La programmation orientée-objet pour le développement de modèles de langages

Guy Lapalme — George Foster — Philippe Langlais

Laboratoire RALI
Département d'informatique et de recherche opérationnelle
Université de Montréal, CP 6128, Succ Centre-Ville
Montréal Québec Canada, H3C 3J7

{lapalme,foster,felipe}@iro.umontreal.ca

RÉSUMÉ. *Nous présenterons le développement de modèles probabilistes de langues naturelles que nous avons effectué dans un cadre orienté objets. Nous montrerons comment l'approche orientée objets a permis de factoriser certaines opérations et de faciliter le développement rapide de modèles alternatifs. Nous en illustrerons un résultat pratique d'application dans le cadre de* TransType*, un système d'aide à la saisie de frappe de traduction.*

ABSTRACT. *We present the implementation of statistical natural language models that we developped within an OO framework. We show how we could factor out operations and speed up the development of new models. A pratical example will be given in the context of* TransType*, a computer aided translation typing tool.*

MOTS-CLÉS : *Modèles probabilistes de langage, outils d'aide à la traduction, programmation orientée-objet*

KEYWORDS: *Statistical language models, computer aided translation tools, object oriented programming*

1. Introduction à la linguistique informatique

La linguistique cherche à caractériser et à expliquer la multitude d'observations sur la langue utilisée dans les conversations, les écrits et les média. Cette discipline porte à la fois sur le côté cognitif de l'acquisition, de la production et de la compréhension du langage ainsi que sur la compréhension de la relation entre l'expression linguistique et le monde réel.

En linguistique, on s'intéresse également à la compréhension des structures utilisées dans le langage. Pour y arriver, on peut proposer des "règles" qui définissent la bonne formation des énoncés. Toutefois, ces règles évoluent dans le temps ; les locuteurs les violent parfois par ignorance, par paresse ou pour être créatifs sans pour autant empêcher le message de passer. C'est en partie cette flexibilité qui rend la formalisation de la langue naturelle si difficile. On peut distinguer deux grandes approches en linguistique : rationaliste et empirique. La première suppose une connaissance à priori de règles linguistiques de compétence alors que la seconde s'appuie sur une capacité d'association, de reconnaissance de patrons et de généralisation à partir d'exemples. Nous allons nous concentrer ici sur cette seconde approche qui revient en force depuis le milieu des années 80. Les méthodes empiriques ont débuté dans les années vingt pour faire place, dans les années soixante, à l'approche rationaliste promue entre autres par Chomsky. La disponibilité grandissante de textes sous format électronique ainsi que des moyens de stockage et de traitement de plus en plus performants ont permis un développement accéléré de l'approche empirique au cours des dernières années. Cette approche suppose qu'on peut apprendre (voire expliquer) la structure du langage en spécifiant un modèle général dont on peut déterminer la valeur de ses paramètres à l'aide de statistiques et d'inférences de schémas sur de grandes quantités de texte.

Les modèles statistiques de langue ont l'avantage de pouvoir traiter de "vrais" textes et non pas seulement des idéalisations théoriques. Ils pourront de plus toujours fournir une réponse (parfois même plusieurs) et ils sont en général plus rapides à appliquer que les méthodes symboliques. Il faut toutefois convenir de quelques inconvénients dont le fait qu'ils sont difficiles à mettre au point et à corriger et qu'il est très difficile d'en expliquer les résultats en termes de principes de plus haut niveau qui pourraient être appréciables par l'usager.

2. Modèle de langage

Un modèle de langue probabiliste peut être présenté comme une fonction qui donne la probabilité de rencontrer un mot en fonction des mots précédents. Cette approche a déjà montré son utilité dans plusieurs applications dont la reconnaissance de la parole et de caractères, la correction de fautes d'orthographe et les systèmes de traduction automatique.

