistes (i.e. pas d'attributs ou de valeurs profonds comme agent, but cummunicatif, etc.)

– l'utilisation de schémas rhétoriques fixes plutôt qu'un mécanisme de planification dynamique de discours

– l'utilisation d'un raffinement successif manuel de cette représentation initiale par la définition de segments de phrases et de phrases complètes *figées*

– l'utilisation *opportuniste* des méthodes motivées par la linguistique en combinaison avec les méthodes basées sur les patrons à trous.

Ce compromis pragmatique de méthodes de génération produit des résultats de bonne qualité avec un minimum de complexité.

4.3 Patrons de réponses

Pour nos patrons de réponse, nous avons étudié le style et le contenu du corpus de réponses envoyées par le personnel du support technique. Cette démarche empirique pour la construction des schémas est semblable à celle qui est utilisée depuis les premiers systèmes de génération. Par exemple, Cancedda [Cancedda et al., 1997b] a procédé au même genre d'analyse pour le système SAX, où les schémas ont été identifiés suite à une analyse de diagrammes de conception de systèmes d'information. Busemann [Busemann, 1998] a étudié

son corpus et a reproduit les patrons de texte les plus fréquents: il a toutefois répertorié un plus grand nombre (sept) de schémas et des schémas plus complexes avec un plus grand nombre d'éléments variables que dans notre corpus. La technique est semblable: tous les schémas viennent du corpus et non d'un modèle explicite de l'usager ou d'un moteur de raisonnement abstrait.

Schéma rhétorique

Nous avons développé le schéma très régulier présenté à la figure 4.1. La correspondance entre cette structure rhétorique et une réponse idéale est donnée à la figure 4.2. <accueil>, <fermeture>, <signature> sont simples. <accueil> est un patron à trou avec en option le nom de la personne. <fermeture> et <signature> peuvent être du texte figé, possiblement avec quelques variantes aléatoires pour donner un peu de variété aux textes. Les autres éléments de cette structure sont expliqués plus loin.

```
<message> ::= <accueil> <réponse> <fermeture> <signature>

<réponse> ::= <réponse_trouvée> |
              <préposé> |
              <infos_incorrectes> |
              <infos_incompletes>

<réponse_trouvée> ::= <phrases_principales>
                      [ <considérations_spéciales> ]
                      [ <url_pour_renseignements> ]
```

FIG. 4.1 – *Structure rhétorique des réponses*

Symboles	Réponse à générer
<accueil>	Bonjour David,
<phrases_principales>	Pour imprimer en recto seulement sur la hp2248, il faut utiliser la commande lpr avec l'option -i-1. Faire: lpr -P hp2248 -i-1 <nom du fichier>
<considérations_spéciales>	Notez que ce mode d'impression ne doit être utilisé que pour la copie finale d'un document.
<url_pour_renseignements>	Pour plus d'info, consutez l'URL: www.unURL/lpr#recto
<fermeture>	Bonne chance,
<signature>	Support technique

FIG. 4.2 – *Réponse à générer pour la question "comment" de la figure 1.2*

Types de réponses

Tel que discuté à la section 3.4, nous n'avons retenu que deux des quatre types de réponses qu'on voudrait avoir dans une implantation idéale: <réponse_trouvée> et <préposé>.

Le type <préposé> est simple. Un texte figé indique que le système a identifié le problème et qu'il fait suivre le message à un préposé. Il suffit d'indiquer qu'un préposé (humain) traitera la requête de l'usager dans les plus brefs délais. Le texte de la réponse n'offre pas d'interprétation plus profonde de la situation. Le générateur choisit de façon aléatoire entre quelques variantes textuelles de ces intentions communicatives.

Le type <réponse_trouvée> est le cœur de notre application. La partie <considérations_spéciales> est implantable par du texte figé. Elle correspond à des instructions et des informations non-variables qu'on pourrait trouver dans une page FAQ, par exemple. Les autres parties sont des patrons à trous, la plus simple étant <url_pour_renseignements> et la plus complexe, <phrases_principales> . Pous plusieurs questions, ce dernier segment décrit des instructions pour l'utilisation de commandes spécifiques.

Pour chaque patron, il y a des segments de texte figé et des trous. Les trous peuvent être instanciés par des paramètres ou par une valeur de défaut. Par exemple, à la figure 4.3, la variable `imprimante::nom` prendra la valeur du nom de l'imprimante passée en paramètre de génération. Si la variable est vide, la chaîne de défaut "<imprimante>" sera utilisée. Des chaînes facultatives peuvent être incluses selon le résultat d'une sélection aléatoire.

Avec une telle approche de composition d'éléments variables, optionnels et aléatoires, on peut voir qu'il est possible de construire rapidement un grand nombre de patrons. Nous avons implanté ce "langage" de patrons de la figure 4.3 en règles DCG[1] de Prolog.

4.4 Ajustements linguistiques et forme moteur

Au début du projet, nous avions pensé nous servir de la génération linguistique pour construire, au minimum, des groupes nominaux bien formés (comme dans CLINT [Gedalia et Elhadad, 1996]). Par exemple, nous aurions pu faire des accords de nombre en fonction de la cardinalité des options données à la section précédente. À la limite, nous aurions pu trouver un prétexte pour conjuger des verbes. À cette fin, un module linguistique existant, Frana [Contant, 1985], aurait été suffisamment puissant pour traiter toutes les situations envisageables. Or, comme dans le projet STOP [Reiter, 1999], nous avons réussi à

1. En anglais, "DCG" signifie "Definite Clause Grammar", ou grammaire à clauses définies

Spécification de patron et trou	Sens
Pour imprimer `commande::description`	Ici la variable est obligatoire, p.e. “en mode recto-verso”.
sur [la [imprimante]] `imprimante::nom`,	Pourrait être “la”, “l’imprimante” ou rien mais doit inclure le nom de l’imprimante.
il faut utiliser [la commande] `commande::nom`	La chaîne “la commande” est optionnelle avant le nom (obligatoire) de la commande, par exemple “lpr”.
avec [l’option] `commande::options`.	La chaîne “l’option” est optionnelle. La variable commande::options pourrait avoir la valeur “-i -1”, par exemple.
Faire: `commande::nom` [-P `imprimante::nom`] `commande::options` `fichier::nom`	Ici, seulement la spécification de la destination est optionnelle.

FIG. 4.3 – *Sémantique de la spécification des patrons à trous*

concevoir des patrons de réponse qui ne requièrent aucune application de règles syntaxiques. Il s’est avéré simple de construire des patrons lisibles qui produisent toujours une sortie grammaticalement correcte, peu importe la valeur des paramètres d’instanciation des patrons. Nous avons donc eu aucun recours à la génération linguistique.

Comme nous avons vu à la section 2.2.3, les règles d’élision, de contraction, de capitalisation et de ponctuation sont appliquées à l’étape moteur de génération. Il n’est donc pas nécessaire d’appliquer ces règles directement dans la spécification de patrons de réponses (c’est–à–dire, dans le code Prolog). Dans les patrons, il suffit de spécifier la suite des mots dans leur forme de base car dans le moteur de génération, nous utilisons un module spécialisé pour faire la réalisation moteur et l’application des règles mentionnées ci-haut.