La tâche de prédiction d'un mot consiste à estimer la probabilité suivante :

$$P(w_n|w_1, w_2 \ldots w_{n-1}) \quad \text{où } w_i \text{ est le } i\text{ème mot d'un texte}$$

Par souci de simplification, on considère généralement que seul le contexte immédiat influence le prochain mot. Des raisons pratiques imposent de plus des restrictions sur l'empan du texte conservé. En principe, on aimerait que n soit grand mais alors on aurait trop de paramètres à estimer comme le montre le tableau 1 pour un vocabulaire de 20 000 mots.

	n	# de paramètres	
bigram	2	$20\,000^2$	400 millions
trigram	3	$20\,000^3$	8 trillions
4-gram	4	$20\,000^4$	1.6×10^{17}

Tableau 1. *Nombre de paramètres à estimer pour des n-gram.*

Différentes méthodes sont appliquées pour restreindre l'espace des probabilités et chercher des estimateurs à vraisemblance maximum et surtout traiter l'éparpillement des données. Manning et Schütze[MAN 99] présentent une excellente introduction au domaine.

2.1. *Exemples d'applications*

Notre laboratoire de Recherche Appliquée en Linguistique Informatique (RALI) a développé toute une gamme d'outils linguistiques innovateurs basés sur des modèles statistiques dont :

SILC qui détermine la langue et le jeu de caractères utilisés dans un texte. Pour y arriver, il dispose pour chaque couple langue/encodage connu, d'un modèle qui associe une certaine probabilité au texte soumis et des critères qui lui permettent de déterminer le modèle le plus probable. La performance de SILC est presque sans faille lorqu'on lui soumet des textes suffisamment longs (plus de 50 caractères). Le système reconnaît actuellement 25 langues différentes chacune encodée avec deux ou trois jeux de caractères différents. De plus amples informations sur SILC ainsi qu'une interface de démonstration sont disponibles à l'URL suivant :

`http://www-rali.iro.umontreal.ca/ProjetSILC.fr.html`

Réacc est un système qui introduit automatiquement les accents et autres marques diacritiques dans un texte français qui en est privé. Ce système utilise un modèle probabiliste basé sur les catégories syntaxiques possibles des mots non-accentués et permet ainsi de trouver le meilleur agencement de mots. Il y a également une version interactive de cet outil qui effectue la réaccentuation en temps réel. On peut en savoir plus et même essayer Réacc à l'URL suivant :

`http://www-rali.iro.umontreal.ca/ProjetReacc.fr.html`

3. Modèles de traduction

Un modèle de traduction peut être présenté comme la combinaison de l'information de deux modèles : un modèle de la langue cible et un modèle de correspondance entre des unités des langues source et cible. La tâche de prédiction d'un mot consiste à estimer la probabilité suivante :

$$P(t|S,T) \text{ où } \begin{cases} t \text{ est le prochain mot dans la langue cible} \\ T \text{ est la traduction partielle disponible} \\ S \text{ est le texte en langue source} \end{cases}$$

Un tel modèle est basé sur de grands ensembles de paramètres dont les valeurs sont déduites à partir d'observations sur des traductions déjà faites. Ces exemples de couples de textes source et cible sont obtenus par l'alignement de textes bilingues dont on sait que l'un est la traduction de l'autre [LAN 98].

3.1. *TransType une application novatrice des modèles de traduction*

Le RALI a développé TransType, un outil inédit d'aide à la traduction, qui propose de compléter des traductions partielles données par le traducteur. La figure 1 illustre l'utilisation de TransType pour traduire un texte de l'anglais vers le français. Le traducteur choisit une phrase à traduire dans la fenêtre du haut et commence à entrer sa traduction dans la fenêtre du bas. Après chaque caractère tapé par le traducteur, le système affiche une liste de propositions de complétion que l'utilisateur peut soit accepter (à l'aide d'une touche ou de la souris) soit ignorer en continuant à taper. Dans ce dernier cas, le système recalcule aussitôt de nouvelles suggestions de complétion de l'entrée du traducteur. Cette interface est facile à utiliser et on peut en mesurer la performance par exemple en calculant la proportion de frappes épargnées en tapant une traduction. Ainsi c'est le traducteur qui garde le contrôle, le système devant s'adapter continuellement. Cette façon innovatrice de voir un outil d'aide à la traduction diffère du contexte habituel de traduction automatique où c'est la machine qui produit une première ébauche qui doit ensuite être corrigée par le traducteur.

Nos tests nous indiquent que cette approche peut sauver environ 60% des frappes si les suggestions se limitent à proposer le prochain mot selon l'approche présentée dans Foster [FOS 97]. Nous développons actuellement une version plus élaborée [LAN 99] où TransType pourra proposer des unités plus longues qu'un seul mot pour laquelle nous avons estimé pouvoir épargner environ 75% des frappes.

Même s'il est raisonnable de penser que cette forme de complétion sera utile pour les traducteurs, nous n'avons pas encore pu vérifier cette conjecture mais nous testons actuellement notre prototype auprès de traducteurs professionnels pour valider cette approche dans le cadre du processus global de traduction.

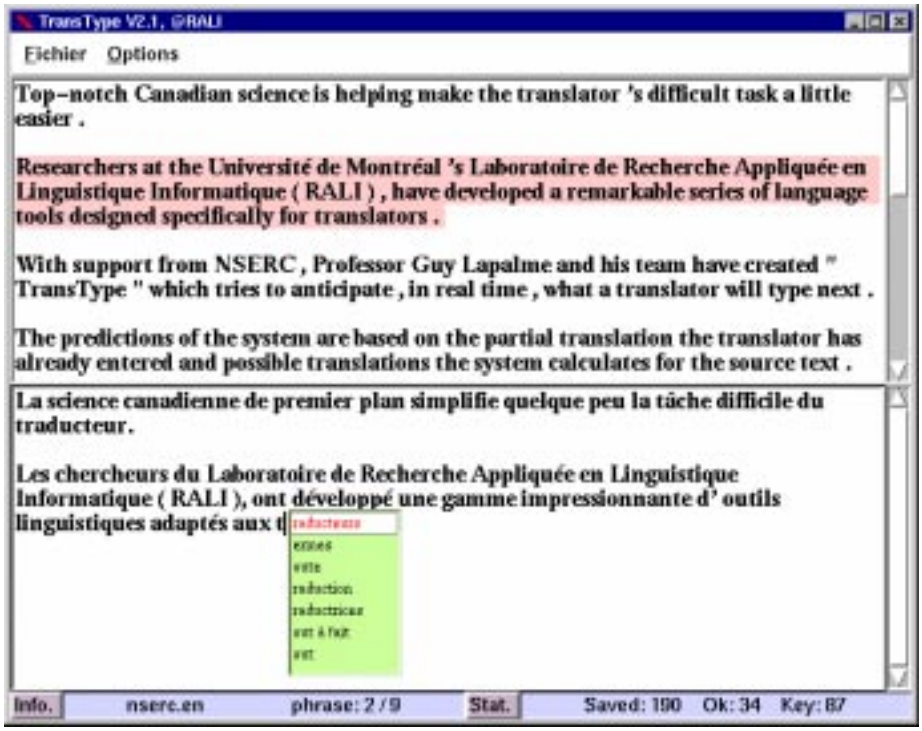

Figure 1. *Exemple d'interaction avec TransType. Le texte source se trouve dans la fenêtre du haut et la traduction est tapée dans le fenêtre du bas. Le menu qui donne les suggestions suit le curseur, il est mis à jour au fur et à mesure de l'avancée de la traduction.*

3.2. *Étapes de développement de modèles probabilistes*

Un aspect important de notre recherche est le développement et l'évaluation de différents modèles statistiques qui comprend entre autre les activités suivantes.

– développement et implantation de nouveaux modèles ;

– entraînement de modèles pour en estimer les valeurs à partir de grandes quantités de textes ;

– ajustement pour optimiser la configuration d'un modèle ; ceci peut impliquer l'ajustement manuel de variables contrôlant l'algorithme d'entraînement, la grandeur de l'espace des paramètres, etc.

– évaluation d'un nouveau modèle à partir de données de test non utilisées durant la phase d'entraînement ;

– analyse des forces et des faiblesses d'un modèle

4. Modélisation orientée-objet

Pour les besoins de TransType et d'autres outils similaires, nous avons conçu un environnement de développement de modèles de langues que nous avons nommé GLM (Generic Language Modeling) qui comprend trois composantes :

– un ensemble de classes C++ liées entre elles par un des liens d'héritage présentés dans la figure 2.

– des programmes d'entraînement et de tests

– un ensemble de modèles de langages, instances des classes présentées à la figure 2 et qui sont des combinaisons linéaires de modèles statiques et dynamiques de n-grams.