4.5 Conclusion

Suite aux essais de notre prototype, nous avons dégagé les grandes lignes suivantes pour guider l’implantation finale du générateur de texte. Avec des patrons bien conçus, il est possible de contourner la génération linguistique. On évite ainsi le traitement des conjugaisons de verbes et même des accords de nombre et de genre. L’application automatique de règles pour générer la forme moteur finale simplifie un peu la spécification des patrons de réponse. Même si nous n’avons repéré aucune circonstance où cela aurait été nécessaire, il est assez évident que dans ce genre d’application, les énoncés agrégés ainsi que les expressions référentielles (par exemple, la pronomialisation) peuvent être codés directement et manuellement dans des patrons variables sans programmation additionnelle. Nous avons ainsi choisi une approche à la génération simple, rapide et compréhensible.

Chapitre 5

Implantation

Dans ce chapitre, nous décrivons l'implantation en Prolog de l'analyse présentée au chapitre précédent.

5.1 Extraction d'information

Afin d'effectuer l'analyse des questions, le système de réponse automatique prévoyait utiliser un logiciel conçu à l'Université de Montréal. Exibum [Kosseim et Lapalme, 1998] est un système d'extraction bilingue (anglais et français) se spécialisant dans le domaine des attentats terroristes. Nous avons commencé son adaptation afin de pouvoir traiter un domaine différent (dans le cas présent, les messages concernant les problèmes d'impression au DIRO). Comme le développement du module d'EI était moins avancé celui du générateur de réponses, les données d'entrée au générateur ont été préparées manuellement, tel qu'expliqué à la section 6.1.

5.2 Utilisation de Prolog

Le langage Prolog implante déjà toute la richesse fonctionnelle requise par un moteur de génération de surface. Par exemple, les schémas rhétoriques ainsi la composition des éléments des patrons peuvent être codés directement en forme d'une DCG. Les facilités pour ce genre de grammaire sont disponibles dans la version de Prolog que nous utilisons, celle de SICStus.

Les extraits de notre code aux figures 5.1 – 5.6 donnent un aperçu de l'implantation des patrons. On peut y voir la composition via les règles DCG, de segments de texte figé, et des patrons à trous. À la figure 5.1, on peut voir le schéma rhétorique très clairement ainsi le noyau des réponses dans les prédicats `template_reponse`. Ce prédicat avec la variable

anonyme, `template_reponse(_)` (ligne 21), permet la définition de la réponse générique et la génération de ce noyaux se termine à cet endroit avec le texte qu'on peut lire dans le corps de la règle DCG. Le prédicat `template_reponse(format_recto)` (ligne 14) est plus intéressant car il marque le début d'une composition de prédicats de génération. Nous avons inclus tous les prédicats pour la composition de cette réponse afin que le lecteur comprenne le mécanisme de génération en cascade. Les figures sont ordonnées afin de faciliter la lecture du code. Les détails sur le fonctionnement du sélecteur de réponse `arbre_de_decision` seront donnés à la section 5.6.

Les figures 5.2 et 5.5 montrent comment spécifier, avec le prédicat `random`, des variations aléatoires dans le texte à générer. Différentes façons d'afficher les même informations peuvent aussi être définies par le programmeur: par exemple à la figure 5.3, nous présentons la spécification d'une chaîne par défaut <`imprimante`> (ligne 69) et la recherche de paires valeur:attribut dans le patrons d'extraction via les prédicats `member`[1] et `val`. La figure 5.4 donne quelques détails sur ce mécanisme de recherche des paires attribut:valeur dans la structure `te_imprimante`. La variable `Nom` prend la valeur `hp2248` après l'unification dans la règle `affiche_imprimante(_,2)` (ligne 81). Les segments de texte fixe peuvent être codés directement dans les règles DCG mais il est souvent plus pratique de mettre les segments plus longs à part (voir la figure 5.6).

5.3 Non-utilisation de FRANA

Au début de ce projet, nous pensions utiliser la génération linguistique avec FRANA [Contant, 1985]. Ce logiciel est une version française du générateur de texte ANA de McKeown [McKeown, 1985]. FRANA était à l'origine implanté en OPS-5 mais il a été traduit par la suite en Prolog. Dans cette version, il y a des prédicats pour faire la flexion de mots en français, c'est–à-dire conjuger les verbes ou assigner le genre ou le nombre aux noms, adjectifs, etc. Comme nous en avons discuté à la section 4.4, il ne semble pas nécessaire de satisfaire ces contraintes morphologiques par des techniques sophistiquées, comme celles implantées dans FRANA. La génération linguistique n'a pas été implantée dans notre système parce qu'il y avait très peu d'endroits dans les textes où les formes fléchies devaient être produites à partir de racines (mots non fléchies). Lorsque c'était le cas, il était facile de spécifier les bonnes formes directement dans les schémas des phrases. Nous avons donc fait aucune utilisation de FRANA ou d'autre module linguistique.

1. Le prédicat `member(A,B)` retourne vrai si élément A est membre de la liste B.

5.4 Utilisation de SPIN

La réalisation moteur, c'est–à–dire la conversion de texte brut en texte mis en forme, a été accomplie par un module du programme SPIN [Kosseim et Lapalme, 2000]. Ce module se charge des questions d'orthographe comme l'elision, la contraction, la capitalisation et la ponctuation. Ce module a été utilisé tel quel, sans modification.

5.5 Entrées et sorties

Pour notre prototype, le but n'était pas d'optimiser la convivialité des entrées et des sorties. Nous nous sommes donc limité à des fichiers ASCII avec un formatage minimal. Les entrées réelles du générateur sont des données de patron d'extraction créés à la main; voir la figure 6.2.

5.6 Sélecteur de réponse

Un élément très important du programme est le sélecteur de réponse, implanté dans le prédicat `arbre_de_decision`. Un extrait du code pour ce prédicat est donné à la figure 5.7. Les cas sont triés du plus spécifique (`probleme_urgent`, ligne 153) au plus générique (`prepose`, ligne 183) et le balayage séquentiel de la base des cas est assuré par les mécanismes standards de Prolog. Dans chaque cas sauf le dernier, il y a au moins un test de valeur qui est effectué. Les tests sont faits par la recherche dans un patron spécifique (par exemple, `te_imprimante`, ligne 154) pour des paires valeur:attribut (par exemple, `imprimante_etat:probleme_urgent`, ligne 156). Dans certains cas, on peut trouver plus d'une paire valeur:attribut, comme pour `probleme_imprimante` (ligne 171).

Cette base de cas a été construite manuellement et de façon incrémentale. À chaque ajout d'un cas, il fallait vérifier qu'on n'appliquait pas les mêmes tests que ceux d'un cas existant. Il était aussi important d'incorporer le nouveaux cas au bon endroit de la liste sinon une réponse trop générique aurait pu être sélectionnée.

Nous avons fait quelques expériences infructueuses pour la construction automatique de la base de cas. Nous avons réorganisé les données des patrons d'extraction pour qu'elles soient compatibles avec l'entrée du programme d'induction d'arbres de décision C4.5 [Quinlan, 1993]. Malheureusement, la petite taille du corpus d'entraînement ainsi que la forme de certains champs (surtout les messages d'erreur) empêchaient l'inducteur de fournir des arbres performants.

```
repond_print -->
        salutation,
        reponse(T),
        considerations_speciales(T),
        plus_de_info(T),
        [nl, nl],
        fermeture,
        signature.

reponse(T) -->
        {arbre_de_decision(T)},
        template_reponse(T).

template_reponse(format_recto) -->
        { recto_verso_defaut(A) },
        [A,nl,nl],
        affiche_but_format(format_recto),
        affiche_commande(format_recto),
        ['.'].

template_reponse(_) -->
        [un, prépose, repondra],
        [à, votre, problème, de, impression],
        bientot,
        ['.'].
```

FIG. 5.1 – *Structure rhétorique des réponses en Prolog*

```
26  affiche_but_format(F) -->
        {random(1, 3, Random)},
28      affiche_but_format(Random, F).

30  affiche_but_format(1, F) -->
        [pour, imprimer, en, format],
32      affiche_format(F),
        affiche_imprimante('sur la', 1).
34
    affiche_but_format(2, _) --> [].
36
    affiche_format(format_recto) --> [recto, seulement].
38
    affiche_commande(X) -->
40      utiliser_cmd,
        commande(X).
42
    commande(format_recto) -->
44      ['lpr -P'],
        affiche_imprimante([], 2),
46      ['-i-1 <fichier>'].

48  utiliser_cmd -->
        {random(1, 3, Random)},
50      utiliser_cmd(Random).

52  utiliser_cmd(1) --> [utilisez, la, commande].

54  utiliser_cmd(2) --> [faites, ':'].
```

FIG. 5.2 – *Patrons de réponse en Prolog*

```
affiche_imprimante(Prefixe, 1) -->
        { te_imprimante(I),
          member(M, I),
          val(M, imprimante_nom, N)},
        [Prefixe, N].

affiche_imprimante(_, 1) --> [].

affiche_imprimante(_, 2) -->
        { te_imprimante(I),
          member(M, I),
          val(M, imprimante_nom, Nom)},
        [Nom].

affiche_imprimante(_, 2) -->['<imprimante>'].

affiche_imprimante(Prefixe, 3) -->
        { te_imprimante(I),
          member(M, I),
          val(M, imprimante_nom, Nom)},
        [Prefixe,la,imprimante, Nom].

affiche_imprimante([], 3) --> [].

affiche_imprimante(Prefixe, 3) -->
        [Prefixe,la,imprimante].
```

FIG. 5.3 – *Patron avec élément optionel et valeur par défaut*

```
    affiche_imprimante(_, 2) -->
82          { te_imprimante(I),
              member(M, I),
84            val(M, imprimante_nom, Nom)},
            [Nom].
86
    val([Att:Val|_], Att, Val).
88  val([_|R], Att, Val) :- val(R, Att, Val).

90  te_imprimante([
                  [imprimante_etat:probleme],
92                [imprimante_nom:hp2248],
                  [imprimante_etat:probleme],
94                ]).
```

FIG. 5.4 – *Mécanisme valeur-attribut en Prolog*

```
    salutation -->
96          {random(1, 4, Random)},
            salutation(Random),
98          [nl, nl].

100 salutation(1) --> [bonjour],  affiche_prenom(opt), [','].
    salutation(2) --> [salut],  affiche_prenom(opt), [','].
102 salutation(3) --> affiche_prenom(obl), [','].

104 fermeture -->
            {random(1, 4, Random)},
106         fermeture(Random),
            [nl, nl].
108
    fermeture(1) --> ['Bonne chance!'].
110 fermeture(2) --> ['Merci.'].
    fermeture(3) --> ['Cordialement,'].
112
    signature --> ['--', nl,'Support', 'Technique', nl, nl].
114
    plus_de_info(format_recto) -->
116         affiche_info('www.URL/lpr/#recto').

118 affiche_info(A) -->
            {random(1, 3, Random)},
120         affiche_info_rand(Random, A).

122 affiche_info_rand(1, A) -->
            [nl,nl,pour, plus, de , information, consultez, ':', A,'.'].
124
    affiche_info_rand(2, A) -->
126         [nl,nl,voyez,  A, pour, plus, de, info,'.'].
```

FIG. 5.5 – *Variations aléatoires en Prolog*

```
128  recto_verso_defaut(A) :-

130          name(A,

132  "L'impression d'un fichier texte s'effectue à raison
     de 2 pages par côté de feuille sur toutes les imprimantes.
134  Par surcroît, les imprimantes hp2248, hp3252 et hp2153
     impriment recto-verso par défaut."
136              ).

138  considerations_speciales(format_recto) -->
             {considerations_speciales_format_recto(A)},
140          [nl],[nl],[A].

142  considerations_speciales(_) --> [].

144  considerations_speciales_format_recto(A) :-

146          name(A,

148  "Cette commande ne fonctionne que sur les fichiers en
     format texte ou en format PostScript structuré. Nous
150  considerons que ce mode ne doit être utilisé ..."
     ).
```

FIG. 5.6 – *Text figé en Prolog*

```
arbre_de_decision(probleme_urgent) :-
        te_imprimante(F),
        member(M1, F),
        val(M1, imprimante_etat, probleme_urgent).

arbre_de_decision(format_couleur) :-
        te_fichier(F1),
        member(M1, F1),
        val(M1, fichier_format_page_desire, format_couleur).


arbre_de_decision(grosse_impression) :-
        te_imprimante(F1),
        member(M1, F1),
        val(M1, imprimante_etat, grosse_impression).

...

arbre_de_decision(probleme_imprimante) :-
        te_imprimante(F),
        member(M1, F),
        val(M1, imprimante_etat, probleme),
        member(M2, F),
        val(M2, imprimante_nom, _).

arbre_de_decision(E) :-
        te_imprimante(F),
        member(M, F),
        val(M, imprimante_etat, E).

arbre_de_decision(prepose).
```

FIG. 5.7 – *Extrait du sélecteur de réponse en Prolog*

Chapitre 6

Évaluation

6.1 Données d'entraînement et de test

Le corpus d'entraînement comprenait 122 messages et le corpus de test, 69 messages, tel qu'expliqué à la section 1.3.1. Nous avons constaté que le corpus était trop petit pour faire le développement systématique de patrons d'extraction. Cette situation était agravée par le fait que les textes étaient très "bruités". Puisque les travaux sur Exibum n'étaient pas suffisamment avancés pour donner des patrons d'extractions remplis avec une précision acceptable, nous avons procédé à la préparation manuelle des données des patrons d'extraction. Ce travail nous a permis de tester des idées pour le système en aval de l'EI.

Notre critère était de repérer et d'extraire des informations qu'un système automatique serait raisonnablement en mesure de faire. La figure 6.1 donne un exemple du genre de mots clés et d'expressions repérés dans les questions. Les éléments en caractères gras et soulignés sont utilisés tel quels dans les patrons ou bien servent à faire des inférences très évidentes. Les éléments en italique font partie d'un message d'erreur en anglais[1]. La figure 6.2 donne le patron rempli correspondant. Les données sont simplement des clauses Prolog préparées avec un éditeur avec du formatage pour en faciliter la lecture.