GLM comprend maintenant 27 classes réparties sur environ 4400 lignes de code auxquelles il faut ajouter environ 6400 lignes pour une librairie auxiliaire utilisée par GLM. Même si GLM a été développé pour des fins de recherche, les modèles tournent assez rapidement et peuvent traiter de grands corpus.

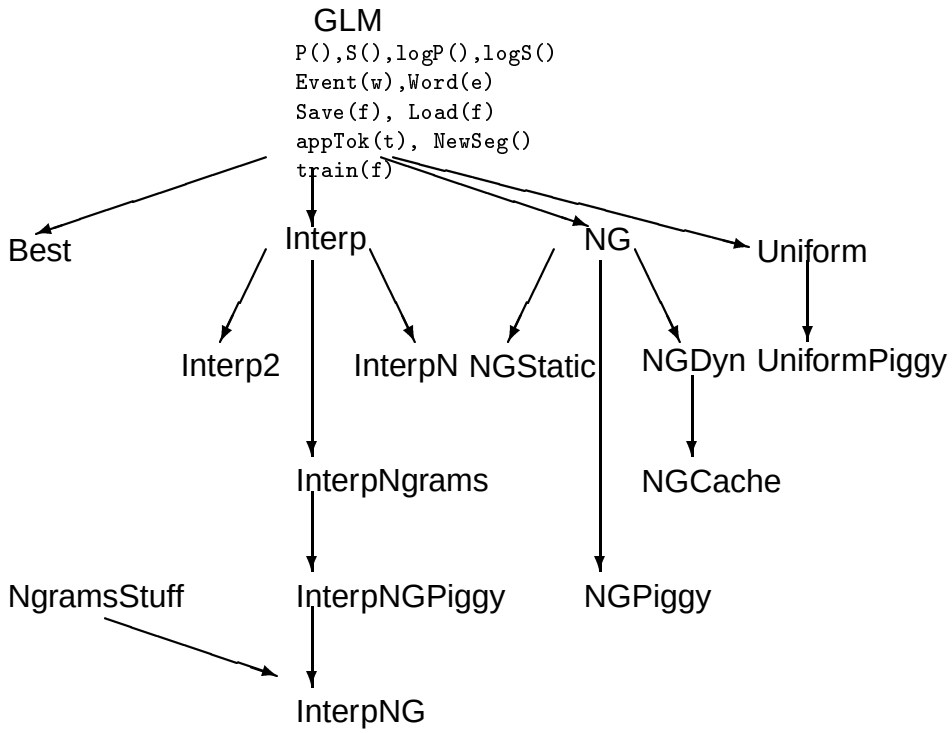

Figure 2. *Hiérarchie des classes pour les modèles statistiques*

Nous présentons maintenant quelques avantages qui nous sont fournis par cette structuration.

4.1. *Interface générique*

GLM est une interface générique d'un modèle de langage qui permet de s'abstraire des détails des modèles individuels. Ainsi on peut développer des procédures d'entraînement, d'évaluation et même TransType indépendamment des modèles eux-mêmes. Ceci minimise la quantité de code à réécrire et évite ainsi des dépendances sur des particularités de certains modèles.

Les modèles sont développés en termes d'événements plutôt qu'en termes de mots car ceux-ci pourraient être trop nombreux si on tient compte des nombres, des noms propres ou des mots inconnus.

Chaque modèle conserve son contexte qui est mis à jour de façon incrémentale par l'ajout d'un mot, l'indication de changement de phrase (frontière de phrase) ou de document. On peut aussi réinitialiser le modèle. GLM fournit également un processus d'apprentissage itératif des modèles qui termine après un certain nombre d'itérations ou lorsqu'un certain critère de convergence est atteint.

Nous avons aussi développé un "constructeur virtuel" qui permet de construire un modèle à partir d'une chaîne de caractères qui la décrit. Lorsque de nouveaux modèles sont ajoutés à la hiérarchie ou que la syntaxe d'un modèle change, il suffit de modifier ce constructeur virtuel.