6.2 Résultats

La base de cas à été construite en utilisant les données d'entraînement. Nous avons ajouté (en Prolog) des cas entraînement et des tests sur les valeurs de patrons jusqu'à ce que

1. Si la langue de chaque ligne du texte était identifiée, un extracteur automatique pourrait facilement séparer les messages de système (en anglais) du texte principal de l'usager (en français). Le programme SILC, développé au RALI et commercialisé par Alis Technologies, peut accomplir cette tâche d'identification.

```
1   Subject: Un petit problème d'impression
2   Bonjour,
3   Il y un problème avec l'impression d'un fichier PostScript. Le fichier
    s'affiche sans problème dans Ghostview, mais lors de l'impression, ce
    message est envoyé
4   --
5   Your printer job (1648-1) could not be printed
6   Messages from output filter: /usr/local/bin/PostScript/postio: PostScript
    Error
7   --
8   Le fichier se trouve dans ~quidam/Pub/proceedings.ps.Apparemment, ce
    fichier a été généré grâce à (malgré!) FrameMaker. Ce serait gentil d'y
    jeter un coup d'oeil.
9   Merci!
10  ================================
11  Martin Quidam
12  Labo. Incognito
13  quidam@iro.umontreal.ca
14  Etudiant de deuxième cycle
15  Universite de Montreal
16  ================================
```

FIG. 6.1 – *Extraction d'information manuelle*

```
te_message([
            [message_priorite:normal],
            [message_classe:probleme],
            [message_categorie:imprimante],
            [message_expediteur:usager],
            [message_destinaire:support],
            [message_repondre_a:usager]
          ]).
te_imprimante([
              [imprimante_etat:probleme]
            ]).
te_usager([
            [usager_nom:['Quidam','Martin']],
            [usager_e_mail:['quidam@iro.umontreal.ca']],
            [usager_labo:incognito]
          ]).
te_fichier([
             [fichier_format_actuel:format_ps],
             [fichier_nom:['~quidam/Pub/proceedings.ps']],
             [fichier_job:['1648-1']]
           ]).
te_machine([
           ]).
te_action([
            [action_nom:['Ghostview']],
            [action_message:[]]
          ]).
te_action([
            [action_nom:['FrameMaker']],
            [action_message:[]]
          ]).
te_action([
            [action_message:['Your', 'printer','job',
            '(1648-1)','could', 'not','be', 'printed']]
          ]).
te_action([
            [action_message:[ 'Messages', 'from', 'output',
            'filter:', '/usr/local/bin/PostScript/postio',
            ':', 'PostScript', 'Error']]
          ]).
```

FIG. 6.2 – *Patron d'information*

Nom de catégorie	*Sens*
COR-EGAL	Correcte, équivalente à la réponse originale
COR-DIFF	Correcte, différente de la réponse originale
COR-TROP	Correcte, plus d'information que la réponse originale
COR-PEU	Correcte, moins d'information que la réponse originale
INC	Incorrecte
INC-GEN	Incorrecte, trop général

TAB. 6.1 – *Catégories de correction*

Catégorie de réponse	Évaluateur 1		Évaluateur 2		Moyenne	
	compte	%	compte	%	compte	%
COR-EGAL	35	50.7	36	56.5	35.5	***51.4***
COR-DIFF	7	10.2	5	7.3	6.0	8.7
COR-TROP	1	1.5	6.33	9.2	3.7	5.3
COR-PEU	1	1.5	0.33	0.5	0.7	1.0
Total correcte	44	63.8	47.67	69.1	45.9	***66.5***
INC	9	13.0	9.33	13.5	9.2	13.3
INC-GEN	16	23.2	12	17.4	14.0	20.3
Total incorrecte	25	36.2	21.33	30.9	23.2	***33.5***
Grand total					69	100

TAB. 6.2 – *Évaluation de la génération de réponses*

le système génère des réponses pratiquement identiques aux messages originaux du corpus d'entraînement. Deux évaluateurs internes ont ensuite examiné les réponses générées par le programme à partir des données de test. La directive première était d'évaluer le contenu des réponses et non leur qualité rédactionnelle. Les évaluateurs se sont servi des catégories de correction telles que définies dans le tableau 6.1. Les résultats, présentés dans le tableau 6.2, démontrent que les deux tiers des cas sont traités correctement et que plus de la moitié des cas résultent en des réponses à toutes fins pratiques identiques aux réponses originales dans le corpus de test.

Les figures 6.3 à 6.8 donnent un exemple des réponses générées dans chacune des catégories d'évaluation.

À première vue, un taux d'erreur de 33.5 % parait élevé. Un examen plus attentif des résultats révèle qu'en fait seulement 13.3 % sont véritablement des erreurs de sélection de réponse, résultant possiblement en un client insatifait ou confus. Le reste (33.5 % - 13.3 % = 20.3 %) sont des erreurs moins graves du point de vue du client, puisqu'elles sont de nature générique et générale. Ce qui est en train de se produire est que la recherche dans la base de cas aboutit sur le cas de défaut "Faire suivre au préposé". Dans un vrai système de

réponse automatique, on peut supposer que ce pourcentage d'erreurs INC-GEN diminuerait au fur et à mesure que la base de cas serait mise à jour. Avec notre prototype par exemple, on aurait pu ajouter tous les cas paraîssant seulement dans le corpus de test à la base de cas contenant alors déjà les cas du corpus d'entraînement. Ceci aurait eu comme effet d'éliminer la classe d'erreurs INC-GEN, pour donner un taux de réponses correctes frôlant les 85 %.

6.3 Performance

Dans notre prototype, comme la génération de texte se fait par des techniques de surface, il n'y a pas de grandes demandes faites aux ressources computationnelles. La plus grande partie des opérations machine concerne le méchanisme d'unification de Prolog ainsi que la gestion des entrées et des sorties. Il n'y a presqu'aucune opération numérique. La génération de tout les 69 messages de test prend (à partir des données "extraites" à la main) moins de trente secondes sur un PC avec processeur Pentium III cadencé à 450 Mhz.

Même s'il y a des différences dans la complexité des problèmes traités et des techniques utilisées entre notre système et d'autres, les temps de génération de ceux-çi restent comparable: ils sont de l'ordre de secondes par réponse. Le système pour la génération de lettre d'affaires personnalisées "La Redoute" de Coch peut générer une lettre (15–20 phrases) en 2.5 secondes [Coch et al., 1995] (en supposant que les paramètres de génération sont déjà spécifiés). Le débit de génération est donc de l'ordre de milliers de lettres par heure[2]. Ce système compense pour la lenteur de son moteur de génération plus complexe (AlethGen) par son implantation en C++ (un langage compilé). Dans [Busemann et al., 2000], on nous dit que le traitement manuel (lecture du courriel du client et ensuite la sélection et la composition d'une réponse) prends environs deux minutes en utilisant le système de gestion de courriel AOL. On estime qu'avec l'automatisation des étapes de classification des messages et de composition (semi-automatique) des réponses, ce temps de traitement serait réduit de moitié, ou à environ 50–70 secondes par message. Le temps pour la composante de génération de texte est donc bien en déçà d'une minute (et probablement en bas de 10 secondes). On peut donc conclure que les ressources nécessaires pour la génération sont négligeables par rapport à ceux des autres modules d'un système de réponse automatique.

2. Communication personnel avec J. Coch, juillet 2000

Question originale:

```
Subject: hp3252
Cette imprimante n'a plus d'encre.
Jan Hagnau
```

Réponse automatique:

```
Salut Jan,
Un technicien viendra changer la cartouche d'encre de l'imprimante hp3252
sous peu.
Merci.
--
Support Technique
```

Réponse originale:

```
Subject: Re: hp3252
Bonjour,
La cartouche d'encre a été changée. Donc tout en maintenant en ordre.
Salutations!
Support.
Universite de Montreal, I.R.O., C.P. 6128, Succ. Centre-Ville,
```

FIG. 6.3 – *Exemple de réponse COR-EGAL*

Question originale:

```
Subject: Serrures a combinaison
Bonjour,
Je suis etudiant dans le cadre du projet colibri. Lorce que j'ai demande les
codes de combinaisons a la secretaire elle les avait deja mis a la corbeil.
J'apprecirais avoir les combinaisons pour demain matin.
Merci.
Marc Dupont
```

Réponse automatique:

```
Salut Marc,
Il y a des combinaisons aux serrures des portes 2248, 3252
(salles d'imprimantes), 3248 (salle de stations de travail).
Veuillez obtenir ces combinaisons auprès de votre
superviseur ou tuteur.
Bonne chance!
--
Support Technique
```

Réponse originale:

```
Subject: Re: Serrures a combinaison...
Bonjour,
Vous devez consulter votre co-administrateur.
Salutations!
Support.
Universite de Montreal, I.R.O., C.P. 6128, Succ. Centre-Ville,
```

FIG. 6.4 – *Exemple de réponse COR-DIFF*

Question originale:

```
Comment fait-on pour que le recto et le verso soient dans le meme sens de la
feuille quand on fait lpr?
```

Réponse automatique:

```
Bonjour,
L'impression d'un fichier texte s'effectue à raison de 2 pages
par côté de feuille sur toutes les imprimantes. Par surcroît, les
imprimantes hp2248, hp3252 et hp2153 impriment recto-verso par défaut.
Pour imprimer en format recto seulement faites : lpr -P <imprimante>-i-1
<fichier>.
Cette commande ne fonctionne que sur les fichiers en
format texte ou en format PostScript structuré.
Nous considerons que ce mode ne doit être utilisé que
pour la copie FINALE d'une thèse, d'un mémoire ou d'un texte
pour publication.
Le fichier de comptabilité des impressions conserve
une marque pour chaque utilisation du mode recto.
Voyez www.URL/lpr/#recto pour plus d'info.
Bonne chance!
--
Support Technique
```

Réponse originale:

```
Subject: Re: lpr
Salut!
D'après la doc, l'option -Zsduplex (ou encore -Zlduplex si j'ai mal compris
la doc) fait ce que tu veux.
donc,
lpr -Zsduplex
Support technique.
Universite de Montreal, I.R.O., C.P. 6128, Succ. Centre-Ville,
```

FIG. 6.5 – *Exemple de réponse COR-TROP*

Question originale:

```
Subject: Impression sur hp2248
L'impression a partir de mon MAC sur hp2248 en passant par NIL ne fonctionne
toujours pas.
- Sur NIL, je peut voir ma job passer dans la queue de hp2248 et se faire
'traiter' mais rien ne sort.
- Je n'ai pas de problemes a imprimer sur lw2248
Prof. Tournesol
Dep. informatique et ro
Universite de Montreal
Tel: (514) 343-6111
```

Réponse automatique:

```
Salut Prof,
Un prépose vérifiera la file pour l'imprimante hp2248 bientôt.
Merci.
--
Support Technique
```

Réponse originale:

```
Subject: Re: Impression sur hp2248
Bonjour,
Vous n'êtes pas le premier qui éprouve des problèmes. .. Cela dit le dossier
progresse.
D'une part je viens tout juste de compiler une nouvelle version de Aufs dans
laquelle il y a un support pour un nouveau pilote d'imprimante. Je dois
faire des tests d'ici peu avec M. Saint-Gapour notamment pour FrameMaker.
...je termine rapidement...puisque l'on s'est vu tout à l'heure...
Salutations!
Support technique.
Universite de Montreal, I.R.O., C.P. 6128, Succ. Centre-Ville,
```

FIG. 6.6 – *Exemple de réponse COR-PEU*

Question originale:

```
Salut,
Un post-doc de Jorge Cuanta (Stephane duLac, dulacs@iro) desire utiliser
l'imprimante hpc2153. Le systeme lui repond:
Can not acces...
J'aimerais savoir qui a acces a cette imprimante et comment en obtient le
droit?
```

Réponse automatique:

```
Benoit,
Pour voir la file d'impression, faites la commande
lpq -Pnom_d_imprimante.
Pour annuler l'impression d'un fichier, faites la commande
lprm -Pnom_d_imprimante no_de_job.
Merci.
--
Support Technique
```

Réponse originale:

```
Subject: Re: hpc2153
Salut,
Cette imprimante est sur yamaska (accès secrétariat + profs + coadm). Il
devra passer par toi pour imprimer.
Support technique.
Universite de Montreal, I.R.O., C.P. 6128, Succ. Centre-Ville,
```

FIG. 6.7 – *Exemple de réponse INC*

Question originale:

```
Subject: imprimante
Bonjour,
Pourrais-je savoir comment n'imprimer qu'un nombre restreint de pages d'un
fichier PostScript, les pages 11 a 23 par exemple.
Merci,
Pascal Massalia
--
Pascal Massalia
massalia@iro.umontreal.ca
DIRO
Universite de Montreal
CP 6128 Succursale Centre-Ville
Montreal, Canada H3C 3J7
```

Réponse automatique:

```
Pascal,
Un prépose repondra à votre problème d'impression d'ici peu.
Cordialement,
--
Support Technique
```

Réponse originale:

```
Subject: Re: imprimante
Bonjour,
Essayez avec ghostview, il faut sélectionner les pages à imprimer.
Salutations!
Support technique.
Universite de Montreal, I.R.O., C.P. 6128, Succ. Centre-Ville,
```

FIG. 6.8 – *Exemple de réponse INC-GEN*

Chapitre 7

Conclusion

Nous constatons que l'implantation de systèmes de génération de surface est essentiellement un défi technique. Il n'y a pas eu de défis théoriques à relever et aucune question scientifique fondamentale n'a été soulevée avec cette approche. Les techniques de génération profonde seraient, par leur complexité même, un frein à l'insertion de l'IA dans des contextes plus commerciaux. La solution que nous avons retenue et implantée serait pratique et utile dans l'industrie. Dans l'état actuel des connaissances en génération, notre approche de surface pourrait satisfaire à certains besoins commerciaux de façon simple, rapide et compréhensible.

7.1 Ingénierie des connaissances

Comme c'est presque toujours le cas dans l'approche basée sur l'ingénierie des connaissances, presque toutes les ressources de notre système sont spécifiques au domaine de l'application. À part des règles d'extraction pour les locaux, les noms de personnes et quelques expression fixes, tous les éléments sont spécifiques au domaine des imprimantes, et plus encore, ils sont spécifiques au domaine des *problèmes d'imprimante de notre département d'informatique*. Les ressources sont donc difficilement réutilisables. Il est clair que cette approche d'ingénierie des connaissances est, d'un certain point de vue, très inefficace. Par contre, il faut mettre cette problématique en perspective. Faire ce travail d'ingénierie fastidieux peut quand même être rentable car il est fait une seule fois et car le système résultant peut apporter d'importants gains en productivité. Ceci est encore plus vrai si le volume de messages est élevé et si le traitement manuel des ces messages est quasi–impossible. Afin d'assurer la rentabilité globale de ce genre de système, sa configuration initiale doit se faire d'une façon efficace.