Voici un bout de code qui illustre l'utilisation de GLM pour évaluer un modèle quelconque en calculant la probabilité qu'il affecte aux mots de la langue cible dans un texte bilingue :

```
GLM* glm = ConstructGLM(model_description);
double logpr = 0.0;      // log probability of test text

while (test_text.GetNextSentence(sent)) {
   glm->NewSentence();
   while (sent.GetNextWordInSentence(word)) {
      logpr += log(glm->Pr(word));
      glm->AppendWord(word);
   }
}
```

4.2. *Implantation de nouveaux modèles*

L'approche objet nous a facilité l'implantation de nouveaux modèles de plusieurs façons :

– la classe abstraite GLM donne une description précise et succincte de ce qui est requis pour implanter un nouveau modèle et permet de vérifier la présence de tout ce qui est nécessaire ;

– il est relativement facile de dériver un nouveau modèle à partir d'un autre en ajoutant seulement quelques lignes de code ;

– comme les modèles sont souvent des combinaisons de composantes orthogonales, il serait imaginable de les combiner à l'aide de l'héritage multiple quoiqu'ici nous ne l'ayons fait que très parcimonieusement. Dans la première version de GLM, nous avions utilisé l'héritage multiple mais il s'est avéré que le code devenait très compliqué. C'est pourquoi un des buts de la réorganisation présentée ici a été d'éliminer l'héritage multiple autant que possible.

4.3. *Combinaison dynamique*

Nous utilisons de manière intensive certaines propriétés dynamiques de l'héritage en programmation orientée-objet. Par exemple, il est possible de développer un modèle d'interpolation de deux autres modèles de façon complètement générale.

```
Interp::Pr(char* word) {
   return c1 * m1->Pr(word) + c2 * m2->Pr(word);
}
```

ou `m1` et `m2` sont des modèles de traduction génériques qui ont servi à construire le modèle `Interp`, et `c1` et `c2` sont des poids sur ces modèles. L'implantation dans un langage plus classique serait beaucoup plus complexe car nous dépendons de façon cruciale de l'utilisation de fonctions virtuelles.

5. Travaux connexes

Il y a très peu de systèmes comparable à GLM car la plupart du temps, les chercheurs ont développé des systèmes ad-hoc. Il faut toutefois signaler les systèmes de modélisation de langage suivants :

CMU-Cambridge Toolkit est un ensemble de programmes Unix qui peuvent être combinés avec un ensemble d'options pour créer des modèles n-grams. Il est disponible pour des fins de recherche à

`http://svr-www.eng.cam.ac.uk/~prc14/toolkit.html`

Entropic est un atelier logiciel pour construire des composantes bigrammes et trigrammes pour des sytèmes de reconnaissance de la parole. Les modèles sont assez figés et le code source n'est pas disponible. On peut l'acheter à

`http://www.entropic.com`

SRI Language Modeling Toolkit est un ensemble de classes C++ assez semblable à GLM. Il est en général plus complet que GLM sauf pour certains points qui

manquent à SRILM. Ce système a été utilisé pendant plusieurs années de façon interne à SRI et il a été récemment rendu public pour fins de recherche à

`http://www.speech.sri.com/projects/srilm`

6. Conclusion

Nous avons montré le développement d'un atelier logiciel orienté objets pour le développement de modèles statistiques de langages naturels. Cette application illustre les avantages et surtout la flexibilité que peut apporter la structuration objet dans un domaine qui est rarement associé à la programmation orientée objets.

7. Bibliographie

[FOS 97] FOSTER G., ISABELLE P., PLAMONDON P., "Target-Text Mediated Interactive Machine Translation", *Machine Translation*, vol. 12, 1997, p. 175–194.

[LAN 98] LANGLAIS P., SIMARD M., VÉRONIS J., "Methods and Practical Issues in Evaluating Alignement Techniques", *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistic*, vol. 1, Montréal, Canada, 1998, p. 711–717.

[LAN 99] LANGLAIS P., FOSTER G., LAPALME G., "Unit Completion for a Computer-aided Translation Typing System", *soumis à Machine Translation*, , 1999, page 25 p.

[MAN 99] MANNING C. D., SCHÜTZE H., *Foundations of Statistical Natural Language Processing*, MIT Press, 1999.

On peut trouver plus d'informations sur les travaux du RALI ainsi que des démonstrations à l'adresse suivante :

`http ://www-rali.iro.umontreal.ca/`