7.2 Sélection de réponse

Nous avons vu que le sélecteur de réponse, basé sur une liste de cas et des tests sur les données dans les patrons d'extraction, fonctionne très bien: 66 % des messages sont répondus correctement et un autre 20 % des messages reçoivent un accusé de réception sans conséquence majeure. Nous considérons que c'est un très bon résultat étant donnée la taille restreinte du corpus d'entraînement et la technique simple utilisée. Les résultats nous laisse espérer des performances semblables sinon meilleures lors de l'utilisation d'un grand corpus. Le nombre limité de cas dans notre corpus nous a permis de faire la construction manuelle de la base de cas en Prolog. Cette approche ne serait pas viable avec un très grand nombre de cas ou lorsque le domaine de discours est plus large – la tâche deviendrait rapidement ingérable. Nous croyons que des techniques d'apprentissage automatique, par exemple l'induction d'arbres de décision et la classification de textes, pourraient alors devenir utiles. Dans toute cette discussion, par contre, il ne faut pas oublier que toutes les données qui ont servi comme entrée à notre générateur ont été préparées à la main. Nous avons supposé que l'étape d'EI est faisable et que les données extraites ressemblent à nos données "cuisinées". La réalité est peut-être toute autre.

L'optimisation de la sélection de réponses pourrait consister en un processus itératif et interactif visant la construction d'une base de cas qui donne la meilleure couverture des messages venant des usagers. Il suffirait de rajouter les nouveaux cas au fur et à mesure qu'ils se présentent (tout en faisant attention aux problèmes de chevauchements de classes et de règles ainsi qu'au pullulement de ces dernières). Il est important que les systèmes de gestion de la clientèle soient biaisés vers la *précision* et non vers le *rappel*. Il vaut mieux traiter les messages non-répertoriés dans la base de cas comme inconnus plutôt que d'estimer ou de deviner leur sens par quelque algorithme automatique que se soit. C'est le comportement désiré: on ne veut pas faire une classification approximative des réponses à partir de réponses/cas déjà vus. Il vaut mieux identifier les messages comme nouveaux et problématiques et les transférer à un préposé pour un traitement manuel. Des techniques de recherche d'information pourraient être utilisées pour ce problème de présélection de candidats.

7.3 Patrons à trous: pratiques, efficaces mais décevants

Malgré le grand nombre de recherches qui ont été faites sur la génération profonde, il n'est pas certain que la complexité de cette approche en vaille la peine. La génération de surface est donc perçue comme un compromis raisonnable étant donné l'état présent en génération. Nous sommes par contre un peu déçu de l'approche de surface, tout comme Reiter disait

l'être pour la partie syntaxique du système STOP [Reiter, 1999]. Cet aspect du projet laisse à désirer au niveau scientifique car il semble un peu simpliste.

Pour éviter cette situation dans l'avenir, l'auteur souhaite qu'un jour un groupe de TALN développe une composante de réalisation linguistique avec une large couverture des phénomènes linguistiques, prenant la forme d'un artéfact logiciel bien conçu et documenté. Ceci permettrait des projets comme le nôtre d'utiliser des techniques profondes sans trop de risques de s'embourber dans les détails, comme des milliers de règles grammaticales et des jeux de paramètres interdépendants. Par exemple, il serait très utile d'avoir des classes Java ou C++ implantant un moteur linguistique (pour l'analyse et la génération), son fonctionnement profond restant transparent au programmeur d'applications. On pourrait alors s'en servir pour simplifier l'implantation de la génération hybride et les patrons simples. Par exemple, on pourrait créer une forme de pre-processeur pour les messages d'erreur, le moteur les convertissant automatiquement en patrons hybrides. En conclusion, il y aurait plein d'opportunités pour l'utilisation de la génération profonde, en autant que c'est facile et transparent.

7.4 Travaux futurs

7.4.1 Génération de texte

Il serait possible d'éliminer certaines tâches d'ingénierie des connaissances par l'analyse automatique de corpus. On pense à la découverte de terminologie spécifique au domaine de discours, la construction d'ontologies de concepts, et la découverte d'expressions et autres segments de textes utiles. Une fois ces ressources extraites, on pourrait plus facilement construire les patrons pour les textes à générer.

Si on suppose que des artéfacts logiciels (modules, fonctions, programmes) pour le traitement de la langue naturelle deviennent disponibles au cours des prochaines années (elles ne le sont pas présentement), un parsage bi-directionnel ("reversible grammars") pour la découverte et la regénération de patrons par exemplars [Bateman et Henschel, 1999] [Cancedda, 1999] serait très utile. L'apprentissage par recherche d'explication pour le parsage [Uszkoreit et al., 1994] pourrait aussi s'appliquer à la génération. Bateman et Henschel ont déjà fait des expériences dans la spécialisation automatique de grammaires [Bateman et Henschel, 1999] et dans l'extraction de sous-langages spécifiques aux domaines d'application [Henschel et Bateman, 1997]. L'effet de ses deux techniques est d'élaguer les parties d'une grammaire à couverture large qui ne sont jamais utilisées dans les textes de l'application. Tout en gardant des bases solides d'une grammaire complète, on obtient des

gains considérables en termes de performance. Rayner et Carter [Rayner et Carter, 1996] vont dans le sens du parsage en spécialisant une grammaire complexe avec un corpus d'entraînement.

Pour les rares fois où ça se produit, les questions multiples dans un même message mériteraient des réponses multiples [McKeown et Radev, 1995]. Pour ce faire, il faudrait trouver des mécanismes pour générer des expressions référentielles adéquates et pour faire la coordination de réponses séparées pour en faire un texte cohérent. On pourrait faire appel à l'aggrégation, par exemple.

7.4.2 Gestion de courriel et réponse automatique

Les autres étapes de traitement dans un système de gestion de courriel et de réponse automatique sont des avenues de recherche intéressantes. Il y a beaucoup de possibilités pour la réduction des coûts pour la configuration et l'opération de ce type de système grâce à l'utilisation des techniques de pointe en informatique et en apprentissage machine:

Classification La classification automatique des messages [Busemann et al., 2000] via les SVM ("Support Vector Machines") ou le boosting des arbres de décision serait intéressante dans un contexte où il y a un grand volume de messages.

Règles d'extraction Il y a plusieurs techniques pour l'induction automatique de règles d'extraction (par exemple, Ripper, AutoSlog, WAVE) ou semi-interactive (par exemple, Asium et Intex). Riloff [Riloff et Lehnert, 1994] propose même que les règles d'extraction seraient une base pour la classification de haute précision.

Interfaces conviviales Des interfaces graphiques conviviales destinées aux préposés humains (utilisant Java et XML, par exemple) aideraient grandement à l'acceptation d'un système (semi–)automatique. Les tâches de paufinage de la classification, de configuration des règles d'extraction, de préparation des patrons de réponses, et de modification manuelle des patrons de réponses instanciés automatiquement pourrait être rendues très efficaces par des interfaces biens conçues.

Gestion de dialogue La gestion de dialogue est compliquée et elle est difficile à automatiser. Par contre, avec les éléments de gestion et les interfaces conviviales mentionnés ci-haut, on devrait être en mesure d'ajouter une fonctionalité de gestion *semi–automatique* de dialogues. Si un grand corpus de *conversations* bien structurées était accumulé, on aurait les bases pour l'automatisation de certaines parties des dialogues via des algorithmes d'apprentissage et des analyses syntaxiques et sémantiques profondes. Ça serait un pas vers la gestion autonome des dialogues, la clé de voûte pour du Gestion des relations avec la clientèle ("Customer Relationship Management") de l'avenir.

7.5 Perspectives commerciales

Nous allons conclure avec quelques réflexions sur la viabilité du traitement automatique des courriels dans un contexte commercial. Peu importe la couverture et la précision d'un système de réponse automatique, il restera toujours des messages qui échapperont à l'analyse automatique. Il y aura toujours des situations où l'ordinateur n'a tout simplement pas les connaissances ni les méchanismes de raisonnement pour y répondre adéquatement. Il aura aussi des cas ambigus. C'est une mauvaise stratégie d'envoyer des réponses automatiques à peu près correctes; il faut être prudent pour ne pas perdre des clients. Ces considérations auront pour effet de réduire la couverture automatique des messages et d'augmenter le nombre de cas qu'il faut traiter manuellement. Un humain sera donc toujours nécessaire et celui-ci devra lire et comprendre les nouveaux messages afin de formuler des nouveaux patrons de réponse adéquats. Ainsi, la couverture automatique du système s'élargira au fur et à mesure que de nouveaux messages se présenteront et seront traités manuellement.

Si un système automatique peut traiter un pourcentage important des messages (par exemple 80 %), le reste sera traité manuellement par un préposé humain (ou une équipe) stimulé et non abruti par un flot constant de messages de routine. Lorsque le volume de messages augmente, des techniques plus sophistiquées de traitement de la langue naturelle (par exemple, la classification, l'induction automatique de règles d'EI, et la sélection probabiliste de plusieurs réponses possibles pour un seul message) peuvent être appliquées pour ramener le pourcentage de traitement manuel en bas d'un seuil de rentabilité globale. C'est une façon de dire que les gains en productivité et la réduction des coûts par l'utilisation de ces techniques sophistiquées ne valent peut être pas toujours la peine. Il faudra toujours des préposés pour répondre aux exceptions, pour faire l'entretien du système, pour établir les seuils de confiance aux processus automatiques et pour maintenir des relations humaines avec les clients. Mais il est aussi important d'utiliser ses ressources humaines à 100% de leur capacité. Il faut donc trouver le juste millieu entre la fiabilité, la complexité et le coût du traitement automatique et ces même paramètres pour l'intervention manuelle.

Bibliographie

[AECMA, 1986] AECMA (1986). *A guide for the preparation of aircraft maintenance documentation in the international aerospace maintenance language*. AECMA, Slack Lane, Derby, UK, édition BDC Publishing Services.

[Appelt, 1985] Appelt, D. (1985). Planning English Referring Expressions. *Artificial Intelligence*, 26:1–33.

[Appelt et al., 1993] Appelt, D. E., Hobbs, J. R., Bear, J., Israel, D., Kameyama, M. et Tyson, M. (1993). The SRI MUC–5 JV–FASTUS information extraction system. Dans *Proceedings of the Fifth Message Understanding Conference (MUC–5)*, Baltimore, Maryland.

[Bateman et Henschel, 1999] Bateman, J. et Henschel, R. (1999). From full generation to 'near-templates' without loosing generality. Dans *Proceedings of the KI-99 Workshop for Artificial Intelligence: May I Speak Freely: Between Templates and Free Choice in Natural Language Generation*, Bonn, Germany.

[Becker, 1975] Becker, J. D. (1975). The phrasal lexicon. Dans Schank, R. C. et Webber, B. L., éditeurs, *Theoretical Issues in Natural Language Processing (*TINLAP*): 1*, pages 70–73. Cambridge, MA.

[Becker et Busemann, 1999] Becker, T. et Busemann, S. (1999). May I Speak Freely? Between Templates and Free Choice in Natural Language Generation. Document D-99-01, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany.

[Boufaden, 1999] Boufaden, N. (1999). Les systèmes d'extraction d'information. Rapport technique, Laboratoire RALI, Université de Montréal, Montréal, Québec, Canada.

[Buseman, 1999] Buseman, S. (1999). Natural language generation. Page Web http://www.dfki.de/fluids/Natural_Language_Generation.html.

[Busemann, 1993] Busemann, S. (1993). Towards Configurable Generation Systems: Some Initial Ideas. Dans Busemann, S. et Harbusch, K., éditeurs, *Proceedings of the DFKI Workshop on Natural Language Systems: Re-usability and Modularity*, Document D-93-03, pages

57–64. Deutsches Forschungszentrum für künstliche Intelligenz (DFKI), Saarbrücken, Germany.

[Busemann, 1998] Busemann, S. (1998). A shallow formalism for defining personalized text. Dans *Workshop* Professionelle Erstellung von Papier- und Online-Dokumenten: Perspektiven für die automatische Textgenerierung *at the 22nd Annual German Conference on Artificial Intelligence (KI-98)*, Bremen, Germany.

[Busemann et al., 1997] Busemann, S., Declerck, T., Diagne, A. K., Dini, L., Klein, J. et Schmeier, S. (1997). Natural language dialogue service for appointment scheduling agents. Rapport de recherche RR-97-02, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Saarbrücken, Germany.

[Busemann et Horacek, 1997] Busemann, S. et Horacek, H. (1997). Generating air quality reports from environmental data. Rapport technique, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Saarbrücken, Germany.

[Busemann et al., 1994] Busemann, S., Oepen, S., Hinkelman, E. A., Neumann, G. et Uszkoreit, H. (1994). COSMA – multi-participant NL interaction for appointment scheduling. Rapport de recherche RR-94-34, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Saarbrücken, Germany.

[Busemann et al., 2000] Busemann, S., Schmeier, S. et Arens, R. G. (2000). Message classification in the call center. Dans *Proceedings of the* 6^{th} *Conference on Applied Natural Language Processing (ANLP–NAACL2000)*, Seattle, WA.

[Cancedda, 1999] Cancedda, N. (1999). Text generation from MUC templates. Dans *Proceedings of the 1999 European Workshop on Natural Language Generation*. To appear.

[Cancedda et al., 1997a] Cancedda, N., Kamstrup, G., Pianta, E. et Prietrosanti, E. (1997a). A hybrid approach to hypertext generation. Dans *Proceedings of the Fifth Congress of the Italian Association for Artificial Intelligence (AI*AI '97)*, Rome, Italy.

[Cancedda et al., 1997b] Cancedda, N., Kamstrup, G., Pianta, E. et Prietrosanti, E. (1997b). SAX: Generating hypertext from SADT models. Dans *Proceedings of the Third Workshop on Applications of Natural Language to Information Systems (NLDB '97)*, Vancouver, Canada.

[Coch, 1998] Coch, J. (1998). Applications industrielles de la génération: Pourquoi et comment. *Traitement automatique des langues*, 39(2):89–105.

[Coch et al., 1995] Coch, J., David, R. et Magnoler, J. (1995). Quality test for a mail generation system. Dans *Proceedings of Linguistic Engineering '95*, Montpellier, France.

[Contant, 1985] Contant, C. (1985). Génération automatique de texte: application au sous-langage boursier français. Mémoire de maîtrise, Université de Montréal, Montréal, Québec, Canada.

[Dale, 1995] Dale, R. (1995). An introduction to natural language generation. Dans *European Summer School in Logic, Language and Information, ESSLLI'95*, Barcelona, Spain.

[Dalianis et Hovy, 1996] Dalianis, H. et Hovy, E. (1996). Aggregation in natural language generation. Dans Adorni, G. et Zock, M., éditeurs, *Trends in natural language generation: an artificial intelligence perspective*, numéro 1036 dans Lecture Notes in Artificial Intelligence, pages 88–105. Springer-Verlag.

[Elhadad, 1993] Elhadad, M. (1993). *FUF: The Universal Unifier - User Manual Version 5.2*. Department of Computer Science, Ben Gurion University of the Negev, Israel.

[Gaizauskas et Wilks, 1998] Gaizauskas, R. et Wilks, Y. (1998). Information extraction: beyond document retrieval. *Journal of Documentation*, 54(1):70–105.

[Gedalia et Elhadad, 1996] Gedalia, R. et Elhadad, M. (1996). CLINT - a hybrid template/word-based text generator. http://www.cs.bgu.ac.il/~elhadad/clint.html.

[Geldof et Van de Velde, 1997] Geldof, S. et Van de Velde, W. (1997). An architecture for template based (hyper)text generation. Dans *Proceedings of the* 6^{th} *European Workshop on Natural Language Generation (EWNLG'97)*, pages 28–37, Duisburg, Germany. Inst. fuer Informatik, Gerhard Mercator Universitaet.

[Henschel et Bateman, 1997] Henschel, R. et Bateman, J. (1997). Application-driven automatic subgrammar extraction. Dans *ACL–97/EACL–97 Workshop: ENVGRAM: Computational Environments for Grammar Development and Linguistic Engineering*, Madrid, Spain. Association for Computational Linguistics. E-Print Archive: cmp-lg/9711010.

[Horacek] Horacek, H. *An Integrated View of Text Planning*, pages 29–44.

[Horacek et Busemann, 1998] Horacek, H. et Busemann, S. (1998). Towards a methodology for developing application-oriented report generation. Dans *Lecture Notes in Computer Science*, volume 1504, pages 189–?? Springer–Verlag, Heidelberg, Germany.

[Hovy, 1993] Hovy, E. (1993). Automated Discourse Generation Using Discourse Structure Relations. *Artificial Intelligence*, 63(1–2):341–385.

[Hovy, 1988] Hovy, E. H. (1988). *Generating Natural Language under Pragmatic Constraints*. Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey.

[Impacts–2000, 2000] Impacts–2000 (2000). *SIGGEN Conference Workshop: Impacts in Natural Language Generation*, Schloss Dagstuhl, Saarland, Germany.

[Kosseim, 1999] Kosseim, L. (1999). Systèmes de réponse automatique: État de l'art. Rapport technique, Laboratoire RALI, Université de Montréal, Montréal, Québec, Canada.

[Kosseim et Lapalme, 1998] Kosseim, L. et Lapalme, G. (1998). EXIBUM: un système expérimental d'extraction bilingue. Dans *Actes des rencontres internationales sur l'extraction, le filtrage et le résumé automatique (RIFRA '98)*, pages 129–140, Sfax, Tunisia.

[Kosseim et Lapalme, 2000] Kosseim, L. et Lapalme, G. (2000). Choosing rhetorical structures to plan instructional texts. *Computational Intelligence: An International Journal*, 16(3):408–445. Blackwell. Boston.

[Mann et Thompson, 1986] Mann, W. et Thompson, S. (1986). Rhetorical Structure Theory: Description and Construction of Text Structures. Dans *Proceedings of the* 3^{th} *International Workshop on Text Generation*, Nijmegen, Pays-Bas.

[Matthiessen et Bateman, 1991] Matthiessen, C. et Bateman, J. (1991). *Text Generation and Systemic–Functional Linguistics: Experiences from English an Japanese*. Communication in Artificial Intelligence. Pinter Plublishers, London.

[McKeown, 1985] McKeown, K. (1985). *Text generation: Using discourse strategies and focus constraints to generate natural language text*. Studies in Natural Language Processing. Cambridge University Press, Cambridge, UK.

[McKeown et Radev, 1995] McKeown, K. et Radev, D. R. (1995). Generating summaries of multiple news articles. Dans *Proceedings, ACM Conference on Research and Development in Information Retrieval SIGIR'95*, Seatle, WA.

[Milosavljevic et al., 1996] Milosavljevic, M., Tulloch, A. et Dale, R. (1996). Text generation in a dynamic hypertext environment. Dans *Nineteenth Australasian Computer Science Conference (ACSC'96)*, Melbourne, Australia. http://www.ics.mq.edu.au/~mariam/papers/acsc96/.

[Neumann, 1997] Neumann, G. (1997). Applying explanation-based learning to control and speeding-up natural language generation. Dans 35^{th} *Annual Meeting of the Association for Computational Linguistics /* 8^{th} *Conference of the European Chapter of the Association for Computational Linguistics*, Madrid, Spain.

[Neumann, 1998] Neumann, G. (1998). Interleaving natural language parsing and generation through uniform processing. *Artificial Intelligence*, 99:121–163.

[Neumann et Flickinger, 1999] Neumann, G. et Flickinger, D. (1999). Learning stochastic lexicalized tree grammars from HPSG. Rapport technique, Saarbrücken, Germany.

[Pazienza, 1997] Pazienza, M. T., éditeur (1997). *Information extraction: a multidisciplinary approach to an emerging information technology*. Lecture Notes in Computer Science. Springer–Verlag, Heidelberg, Germany.

[Quinlan, 1993] Quinlan, J. R. (1993). *C4.5:Programs for Machine Learning*. Morgan Kaufmann Publishers.

[Rayner et Carter, 1996] Rayner, M. et Carter, D. (1996). Fast parsing using pruning and grammar specialization. Dans *Proceedings of ACL'96*, University of California, Sant Cruz, California, USA.

[Reiter, 1994] Reiter, E. (1994). Has a Consensus NL Generation Architecture Appeared, and is it Psycholinguistically Plausible? Dans *Proceedings of the* 7^{th} *International Workshop on Natural Language Generation*, pages 163–170, Kennebunkport, Maine.

[Reiter, 1995] Reiter, E. (1995). NLG vs. Templates. Dans *Proceedings of the Fifth European Workshop on Natural-Language Generation (ENLGW-1995)*, Leiden, The Netherlands.

[Reiter, 1999] Reiter, E. (1999). Shallow vs. Deep Techniques for Handling Linguistic Constraints and Optimisations. Dans *Proceedings of the KI-99 Workshop for Artificial Intelligence: May I Speak Freely: Between Templates and Free Choice in Natural Language Generation*, Bonn, Germany.

[Reiter et Dale, 2000] Reiter, E. et Dale, R. (2000). *Building Applied Natural Language Generation Systems*. Cambridge University Press, Cambridge, UK.

[Reiter et Mellish, 1993] Reiter, E. et Mellish, C. (1993). Optimizing the costs and benefits of natural language generation. Dans *Proceedings of the* 13^{th} *International Joint Conference on Artificial Intelligence (IJCAI-1993)*, volume 2, pages 1164 – 1169, San Mateo, California. Morgan Kaufmann.

[Reiter et al., 1995] Reiter, E., Mellish, C. et Levine, J. (1995). Automatic Generation of Technical Documentation. *Applied Artificial Intelligence*, 9.

[Riloff et Lehnert, 1994] Riloff, E. et Lehnert, W. (1994). Information extraction as a basis for high-precision text classification. *ACM Transaction on Information Systems*, 12(3):296–333.

[TREC, 1999] TREC (1999). Dans *Proceedings of The Eighth Text REtreival Conference (TREC–8)*, Gaithersburg, Maryland.

[US Department of Defense, 1988] US Department of Defense (1988). *Military Standard DOD-STD-2167A: Defense System Software Develpment*.

[Uszkoreit et al., 1994] Uszkoreit, H., Backofen, R., Busemann, S., Diagne, A. K., Hinkelman, E. A., Kasper, W., Kiefer, B., Krieger, H.-U., Netter, K., Neumann, G., Oepen, S. et Spackman, S. P. (1994). DISCO – an HPSG-based NLP system and its application for appointment scheduling. Rapport de recherche RR-94-38, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Saarbrücken, Germany.